UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Especificación y desarrollo de mecanismos de interoperabilidad a nivel de Middleware y Aplicaciones/Servicios entre Plataformas Heterogéneas de Internet de las Cosas

Departamento de Comunicaciones

Universitat Politècnica de València

Tesis presentada para la obtención del grado de

Doctor por la Universitat Politècnica de València

Valencia, Febrero de 2022

Autor:
Andreu Belsa Pellicer

Directores:
Dr. Carlos E. Palau Salvador
Dr. Fernando Boronat Seguí

A Julia y Sergio, lo que más amo, que han nacido y crecido durante esta aventura.

A Noelia por todo, por tanto y para siempre.

Resumen

El interés en la industria y a nivel académico en el desarrollo en el campo de Internet de las Cosas (IoT) es muy alto. Se han diseñado e implementado una gran cantidad de soluciones a diferentes niveles. Desde soluciones a nivel de dispositivo hasta plataformas IoT completas. No obstante, desarrollar nuevas soluciones IoT en muchos casos puede suponer un esfuerzo complejo. Esta no es una tarea que se deba realizar desde cero. Las plataformas IoT ofrecen las herramientas necesarias para administrar y trabajar con los dispositivos y objetos conectados a ellas. Las plataformas utilizan estos datos para producir resultados y ofrecer servicios y aplicaciones.

El ecosistema IoT abarca una amplia gama de dispositivos, sensores, actuadores, entidades de conocimientos, protocolos, tecnologías, redes, plataformas, servicios, aplicaciones, sistemas y datos muy diversos. Como consecuencia de su naturaleza heterogénea y la ausencia de un estándar global de IoT, un hecho que tampoco se espera lograr en un futuro próximo, en lugar de lograr la integración perfecta entre los diferentes sistemas IoT proliferan diferentes tecnologías y sistemas que implementan sus propios protocolos de interoperabilidad para los objetos que componen IoT.

El trabajo realizado en esta Tesis Doctoral se encarga de revertir esta problemática asociada a la heterogeneidad de las plataformas IoT y la falta de un estándar de interoperabilidad predominante en el mercado. Por tanto, el objetivo de la misma es ofrecer una solución centrada en aprovechar las diferentes ventajas que ofrecen las plataformas, aplicaciones y servicios IoT disponibles, para ofrecer una serie de mecanismos de interoperabilidad y un marco común que permitan poder acceder, interactuar e intercambiar información y funcionalidades entre las diferentes plataformas IoT. Concretamente, la Tesis Doctoral se centra en las necesidades de interoperabilidad de plataformas IoT en las capas de Middleware y Aplicación y Servicios.

Desde la perspectiva de los mecanismos de la capa middleware, la Tesis Doctoral establece soluciones basadas en una capa de abstracción que facilita el acoplamiento de las diferentes plataformas. Esto proporciona funcionalida-

des para acceder a las principales características e información de las diferentes plataformas IoT. Desde la perspectiva de los mecanismos de la capa de aplicación y servicios, se diseñan y definen soluciones para el acceso común y la interacción entre los distintos servicios y aplicaciones heterogéneos ofrecidos por las plataformas.

Además, en la Tesis Doctoral se presentan aquellos elementos transversales para ofrecer una solución de interoperabilidad completa. En primer lugar, se exponen aquellos requisitos necesarios para gestionar la confianza, seguridad, privacidad, virtualización, extensibilidad o escalabilidad. En segundo lugar, se presenta la definición de un marco común de interoperabilidad que proporciona una forma de unificar los diferentes mecanismos de interoperabilidad presentados. También se ofrecen herramientas para gestionar, acceder y hacer un uso adecuado de los mecanismos de interoperabilidad.

Finalmente, se presenta la aproximación a la solución propuesta llevada a cabo en los proyectos europeos H2020: INTER-IoT, ACTIVAGE, PIXEL y DataPorts. Estos proyectos han servido para definir, desarrollar y validar los mecanismos de interoperabilidad y la solución presentada en esta Tesis Doctoral.

Resum

L'interés en la indústria i a nivell acadèmic en el desenvolupament en el camp d'Internet de les Coses (IoT) és molt alt. S'han dissenyat i implementat una gran quantitat de solucions a diferents nivells. Des de solucions a nivell de dispositiu fins a plataformes IoT completes. No obstant això, desenvolupar noves solucions IoT en molts casos pot suposar un esforç complex. Aquesta no és una tasca que s'haja de realitzar des de zero. Les plataformes IoT ofereixen les eines necessàries per a administrar i treballar amb els dispositius i objectes connectats a elles. Les plataformes utilitzen aquestes dades per a produir resultats i oferir serveis i aplicacions.

L'ecosistema IoT es compon d'una una àmplia gamma de dispositius, sensors, actuadors, entitats de coneixements, protocols, tecnologies, xarxes, plataformes, serveis, aplicacions, sistemes i dades molt diverses. A conseqüència de la seua naturalesa heterogènia i l'absència d'un estàndard global de IoT, un fet que tampoc s'espera aconseguir en un futur pròxim, es produeix que en lloc d'aconseguir la integració perfecta entre els diferents sistemes IoT, proliferen diferents tecnologies i sistemes que implementen els seus propis protocols d'interoperabilitat per als objectes que componen Internet de les Coses.

El treball realitzat en aquesta tesi doctoral s'encarrega de revertir aquesta problemàtica associada a l'heterogeneïtat de les plataformes IoT i la falta d'un estàndard d'interoperabilitat predominant en el mercat. Per tant, l'objectiu és oferir una solució centrada en aprofitar els diferents avantatges que ofereixen les plataformes, aplicacions i serveis IoT disponibles, per a oferir una sèrie de mecanismes d'interoperabilitat i un marc comú que permeten poder accedir, interactuar i intercanviar informació i funcionalitats entre les diferents plataformes IoT. Concretament, el treball se centra en les necessitats d'interoperabilitat de plataformes IoT en les capes de Middleware i Aplicació i Serveis.

Des de la perspectiva dels mecanismes de la capa Middleware, el present treball estableix solucions basades en una capa d'abstracció que facilita la unificació de les diferents plataformes. Això proporciona les funcionalitats per a accedir a les principals les característiques i informació de les diferents platafor-

mes IoT. Des de la perspectiva dels mecanismes de la capa d'aplicació i serveis, es dissenya i defineixen solucions per a l'accés comú i la interacció entre els diferents serveis i aplicacions heterogenis oferits per les plataformes.

A més, es presenten en el present treball aquells elements transversals per a oferir una solució d'interoperabilitat completa. En primer lloc, aquells requisits necessaris per a gestionar la confiança, seguretat, privacitat, virtualització, extensibilitat o escalabilitat. En segon lloc, la definició d'un marc comú d'interoperabilitat que proporciona una manera d'unificar els diferents mecanismes d'interoperabilitat presentats. Oferint eines per a gestionar, accedir i fer un ús adequat dels mecanismes d'interoperabilitat.

Finalment, es presenta l'aproximació a la solució proposada duta a terme en els projectes europeus H2020: INTER-IoT, ACTIVAGE, PÍXEL i DataPorts. Aquests projectes han servit per a definir, desenvolupar i validar els mecanismes d'interoperabilitat i la solució oferida en aquesta tesi doctoral.

Abstract

There is a strong interest in the field of the Internet of Things (IoT) in the industry and the academia. A large number of solutions have been designed and implemented at different levels. From device level solutions to complete IoT platforms. However, developing new IoT solutions can be a challenging task. This is not a task that needs to be done from scratch. IoT platforms provide the tools needed to manage and access to the devices and objects connected to them. The platforms can take advantage of this data to produce results and deliver services and applications.

The IoT ecosystem encompasses a wide range of diverse devices, sensors, actuators, knowledge entities, protocols, technologies, networks, platforms, services, applications, systems and data. As a consequence of its heterogeneous nature and the absence of a global IoT standard, something that is also not expected to be achieved soon, instead of achieving seamless integration between different IoT systems, different technologies and systems proliferate and providing their own interoperability protocols for the objects related with Internet of Things.

The work carried out in this PhD thesis aims to address this problem associated with the heterogeneity of IoT platforms and the lack of a predominant interoperability standard in the market. Therefore, the objective is to offer a solution focused on taking advantage of the different benefits offered by the available IoT platforms, applications and services, in order to offer a series of interoperability mechanisms and a common framework that allows accessing, interacting and exchanging information and functionalities between the different IoT platforms. Specifically, the work is focused on the interoperability needs at the Middleware and Application and Services layers of the IoT Platforms.

From the perspective of the Middleware layer mechanisms, this work establishes solutions based on an abstraction layer that facilitates the coupling of the different platforms. This provides functionalities to access to the main features and information of the different IoT platforms. From the perspective of

the Application and Service layer mechanisms, this work designs and defines solutions for common access and interaction between the different heterogeneous services and applications offered by the IoT platforms.

In addition, this PhD tesis presents those cross-cutting aspects needed to provide a complete interoperability solution. Firstly, those requirements involved in to manage trust, security, privacy, virtualisation, extensibility or scalability. Secondly, the definition of a common interoperability framework that provides a way to unify the different interoperability mechanisms presented. It offers tools for managing, accessing and making appropriate use of the interoperability mechanisms developed in this work.

Finally, it describes the approach to the proposed solution carried out in the following H2020 european projects: INTER-IoT, ACTIVAGE, PIXEL and DataPorts. These research projects have been used to define, develop and validate the interoperability mechanisms and the solution offered in this PhD tesis.

Agradecimientos

Este es el final de una etapa, que empezó el día que comencé a trabajar en el grupo de investigación SATRD (Grupo de Sistemas y Aplicaciones de Tiempo Real Distribuidos) de la UPV, y que, en estos momentos, finaliza con la realización de esta Tesis Doctoral. Esto simplemente es el fin de una etapa académica dentro de un largo camino profesional. Mi ilusión es continuar trabajando en este campo, en nuevos proyectos y oportunidades, aprovechando toda la experiencia adquirida hasta ahora.

Se ha cruzado en este recorrido mucha gente. Ellos me han apoyado y animado para conseguir este objetivo, y de todos he aprendido algo. No obstante, también se ha cruzado una pandemia, que ha hecho que los últimos años de trabajo hayan sido más difíciles.

En primer lugar, mi agradecimiento a Carlos Palau, uno de mis directores de la Tesis Doctoral. Son muchas las cosas que le que podría agradecer, principalmente, su dedicación al campo de la investigación en general y a mí trabajo en particular. De todas las cosas que puedo agradecerle, me gustaría destacar la confianza que ha tenido en mí. Personalmente, yo no era consciente de todo lo que podía lograr hasta que comencé a trabajar con él. También deseo agradecer al otro director de la Tesis Doctoral, Fernando Boronat, su gran apoyo y ayuda.

En segundo lugar, quiero agradecer a todos los compañeros que están o han estado en algún momento trabajando conmigo en el SATRD, sin los cuáles, este trabajo no hubiera sido posible. Son innumerables los buenos momentos que hemos pasado juntos durante estos años.

En tercer lugar, también quiero agradecer a mis amigos. Nuestra vida ha cambiado mucho últimamente y ya no nos vemos tanto. Las responsabilidades, la Tesis Doctoral o el trabajo nos han distanciado. Muchas gracias por seguir

ahí, porque cada vez que nos volvemos a encontrar es como si no hubiera pasado el tiempo.

En cuarto lugar, destacar a mi familia. A mis padres, que siempre me han ayudado y se han sacrificado en todo lo necesario para que pudiera alcanzar mis metas académicas. No puedo estar más agradecido a ellos. También a mis tíos y primos porque sé que puedo contar con ellos cuando los necesito. Especialmente, quiero mostrar todo mi amor a mi abuelo que se marchó hace poco, y a mi abuela, a quien por culpa de la pandemia no he podido ver tanto como me gustaría. Finalmente, a los padres de Noelia, su abuela, mi sobrino, mi cuñada y mi cuñado. Ellos son mi familia en mi nuevo pueblo y me han ayudado en todo, y más, para que pueda realizar esta Tesis Doctoral.

Dejo para el final a las personas más especiales de mi vida. Noelia, la persona que cambió mi vida desde el día que la conocí, con la que he pasado los momentos más maravillosos y que ha estado acompañándome en cada difícil momento. Y a ellos, Julia y Sergio, no tengo palabras para definir el amor que siento por ellos. Julia apenas tenía unos meses cuando empecé y Sergio aún no había nacido. Ahora los veo y no puedo creer cómo ha pasado el tiempo. Son demasiado pequeños para entender qué es una Tesis Doctoral, y sólo lo ven como algo que les aleja cada día un poco de su padre. A ellos les prometo que les dedicaré todos los momentos que tenga, porque son lo más importante de mi vida.

Mi más sincero y afectuoso agradecimiento a todos vosotros.

Andreu Belsa Pellicer
Valencia, Enero de 2022

Índice

Índice de figuras

ÍNDICE DE FIGURAS

Índice de tablas

ÍNDICE DE TABLAS

Acrónimos

AAA	Autenticación, Autorización y Contabilidad
ACTIVAGE	ACTivating InnoVative IoT smart living environments for AGEing well
AHA	Envejecimiento Activo y Saludable
API	Interfaces de Programación de Aplicaciones
BDAaaS	Big Data Analytics as a Service
BDVA	Big Data Value Association
CEF	Connecting Europe Facility
CEP	Procesador de Eventos Complejos
CLI	Interfaz de Línea de Comandos
CSA	Coordination and Support Actions
CSS	Hojas de Estilo en Cascada
DataPorts	A Data Platform for the Cognitive Ports of the Future
DAV	Abstracción y Virtualización de Datos
DS	Entornos de Despliegue
EC	Comisión Europea
GUI	Interfaz Gráfica de Usuario
H2020	Horizonte 2020
HLA	Arquitectura de Referencia de Nivel Alto
HTML	Lenguaje de Marcas de Hipertexto
IDS	Espacios de Datos Internacionales
IA	Inteligencia Artificial
IDSA	Asociación de Espacios de Datos Internacionales

INTER-IoT	Interoperability of Heterogeneous IoT Platforms
INTER-METH	INTER-IoT Engineering Methodology
IoT	Internet de las Cosas
IoT-EPI	Iniciativa de Plataformas Europeas de Internet de las Cosas
IoT-LSP	Programa Europeo de Pilotos a Gran Escala
ITU	International Telecommunication Union
JS	JavaScript
JSON-LD	JavaScript Object Notation for Linked Data
KPI	Indicador clave de rendimiento
LCIM	Levels of Conceptual Interoperability Model
M2M	Machine to Machine
MoM	Middleware orientado a Mensajes
NG-IoT	Próxima Generación de Internet de las Cosas
NGSI	Next Generation Service Interfaces
ORB	Object Request Broker
OWL	Lenguaje de Ontología Web
PaaS	Platform-as-a-Service
PIXEL	Port IoT for Environmental Leverage
PPP	Participación Público-Privada
RDF	Marco de Descripción de Recursos
REST	Transferencia de Estado Representacional
RIA	Research and Innovation Action
RPC	Llamada a Procedimiento Remoto
SDOs	Standard Development Organisations
SDN	Redes Definidas por Software
SDK	Software Development Kit
SEAMS	Smart Energy-efficient and Adaptive Management System
SoM	Middleware orientado a Servicios
TIC	Tecnologías de la Información y la Comunicación
UI	Interfaz de Usuario
URL	Localizador de Recursos Uniforme
WYSIWYG	Lo que Ves Es lo que Obtienes

Capítulo 1

Introducción

1.1. Introducción

Internet de las Cosas (IoT) abarca una amplia variedad de elementos interconectados entre sí. Se trata, entre otros, de: dispositivos, sensores, actuadores, protocolos, redes físicas y virtuales, plataformas middleware, servicios, aplicaciones, sistemas, ontologías, datos de diversa procedencia, tecnologías... Como consecuencia de esta naturaleza heterogénea, en lugar de lograrse una integración perfecta entre los diferentes elementos que componen el ecosistema IoT, están apareciendo diferentes iniciativas que implementan sus propios protocolos de interoperabilidad para los objetos que componen IoT. Esta ausencia de un estándar global en las plataformas IoT, es la principal motivación de este trabajo. Por tanto, la interoperabilidad es uno de aspectos fundamentales y uno de los desafíos más difíciles de enfrentar en el campo de IoT. El interés, tanto en la industria como a nivel académico, en el desarrollo de mecanismos para garantizar la interoperabilidad en el campo de IoT es muy alto.

El mercado relacionado con IoT ha empezado a crecer exponencialmente. Las empresas se están dando cuenta de que construir sistemas de IoT desde cero es una tarea tediosa y un gran riesgo, ya que muchos casos de negocio siguen siendo inestables y no están claramente validados. Frente a una continua reinvención de la rueda, el mercado ha respondido generando una oleada de empresas que ofrecen soluciones listas para usar. Estas soluciones encapsulan las principales partes de un sistema de IoT en bloques funcionales que pueden utilizarse en diferentes dominios de aplicación del mercado. Estas son las conocidas como plataformas de IoT.

Las plataformas de IoT permiten a las empresas introducir soluciones rápidamente en el mercado, reduciendo el tiempo de desarrollo y los gastos. Como

resultado, las plataformas de IoT han crecido significativamente en popularidad. Sin embargo, este creciente impacto ha llevado a una situación de un exagerado número de plataformas en el mercado. Por tanto, surge la necesidad de desarrollar nuevas soluciones enfocadas a superar la creciente fragmentación del mercado. Concretamente, el objetivo es proporcionar soluciones encargadas de gestionar la interoperabilidad de las plataformas de IoT existentes en el mercado. El resultado de estas soluciones es destacar el beneficio de la interoperabilidad resultante en los diferentes dominios del mundo real y en los ecosistemas de innovación tecnológica [1]. La heterogeneidad de las plataformas es otro aspecto a señalar. Existen una gran cantidad de soluciones IoT disponibles, las cuales cubren, desde pasarelas para la interconexión de dispositivos, hasta completas y complejas plataformas IoT, que ofrecen aplicaciones con un alto valor añadido para los datos disponibles en ellas.

Con la intención de clarificar y centrar el objetivo del presente trabajo, se toma como referencia una aproximación a la arquitectura de Internet de las Cosas que se divide en las siguientes capas: dispositivo, red, middleware, aplicación/servicios y semántica. Concretamente, este trabajo se centra en plataformas que se encuentran ubicadas en las capas de Middleware y Aplicación/Servicios. Desde el punto de vista técnico, estas capas de las plataformas ofrecen las herramientas necesarias para facilitar la administración de los dispositivos y objetos conectados a ellas. De este modo, dichas plataformas pueden ofrecer aplicaciones y servicios sobre estos dispositivos y objetos. El objetivo general de este trabajo será estudiar estas plataformas y diseñar una solución que facilite mecanismos para lograr la interoperabilidad entre ellas.

Entrando en detalle en dichas capas, en primer lugar, la infraestructura middleware de las plataformas IoT, es aquella que soporta la configuración flexible y el despliegue de algoritmos para la recopilación y el filtrado de flujos de información derivados de los objetos conectados a Internet, al tiempo que generan y procesan dicha información. Son muchas las soluciones Middleware que han surgido en los últimos años y no existen actualmente soluciones que faciliten la interoperabilidad entre ellas. Es por este motivo, que el objeto de este trabajo en la capa de Middleware, es lograr la interoperabilidad mediante el establecimiento de una capa de abstracción y el acoplamiento de las diferentes plataformas IoT a dicha capa de abstracción, facilitando que los diferentes módulos incluidos en esta solución proporcionen los servicios para administrar los objetos de las diferentes plataformas IoT, y así poder acceder a sus características e información. La interoperabilidad que ofrecerá esta capa de abstracción permitirá una explotación global de objetos inteligentes en sistemas IoT multiplataforma.

En segundo lugar, integrados en la mayoría de las plataformas IoT, se encuentran las aplicaciones y servicios. Estos utilizan los datos extraídos de los

sensores y dispositivos con diferentes objetivos. Cada Plataforma tiene varias aplicaciones y servicios IoT. disponibles, son completamente heterogéneos y, además, en muchas ocasiones, se presentan extremadamente relacionados con sus dominios de uso. Entonces, ¿qué sucede cuando dos servicios diferentes de la misma plataforma IoT o de diferentes plataformas IoT quieren intercambiar información de datos entre ellos? ¿Cómo se puede hacer posible que la información de un servicio alimente la entrada de otro servicio? La heterogeneidad dificulta la interoperación y comunicación de los servicios, pero existen una serie de soluciones en forma de protocolos o paradigmas que han surgido para resolver el problema de la interacción, y que pueden ser aplicados en el contexto de IoT. Por tanto, el objetivo de esta investigación en la capa de Servicio/Aplicación será diseñar y definir una solución para la interacción entre los servicios heterogéneos de las diferentes plataformas IoT.

Para lograr estos objetivos relacionados con la interoperabilidad, ha sido necesario analizar la percepción del mercado sobre las plataformas IoT, analizar aquellas plataformas destacadas, estudiar sus características y puntos clave, los protocolos con los que será necesario interactuar y cómo se crean ecosistemas alrededor de ellas. Finalmente, es importante destacar que el objetivo de esta Tesis no es la creación de una nueva solución que sustituya a las anteriores. Esto significaría caer en un error enunciado en párrafos anteriores como es el de tratar de reinventar la rueda. En esta Tesis se pretende lo opuesto que, de manera resumida, consiste en ofrecer una solución que aproveche las funcionalidades disponibles en las plataformas existentes y, además, proporcione el valor añadido de intercambiar, reusar e integrar los datos, servicios y aplicaciones que estas ofrecen.

1.2. Motivación

Este Tesis toma en consideración el escenario actual del ecosistema IoT, descrito en la introducción, en el que la falta de interoperabilidad acarrea importantes problemas, tanto tecnológicos como comerciales. Se pueden observar estos problemas derivados en diferentes situaciones, como: la imposibilidad de conectar dispositivos no interoperables en las plataformas IoT, la dificultad que implica el desarrollo de aplicaciones IoT basadas en la información de diferentes plataformas, el desánimo en la adopción de las tecnologías IoT, el incremento de costes provocados por la falta de comunicación entre plataformas heterogéneas, las dificultades que suponen la adaptación a diferentes dominios de aplicación o las carencias en reusabilidad de las soluciones existentes con otras para ofrecer valor añadido a las organizaciones.

Mediante el enfoque propuesto, la heterogeneidad de las plataformas IoT pasará de ser un problema crucial a una gran oportunidad. Por tanto, son varios los puntos que motivan la elaboración de la presente Tesis. En primer lugar, mediante esta solución, se evita la incertidumbre y espera de la llegada de un estándar único que garantice las capacidades de un IoT interoperable. En segundo lugar, se pretende superar la fragmentación de sistemas cerrados, arquitecturas y áreas de aplicación para avanzar hacia sistemas abiertos y plataformas que admitan múltiples aplicaciones. En tercer lugar, se pretende capturar los beneficios de desarrollar plataformas orientadas al consumidor mediante una fuerte cooperación entre los agentes implicados y permitir a los desarrolladores producir aplicaciones con un nuevo valor agregado en múltiples sistemas. En cuarto lugar, se busca ofrecer una solución a futuro que permita que las nuevas generaciones de sistemas integrados; protocolos de comunicación; middleware; servicios y aplicaciones se incluyan en la solución desarrollada cuando estén disponibles en el futuro. En quinto lugar, se basa en pruebas de concepto, demostraciones a gran escala y validación impulsada por escenarios de uso innovadores, que estén aprovechando plataformas desplegadas y funcionando en las organizaciones. Finalmente, se incluyen mecanismos eficaces y eficientes, garantizando la gestión de las plataformas heterogéneas, ofreciendo un marco de desarrollo de nuevas soluciones y garantizando la seguridad y privacidad de la información conectada y disponible.

1.3. Objetivos

Partiendo de la problemática y la motivación descrita en los apartados anteriores, el presente trabajo pretende cubrir un objetivo de interoperabilidad general basado en un marco común para la interoperabilidad de plataformas IoT en las capas de Middleware y Aplicación y servicios. Este objetivo, a su vez, se compone de una serie de objetivos más precisos y específicos que abordan la consecución de dicho objetivo general.

1.3.1. Objetivo general

Las plataformas IoT disponibles en la actualidad ofrecen las herramientas necesarias para proporcionar un despliegue efectivo y eficiente para la administración de los dispositivos, sensores, objetos, servicios o aplicaciones que componen Internet de las Cosas. De este modo con el uso de estas plataformas se puede obtener diferentes resultados y tener disponibles una considerable variedad de servicios y aplicaciones. El principal problema ha sido el crecimiento en cantidad y sin ningún estándar de este tipo de plataformas IoT. Esta natu-

raleza heterogénea y la ausencia de un estándar global de IoT, una plataforma de referencia o un grupo reducido de plataformas que sean dominadoras en el mercado supone una barrera. Este asunto que tampoco se prevé que tienda a homogeneizarse en un futuro próximo. En lugar de una tendencia a la integración perfecta entre los sistemas IoT, proliferan diferentes tecnologías y sistemas que implementan sus propios protocolos de interoperabilidad para los objetos que componen Internet de las Cosas.

La interconexión y la interoperabilidad entre las distintas plataformas son aspectos críticos en IoT, así como uno de los desafíos más difíciles de enfrentar en los entornos de IoT. Por tanto, el objetivo general de este trabajo es lograr la interoperabilidad de las distintas plataformas en la capa de middleware y la capa de aplicación y servicios:

- El objetivo, centrado en la interoperabilidad, en la capa Middleware es establecer una capa de abstracción que facilite el acoplamiento de las diferentes plataformas IoT. Esto proporciona funcionalidades para acceder a principales las características e información de las diferentes plataformas.
- El objetivo, centrado en la interoperabilidad, en la capa de Aplicación/Servicio es diseñar y definir una solución para el acceso común y la interacción entre los distintos servicios y aplicaciones heterogéneos ofrecidos por las plataformas IoT.

1.3.2. Objetivos específicos

Los objetivos específicos de la presente Tesis son los siguientes:

- *O1.* Definición e implementación de una capa de abstracción capaz de acoplar los componentes de Middleware y Aplicación/Servicios de las plataformas heterogéneas de IoT
- *O2.* Identificar las diferentes plataformas, fuentes de datos y servicios que se integrarán en la solución desarrollada. Análisis de los gestores de colas y brokers disponibles en el mercado. Análisis de las principales funcionalidades a incorporar.
- *O3.* Diseño e implementación de mediadores, puentes o agentes con las plataformas IoT.
- *O4.* Ofrecer herramientas de gestión y comunicación con las plataformas, dispositivos y/o servicios mediante los mecanismos de abstracción implementados.

- *O5.* Ofrecer un registro y mecanismos de descubrimiento de dispositivos, servicios y aplicaciones.
- *O6.* Satisfacer requisitos de tiempo real y de gestión de suscripciones en el acceso y gestión de datos.
- *O7.* Ofrecer herramientas para el procesamiento en tiempo real y la visualización del contexto actual e histórico de los datos.
- *O8.* Analizar y definir soluciones para requisitos básicos no funcionales correlacionados con la interoperabilidad e integración de capas heterogéneas de fiabilidad, seguridad, privacidad y confianza.
- *O9.* Desarrollar funcionalidades y herramientas dentro de un marco de desarrollo interoperable y seguro en IoT, para proporcionar APIs y soluciones operativas para programar y administrar plataformas IoT interoperables.
- *O10.* Reutilizar e intercambiar servicios heterogéneos entre diferentes plataformas de IoT. Permitir que los desarrolladores de aplicaciones produzcan un nuevo valor agregado en múltiples sistemas, y que diferentes organizaciones puedan desarrollar nuevas aplicaciones basadas en la integración de sus infraestructuras heterogéneas de IoT existentes.
- *O11.* Ofrecer la posibilidad de que un servicio utilizado en una plataforma de IoT se pueda reutilizar y combinar con diferentes servicios de otras plataformas de IoT.
- *O12.* Virtualizar los mecanismos habilitadores de la interoperabilidad desarrollados para facilitar las tareas de acceso, despliegue y gestión de los diferentes componentes.

1.4. Metodología de investigación

La metodología de investigación utilizada para lograr los objetivos de la sección anterior se compone de las siguientes fases:

- Analizar la literatura relacionada con los problemas de interoperabilidad a nivel de Middleware y Aplicación y Servicios.
- Identificar, desplegar, configurar y realizar pruebas con las distintas plataformas IoT y sus correspondientes Aplicaciones/Servicios.

- Definir y diseñar una solución genérica, basada en mecanismos de interoperabilidad, componentes habilitantes y marco común de interoperabilidad, para solucionar los problemas de interoperabilidad en las capas de Middleware y Aplicación/Servicios de las plataformas IoT.
- Ofrecer una implementación de la solución diseñada, basada en una combinación de componentes nuevos implementados para tal fin y componentes existentes con fuerte arraigo en el mercado.
- Implementar un prototipo que implemente los módulos necesarios para lograr la solución de interoperabilidad diseñada.
- Seleccionar casos de uso donde validar el prototipo en sus diferentes escenarios de validación.
- Validar el prototipo, y, por tanto, los distintos mecanismos de interoperabilidad propuestos en dichos escenarios.
- Analizar los resultados obtenidos con el fin de extraer conclusiones relevantes sobre la investigación realizada.
- Ofrecer una serie de lecciones aprendidas durante el desarrollo de la investigación.
- Establecer nuevas líneas de investigación derivadas o que tomen como base los mecanismos de interoperabilidad ofrecidos.

1.5. Principales aportaciones

Se ofrece un listado de los artículos publicados en revistas y congresos, los capítulos en libro y la participación en proyectos de investigación realizados en el marco de la investigación realizada durante la elaboración de esta Tesis Doctoral.

1.5.1. Artículos

- Lacalle Úbeda, Ignacio; Belsa, Andreu; Vaño, Rafael; Palau, Carlos. (2020). Framework and Methodology for Establishing Port-City Policies Based on Real-Time Composite Indicators and IoT: A Practical Use-Case. Sensors. 20. 4131. 10.3390/s20154131.

- Valero, Clara; Medrano Gil, Alejandro; Gonzalez-Usach, Regel; Julián, Matilde; Fico, Giuseppe; Arredondo, Maria; Stavropoulos, Thanos; Strantsalis, Dimitrios; Voulgaridis, Antonis; Roca, Felipe; Jara, Antonio; Serrano, Martín; Zappa, Achille; Khan, Yasar; Guillen, Sergio; Sala, Pilar; Belsa, Andreu; Votis, Konstantinos; Palau, Carlos. (2021). AIoTES: Setting the principles for a Semantic Interoperable and modern IoT-enabled Reference Architecture for Active and Healthy Ageing ecosystems. Computer Communications. 177. 10.1016/j.comcom.2021.06.010.

1.5.2. Congresos

- Belsa-Pellicer, Andreu; Sarabia-Jácome, David Fernando; Palau Salvador, Carlos Enrique; Esteve Domingo, Manuel (2018). Flow-based programming interoperability solution for IoT Platform Applications. En IEEE International Conference on Cloud Engineering (IC2E 2018). (304 - 309). Orlando, USA: IEEE

- Sarabia-Jácome, David; Belsa Pellicer, Andreu; Palau, Carlos E.; Esteve, Manuel (2018). Exploiting IoT Data and Smart City Services for Chronic Obstructive Pulmonary Diseases Risk Factors Monitoring. En IEEE International Conference on Cloud Engineering (IC2E 2018). (351 - 356). Orlando, USA: IEEE

- Yacchirema-Vargas, Diana Cecilia; Belsa-Pellicer, Andreu; Palau Salvador, Carlos Enrique; Esteve Domingo, Manuel (2018). OneM2M Based-Interworking Architecture for Heterogeneous Devices Interoperability in IoT. En IEEE Conference on Standards for Communications and Networking (CSCN 2018) . Paris, France: IEEE.

- Suárez de Puga-García, Jara; Palau Salvador, Carlos Enrique; Belsa-Pellicer, Andreu (2019). Architecture and Use Case for an IoT deployment with SDN at the Edge and dual Physical and Virtual Gateway. EN 28th International Conference on Computer Communications and Networks (ICCCN 2019). (1 - 6). Valencia, Spain: IEEE.

- Suárez de Puga-García, Jara; Palau Salvador, Carlos Enrique; Belsa-Pellicer, Andreu (2019). Architecture proposal for SD-IoT deployments with a decoupling of gateway functionalities. En 28th European Conference on Networks and Communications (EuCNC 2019). (1 - 1). Valencia, Spain: IEEE.

- Lacalle-Úbeda, Ignacio; López-Pinar, César; Belsa-Pellicer, Andreu; Z. Kopertowski; Palau Salvador, Carlos Enrique; Esteve Domingo, Manuel

(2021). Reviewing SDN adoption strategies for Next Generation Internet of Things networks. En 3rd International Conference on Smart Systems, Innovations and Computing (SSIC 2021). Online: Springer.

Además está aceptado y pendiente de ser publicado este trabajo:

- Andreu Belsa Pellicer, Rafael Vaño Garcia, Ignacio Lacalle Ubeda, Matilde J. Seguí, Fernando Boronat Segui and Carlos E. Palau Salvador (2021). A novel approach for calculating real-time Composite Indicators relying on Internet of Things and Industrial Data Spaces. En The 14th International Symposium on Intelligent Distributed Computing (IDC 2021), Sep 16-18, 2021, Online Conference, Italy

1.5.3. Capítulos de libro

- Belsa, Andreu; Fornes-Leal, Alejandro; Valero, Clara; Olivares, Eneko; Puga, Jara; Boronat, Fernando; Fuart, Flavio. (2021). INTER-Layer: A Layered Approach for IoT Platform Interoperability. 10.1007/978-3-030-82446-4_4.

- Valero, Clara; Belsa, Andreu; Fornes-Leal, Alejandro; Boronat, Fernando; Llorente, Miguel; Montesinos, Miguel. (2021). INTER-Framework: An Interoperability Framework to Support IoT Platform Interoperability. 10.1007/978-3-030-82446-4_6.

- López, César; Lacalle Úbeda, Ignacio; Belsa, Andreu; Kopertowski, Zbigniew; E.Palau, Carlos; Esteve, Manuel. (2022). Reviewing SDN Adoption Strategies for Next Generation Internet of Things Networks. 10.1007/978-981-16-2877-1_57.

Además están aceptados y pendientes de ser publicado estos trabajos:

- Santiago Cáceres, Francisco Valverde, Carlos E. Palau, Andreu Belsa Pellicer, Christos A. Gizelis, Dimosthenes Krassas, Hanane Becha, Réda Khouani, Andreas Metzger, Nikos Tzagkarakis, Anthousa Karkoglou, Anastasios Nikolakopoulos, Achilleas Marinakis, Vrettos Moulos, Antonios Litke, Amir Shayan Ahmadian, Jan Jürjens. (2021) Towards cognitive ports of the future.

  El capítulo será incluido en el libro: Technologies and Applications for Big Data Value.

- Claudio Savaglio, Clara I. Valero, Andreu Belsa, Carlos E. Palau, Giancarlo Fortino. (2021) Cloudification of IoT platform.

El capítulo será incluido en el libro: Springer Handbook of Internet of Things, correspondiente a la sección relacionada con Arquitectura e Interoperabilidad.

1.5.4. Participación en proyectos de investigación

El capítulo 5 proporciona la visión general y detallada de todos los proyectos de investigación que han contribuido a la elaboración de la presente Tesis Doctoral. La implicación del autor, el proceso de diseño, el desarrollo software (mediante código abierto disponible en los respectivos repositorios de los proyectos de investigación y con licencias permisivas de software libre), las plataformas de documentación y los resultados derivados de dichos proyectos europeos de investigación están descritos en los capítulos 6, 7 y 8.

Se procede en esta sección a ofrecer el listado de todos los proyectos de investigación en los que se ha participado durante el periodo de elaboración de la Tesis Doctoral:

- Sistema de Monitorización y Control Seguro de Residencias de la Tercera Edad (SAFE-ECH). 2016-2017. I+D Colaborativa competitiva. Ministerio de Economía y Empresa.
- Interoperability Of Heterogeneous Iot Platforms (INTER-IoT). 2016-2019. I+D Colaborativa competitiva. Comisión de las Comunidades Europea.
- Activating Innovative Iot Smart Living Environments For Ageing Well (ACTIVAGE). 2017-2020. I+D Colaborativa competitiva. Comisión de las Comunidades Europea.
- Port Iot For Environmental Leverage (PIXEL). 2018-2021. I+D Colaborativa competitiva. Comisión de las Comunidades Europea.
- A Data Platform For The Cognitive Ports Of The Future (DataPorts). 2020-Actualidad. I+D Colaborativa competitiva. Comisión de las Comunidades Europea.

1.6. Estructura de la memoria

La memoria del presente trabajo está estructurada de la siguiente manera:

- El primer capítulo introduce el contenido, la motivación que ha conducido a la realización de la Tesis, los objetivos generales y específicos que se

pretenden cubrir en la misma, además de un listado de las fases seguidas en la metodología de investigación utilizada, las principales aportaciones científicas realizadas por el autor durante su desarrollo y la estructura del presente documento.

- El segundo capítulo resume el estado del arte que ha fundamentado el trabajo realizado en la presente tesis doctoral. Está compuesto por la visión general y estado actual de las soluciones IoT, la problemática relacionada con la definición, propósito y retos de la interoperabilidad en IoT. Se proporciona una aproximación a las arquitecturas, estándares, iniciativas, proyectos, herramientas, soluciones y plataformas centradas en la interoperabilidad. También se analiza las plataformas IoT más destacadas del ecosistema actual. Finalmente, se detallan tecnologías habilitadoras de la interoperabilidad y los marcos de soporte de este tipo de soluciones.

- El tercer capítulo, describe el diseño de los mecanismos de interoperabilidad a nivel de Middleware y Aplicaciones y Servicios. Se parte de los fundamentos seguidos para llevar a cabo dicho diseño, la visión general de la arquitectura propuesta, los dominios de aplicación que se pretenden cubrir y los requisitos involucrados. Finalmente, se describen con mayor detalle los componentes de la arquitectura, los componentes habilitantes de la solución de interoperabilidad y el marco común de interoperabilidad.

- El cuarto capítulo, detalla el proceso de implementación de la arquitectura presentada en el tercer capítulo Se parte de una visión general de las tecnologías seleccionadas para, a continuación, ofrecer detalles específicos de cada uno de los mecanismos y elementos habilitadores implementados.

- El quinto capítulo, expone una visión general, motivación, cronología, entorno práctico, soluciones técnicas desplegadas y lecciones aprendidas de los casos de uso mediante los cuales se ha llevado a cabo la validación y despliegue de la especificación y desarrollo de mecanismos de interoperabilidad, presentadas en los capítulos 4 y 5.

- El sexto capítulo ofrece la visión general del caso de uso Interoperability of Heterogeneous IoT Platforms (INTER-IoT). Corresponde con un proyecto de investigación, en el marco de la iniciativa de investigación financiada por la Unión Europea Horizonte 2020 (H2020), mediante el cual se ha podido validar la solución propuesta en la presente Tesis Doctoral. Se ofrecen los resultados obtenidos y el alcance de la solución propuesta dentro del proyecto. Además, se describen con detalle los componentes de la arquitectura y piezas de implementación de las Tesis involucradas en dicho proyecto.

- El séptimo capítulo presenta la visión general del caso de uso ACTivating InnoVative IoT smart living environments for AGEing well (ACTIVAGE). Corresponde con un segundo proyecto de investigación, en el marco de la iniciativa H2020, mediante el cual también se ha podido validar la solución propuesta en la presente Tesis Doctoral. Se ofrecen los resultados obtenidos y el alcance de la solución propuesta dentro del proyecto. Además, se describen con detalle los componentes de la arquitectura y piezas de implementación de la Tesis involucradas en dicho proyecto.

- El octavo capítulo relata la visión general del caso de uso A Data Platform for the Cognitive Ports of the Future (DataPorts). Corresponde con un tercer proyecto de investigación, en el marco de la iniciativa H2020, mediante el cual se ha podido validar la solución propuesta en la presente Tesis Doctoral. Se ofrecen los resultados obtenidos y el alcance de la solución propuesta dentro del proyecto. Además, se describen con detalle los componentes de la arquitectura y piezas implementación de las tesis involucradas en dicho proyecto. También se ofrece una visión general del caso de uso Port IoT for Environmental Leverage (PIXEL) otro proyecto H2020 estrechamente relacionado con DataPorts a nivel de diseño y con implicaciones en las decisiones que se han tomado en la solución propuesta.

- El noveno capítulo ofrece un resumen de las ideas principales ofrecidas en el presente trabajo y las conclusiones obtenidas. Esto va acompañado de las lecciones aprendidas durante la realización de la tesis y un análisis de las líneas de investigación futuras derivadas del presente trabajo.

Capítulo 2

Estado del Arte

El contenido de este capítulo proporciona las bases de los posteriores capítulos de este trabajo. Se centra en análisis previo realizado del estado del arte que ha llevado a las decisiones tomadas en el diseño y especificación de la solución propuesta. Por tanto, las líneas generales que pretende cubrir este capítulo son un análisis previo del estado de los mecanismos, componentes habilitantes y marcos de desarrollo disponibles para garantizar la interoperabilidad entre sistemas informáticos. Las soluciones existentes para sistemas informáticos son una base fundamental para encontrar mecanismos de interoperabilidad efectivos y eficientes en el campo de IoT. El análisis destacará las soluciones existentes. Por un lado, aquellas que consisten en soluciones que han sido adaptadas de sistemas previos. Por otro lado, aquellas que consisten en soluciones nuevas desarrolladas siguiendo un enfoque innovador. Se han analizado, de estos dos tipos de soluciones, aquellas que se han centrado en los principales aspectos de la interoperabilidad en el campo de los sistemas informáticos de IoT.

2.1. Introducción

El primer lugar, esta sección ofrece una visión general de las tecnologías IoT involucradas en la presente Tesis Doctoral. Las plataformas IoT son el elemento central de este estudio. Explicado de manera breve, las plataformas proporcionan acceso y permiten gestionar los dispositivos, sensores o entidades conectadas a ellas. Además, ofrecen aplicaciones y servicios que hacen uso de todos los elementos conectados a ellas para ofrecer un valor añadido, como por ejemplo, procesando estos datos, almacenándolos o mediante visualizaciones.

En segundo lugar, se enfatiza en la necesidad de soluciones que permitan la interoperabilidad entre dichas plataformas y sus correspondientes servicios. Como se ha indicado en la introducción, el auge de las plataformas, ha llevado a un crecimiento sin control y sin un estándar predominante de estas soluciones. Por tanto, es necesario analizar las necesidades actuales de la interoperabilidad en IoT. También se destacan los estándares e iniciativas centrados en esta meta.

En tercer lugar, el campo de la interoperabilidad en IoT es muy amplio. Hay diferentes arquitecturas de referencia para las soluciones IoT y múltiples enfoques para abordar la interoperabilidad. La Tesis Doctoral se ubica en la interoperabilidad de la capa middleware y aplicación y servicio de las plataformas. Las soluciones existentes centradas en ambas capas serán, por tanto, el objeto de estudio de este estado del arte.

En cuarto lugar, cabe destacar los proyectos desarrollados en el marco de las diferentes iniciativas de investigación. Los trabajos y análisis ofrecidos por dichos trabajos ha contribuido a alimentar este análisis previo de la situación. Además, es importante resaltar que los proyectos de investigación son el tema de principal relevancia del Capítulo 5 de la presente Tesis, puesto que es donde se han utilizado los mecanismos de interoperabilidad especificados. Todos los proyectos de investigación han realizado un completo análisis previo de la situación y han contribuido con iniciativas enfocadas a lograr la interoperabilidad. Tener en cuenta las conclusiones y resultados obtenidos en dichos trabajos es una información clave.

Finalmente, se resume el estudio previo de las tecnologías habilitadoras y marcos de soporte de la interoperabilidad puesto que son clave para ofrecer al usuario una aplicación atractiva y accesible. Estas soluciones son necesarias en la mayoría de sistemas informáticos y, por lo tanto, son también necesarias en el campo de IoT. El análisis se ha realizado con la finalidad de analizar como integrar soluciones existentes, o también, como desarrollar nuevas soluciones basadas en tecnologías existentes, que faciliten el uso y despliegue seguro de los mecanismos de interoperabilidad.

2.2. Visión general del ecosistema IoT alrededor de la solución propuesta

Hay un extenso numero de definiciones y enfoques al concepto general de IoT [2][3][4]. En dichos trabajos se analiza la literatura, visión, objetivos y una definición común del término. Una interesante visión general del ecosistema IoT consiste en que es una tecnología centrada en transformar los objetos de la realidad en objetos virtuales inteligentes [5]. Partiendo de este tipo de elementos la meta es unificarlos en una infraestructura común que permita, entre otras

cosas, su mantenimiento y gestión. Las soluciones IoT parten del ámbito de las Tecnologías de la Información y la Comunicación (TIC), convirtiéndose en un elemento innovador, en auge y con múltiples posibilidades. Esto se puede ver en la composición de su nombre, que está formado por en primer lugar por el concepto Internet y, en segundo lugar, por el concepto Cosas. La interoperabilidad forma parte de esta necesidad de Internet puesto que es la tecnología que ofrece una interconexión global de ordenadores, servidores, routers, móviles, etc. El concepto Cosas se encarga de llevar más allá los elementos que pueden ser interconectados entre sí. Están formados por elementos que originalmente no eran dispositivos electrónicos conectados a Internet. Los elementos son muy variados como sensores, actuadores, pulseras, relojes, coches, teléfonos, electrodomésticos, etc. [6] El auge de este tipo de dispositivos ha sido imparable, desde que comenzó la presente Tesis a la actualidad algunos han pasado a formar parte de la vida cotidiana de las personas. Por ejemplo, las luces, termostatos o pulseras son elementos que están en gran parte de los hogares, trabajos, ciudades, etc. Por tanto, las denominadas Cosas han pasado de ser elementos que se usen en proyectos industriales o científicos a una realidad a todos los niveles. Esto es lo que da lugar a que las soluciones IoT abarquen múltiples dominios en la actualidad, como ciudades inteligentes, agricultura, energía, calidad medioambiental, salud, etc. Esta es, simplemente, una pequeña muestra de los dominios en los que se pueden aplicar estas tecnologías [7][8][9].

En la actualidad, se está produciendo un importante crecimiento de los datos producidos por los sensores basados en el IoT. El crecimiento del volumen de datos no estructurados, enviados por los dispositivos IoT, supera al de los datos estructurados. Por ello, muchas aplicaciones existentes no se benefician de las oportunidades y la flexibilidad que ofrece la existencia de múltiples fuentes de datos. A medida que los datos crecen en tamaño y heterogeneidad, los problemas de interoperabilidad se convierten en una preocupación creciente [10].

La investigación pone de manifiesto un importante desarrollo de soluciones para una amplia gama de dispositivos y plataformas de IoT en los últimos años. Sin embargo, cada solución ofrece su propia infraestructura, dispositivos, API y formatos de datos, lo que provoca problemas de interoperabilidad. Para aprovechar todo el potencial de IoT, los objetos no tienen que estar simplemente conectados a Internet, sino que también deben ser encontrados, accesibles, gestionados y conectados potencialmente con otros objetos. Para permitir esta interacción, es necesario un grado de interoperabilidad que va más allá del simple protocolo de interoperabilidad que ofrece Internet[11]. Los problemas de interoperabilidad son consecuencia de muchos problemas críticos, como el bloqueo de los proveedores, la imposibilidad de desarrollar aplicaciones de IoT que expongan a través de plataformas y dominios, la dificultad de conectar dis-

positivos de IoT no interoperables en diferentes plataformas de IoT, e impide la aparición de la tecnología de IoT a gran escala. Para hacer frente a estos problemas, han surgido esfuerzos por parte de varios organismos académicos, industriales y de estandarización para ayudar a la interoperabilidad de IoT.

Como se ha indicado en la introducción de este capitulo, en este apartado se presentan los antecedentes y estado actual en el que se ha basado la solución de interoperabilidad de las plataformas IoT a nivel de Middleware y Aplicación/Servicios. El foco principal es detallar las principales características de las plataformas Middleware, enumerar las plataformas IoT principales, los retos que se afrontan desde el punto de vista de la interoperabilidad, explicar la problemática actual con sus aplicaciones/servicios y analizar las soluciones existentes que se han enfocado en ofrecer mecanismos de interoperabilidad a este nivel, tanto en el campo concreto de IoT, como en el campo de las tecnologías de la información en general.

En primer lugar, se ha realizado un estudio de la documentación relacionada con Middleware e Internet de las cosas. Se parte de la definición de Middleware, que indica que se trata de la infraestructura de software y hardware que habilita la capa de comunicación entre los diferentes componentes de un sistema, generalmente en forma de solicitud y respuesta o una comunicación de conexión sostenida para transmisión de datos [12][13][14][15]. El middleware también es una capa conveniente para colocar los servicios adicionales en todo el sistema, como seguridad, auditoría y monitorización [16][17][18].

En el dominio específico de IoT, una plataforma de software definida como middleware tiene el objetivo crítico de conectar el ecosistema heterogéneo de aplicaciones que se comunican a través de interfaces heterogéneas utilizando y operando en diversos tipos de tecnologías [19]. Debido a la dificultad intrínseca de definir y hacer cumplir un estándar común entre todos estos escenarios, los middlewares IoT tienen que proporcionar una capa de abstracción y adaptación a las aplicaciones de Internet de las cosas, además de ofrecer múltiples servicios mediante APIs fáciles de usar y potentes.

El desarrollo de middleware en el dominio de IoT es un área activa de investigación científica e industrial, y hasta el momento se han desarrollado una serie de soluciones interesantes, aunque se está experimentando una abundancia de soluciones de middleware que trabajan con sus propios protocolos y estándares y no bajo un estándar común [12].

Una plataforma middleware IoT debe conectar dispositivos, debe recopilar datos, estándares y debe ser capaz de escalar a un gran número de dispositivos enviando un gran número de mensajes. Para entregar valor verdadero más allá de los datos, una plataforma debe agregar cognición y seguridad. Con estas capacidades y los avances tecnológico, la plataforma IoT se convierte en un elemento con mucho potencial para los usuarios y las empresas [20].

Sus componentes funcionales son los siguientes [21]:

- Interoperabilidad: Compartir información y utilizar la misma información en diversos dominios de aplicaciones que utilizan diferentes interfaces de comunicación.
- Detección de contexto: El contexto es clave en el IoT y sus aplicaciones. Un gran número de sensores producirán grandes cantidades de datos, que no tendrán ningún valor, a menos que sean analizados, interpretados y comprendidos.
- Detección y gestión de dispositivos: El descubrimiento y la gestión de dispositivos permiten a todos los dispositivos de una red IoT descubrir todos los dispositivos vecinos y hacerlos conscientes de la presencia de otros dispositivos accesibles en la red. La ontología de dispositivos se utiliza para almacenar información sobre los dispositivos heterogéneos.
- Seguridad y privacidad: son responsables de la confidencialidad y autenticidad.
- Gestión de datos: Los datos son la clave en las aplicaciones de IoT. En este campo de estudio, los datos se refieren principalmente a datos detectados o cualquier información de infraestructura de red de interés para las aplicaciones. Un middleware IoT debe proporcionar servicios de administración de datos a aplicaciones, incluyendo adquisición de datos, procesamiento de datos (incluyendo preprocesamiento) y almacenamiento de datos. El preprocesamiento puede incluir filtrado de datos, compresión de datos y agregación de datos.

En segundo lugar, para entender bien estas características, se ha realizado un estudio de las principales plataformas IoT, cuyo objetivo final es conseguir la máxima interoperabilidad entre ellas. Dado que un estudio de estas características es de gran extensión, en este apartado se va a citar una muestra representativa de estas plataformas, tanto propietarias como de código abierto. También se incluyen las plataformas alojadas en la nube.

En tercer lugar, es importante analizar los principales problemas para lograr la interoperabilidad entre aplicaciones y servicios de estas plataformas IoT, que son intrínsecos a la naturaleza de estos. Los servicios y aplicaciones de IoT son:

- Múltiples y heterogéneos: Existen muchos servicios y aplicaciones en las plataformas IoT (por ejemplo, CEP, Historical DB, Big Data Processing, Visualization, Analytics, etc.).

- Distribuidas: Esto da lugar a grandes dificultades para conectar sus capacidades.
- Dinámicas: La demanda de nuevas funcionalidades a corto plazo, acelera los cambios y evolución de dichos servicios.

Las plataformas IoT no tienen la capacidad de interactuar entre sí en la capa de aplicación y servicios. Por esta razón, a nivel de Aplicación/Servicios, se han estudiado enfoques para hacer interoperables las aplicaciones y servicios proporcionados por las plataformas IoT heterogéneas que se presentan. En posteriores secciones, han sido analizados diferentes enfoques que son útiles para lograr la interoperabilidad entre servicios y aplicaciones. Allí se resaltará, la composición del servicio como el paradigma principal que contribuye a esta meta.

2.3. Interoperabilidad en IoT

El autor de la presente Tesis comenzó analizando el enfoque de la interoperabilidad en el contexto del proyecto INTER-IoT, Cuyo objetivo principal fue la definición y provisión de un sistema interoperable y abierto de IoT que permita la interconexión sin fisuras de dispositivos, redes de dispositivos y plataformas IoT. El proyecto propone mecanismos y herramientas de interoperabilidad en diferentes capas, desde el dispositivo hasta los datos y los servicios. Este trabajo ha ido evolucionando en el contexto de los diferentes proyectos definidos en el Capítulo 5, que han ido extendiendo el concepto de interoperabilidad a diferentes dominios y mediante diferentes plataformas y tecnologías.

Finalmente, este análisis de la situación de dicho campo concluyó en el estado del arte del proyecto ASSIST-IoT. El autor de la presente Tesis recopiló todos estos conocimientos adquiridos para ofrecer una visión general de la interoperabilidad que fuera el punto de partida de las primeras fases de dicho proyecto. Por tanto, allí el autor destacó cómo ha evolucionado el concepto de interoperabilidad en los últimos años y sentando las bases de las soluciones de Próxima Generación de Internet de las Cosas (NG-IoT). El proyecto ASSIS-IoT tiene como objetivo diseñar, implementar y validar una arquitectura abierta, descentralizada y de referencia de NG-IoT abierta y descentralizada, con sus correspondientes habilitadores, servicios y herramientas para ayudar a las aplicaciones centradas en el ser humano dentro de múltiples verticales. Este objetivo está totalmente alineado con la visión de la Internet de Próxima Generación, una Internet que responda a las necesidades fundamentales de las personas, como la confianza, la seguridad y la inclusión, a la vez que refleje los valores y las normas de las que disfrutan todos los ciudadanos en Europa.

Por tanto, este recorrido cronológico iniciado con INTER-IoT y que transita actualmente por ASSIST-IoT, refleja la evolución natural que han tenido las soluciones de interoperabilidad para ir abarcando cada vez más tecnologías, soluciones y adaptarse a las nuevas necesidades.

2.3.1. Concepto

Desde el punto de vista teórico, hay muchas definiciones diferentes sobre el término Interoperabilidad. Una explicación común es definirla como la capacidad de dos o más componentes de software para cooperar a pesar de las diferencias de lenguaje, interfaz y plataforma de ejecución [22]. La definición anterior puede ampliarse como la capacidad de un sistema informático para ejecutar programas de aplicación de diferentes proveedores, y para interactuar con otros ordenadores a través de redes locales o de área amplia, independientemente de su arquitectura física y sistemas operativos La interoperabilidad es factible a través de componentes de hardware y software que se ajustan a estándares abiertos, como los utilizados para Internet [23].

Para ser interoperables, dos o más sistemas deben ser capaces de intercambiar, interpretar y presentar los datos compartidos de una manera que sea entendida por el otro. Centrándonos en las expectativas de la Tesis Doctoral, es importante destacar que existen dos tipos de interoperabilidad de datos: la interoperabilidad sintáctica y la interoperabilidad semántica. En primer lugar, la interoperabilidad sintáctica implica la adopción de un formato de datos común y de protocolos de estructura de datos comunes. Es un requisito previo a la interoperabilidad semántica y permite que diferentes componentes de software cooperen, facilitando que dos o más sistemas se comuniquen e intercambien datos. En segundo lugar, la interoperabilidad semántica, se refiere a la capacidad de los sistemas informáticos para intercambiar datos significativos con un significado compartido y sin ambigüedades. Implica la adición de metadatos que vinculan cada elemento de los datos a un vocabulario controlado y compartido. Dentro de este vocabulario compartido hay enlaces asociados a una ontología, que es un modelo de datos que representa un conjunto de conceptos dentro de un dominio y las relaciones entre esos conceptos [24].

Desde el punto de vista del entorno de IoT, se observa que la definición del concepto IoT considera la interoperabilidad como un elemento clave. Por ejemplo, la International Telecommunication Union (ITU), una de las principales organizaciones de Standard Development Organisations (SDOs) en este ámbito, define IoT como una infraestructura global para la sociedad de la información, que permite la prestación de servicios avanzados mediante la interconexión de "cosas" (físicas y virtuales) basada en las tecnologías de la información y la comunicación interoperables existentes y en evolución [25]. Así, lograr la in-

teroperabilidad es uno de los principales objetivos de base en IoT. Se trata de conectar las cosas y hacerlas fácilmente accesibles, al igual que ocurre hoy en día con Internet. Por ello, en términos generales, la interoperabilidad puede definirse como una medida del grado en que diversos sistemas, organizaciones y/o individuos son capaces de trabajar juntos para lograr un objetivo común [26].

Aunque existen barreras para la plena realización de la visión de IoT, como la falta de interoperabilidad entre los sistemas y aplicaciones de IoT, muchas empresas ofrecen sistemas de IoT a sus clientes, sin percibir la necesidad de hacerlos comunicar con otras soluciones de este tipo. Desde el punto de vista comercial, el gasto de los recursos necesarios para apoyar la interoperabilidad, fuera de la solución existente, se percibe a menudo como poco razonable [27]. Además, la gran cantidad de sensores, protocolos, plataformas y aplicaciones aumenta la complejidad de la integración de las distintas soluciones [28]. Por tanto, se generan silos IoT en el que las soluciones realizadas para un sistema, se mantienen aisladas de los demás sistemas existentes. Para contrarrestar este proceso de generación de silos de IoT, la Unión Europea ha financiado proyectos de investigación centrados en proporcionar interoperabilidad en el ecosistema IoT. Además, han surgido múltiples iniciativas y soluciones para acercar a las empresas la necesidad de esta interoperabilidad.

2.3.2. Propósito y retos

La Comisión Europea destaca el papel esencial de la interoperabilidad y la estandarización en su Comunicación relativa a las "Prioridades de normalización de las TIC para el mercado único digital"[29]. A continuación, se ofrece un resumen de las directrices publicadas:

"*El panorama de la IoT está actualmente fragmentado porque hay muchas soluciones propietarias o semicerradas junto a una gran cantidad de normas existentes. Esto puede limitar las innovaciones que abarcan varios ámbitos de aplicación. La aplicación y validación a gran escala de soluciones y normas transversales es ahora la clave de la interoperabilidad, la fiabilidad y la seguridad en la UE y en todo el mundo.*

La Unión Europea necesita un enfoque de plataforma abierta que admita múltiples dominios de aplicación y atraviese los silos para crear ecosistemas de IoT competitivos. Esto requiere normas abiertas que apoyen toda la cadena de valor, integrando múltiples tecnologías, sobre la base de una cooperación internacional racionalizada que se apoye en un marco de derechos de propiedad intelectual que permita un acceso fácil y justo a las patentes esenciales de estándar. Se destacan los siguientes puntos:

- *Fomentar un entorno interoperable para la Internet de las cosas, centrándose en arquitecturas, protocolos e interfaces de referencia, la promoción de interfaces de programación de aplicaciones (API) abiertas, el apoyo a las actividades de innovación relacionadas con la implementación y experimentación de referencia y el desarrollo de las normas de interoperabilidad que faltan (especialmente en el ámbito intersectorial de la interoperabilidad semántica).*
- *Promover un espacio de IoT interoperable que trascienda los límites geográficos, y un sistema abierto de identificación y autenticación de objetos.*
- *Explorar opciones y principios rectores, incluido el desarrollo de normas, para la confianza, la privacidad y la seguridad de extremo a extremo, por ejemplo, a través de una etiqueta de IoT de confianza.*
- *Promover la adopción de las normas de IoT en la contratación pública para evitar el bloqueo, especialmente en el ámbito de los servicios de las ciudades inteligentes, el transporte y los servicios públicos, incluidos el agua y la energía.*"

Una de las iniciativas que ha llevado a cabo una investigación detallada, ha recopilado mucha información, ha obtenido conclusiones claras y ha proporcionado algunas orientaciones sobre la interoperabilidad es CREATE-IoT [30]. El objetivo de este proyecto es estimular la colaboración entre iniciativas de IoT, fomentar la adopción de IoT en Europa y apoyar el desarrollo y el crecimiento de ecosistemas de IoT basados en tecnologías y plataformas abiertas. Por ejemplo, publicaron algunos entregables como *Estrategia y plan de coordinación para la interoperabilidad de IoT y enfoques estándar* [31], *Recomendaciones para los puntos comunes y los perfiles de interoperabilidad de las plataformas de IoT* [32] o *Evaluación de la convergencia y la interoperabilidad en las plataformas LSP* [33], que resumen los resultados de los pilotos de IoT a gran escala. Con el fin de abordar la interoperabilidad y la integración a través de plataformas abiertas de IoT, el proyecto CREATE-IoT proporciona una lista de los principales requisitos que se esperan para definir un marco de interoperabilidad de IoT. Dichos requisitos son los siguientes [34]:

- Soporte de protocolos de comunicación IoT comunes.
- Soporte para las comunicaciones Machine to Machine (M2M).
- Protocolos estándar para las comunicaciones de los dispositivos.
- Soporte de las principales plataformas de middleware de IoT.

- Extensibilidad para diferentes tipos de sensores.
- Capacidad de detección de dispositivos de usuario.
- Interoperabilidad sintáctica.
- Interoperabilidad semántica.
- Capacidades de pasarela y conversión de protocolos.
- Identificación/nominación única de dispositivos.
- Protocolos estándar para las comunicaciones de los dispositivos.

Algunos elementos clave a tener en cuenta para crear un marco de interoperabilidad completo son los siguientes [35]:

- Arquitecturas de referencia.
- Soporte de diseño y desarrollo.
- Plataformas y tecnologías.
- Normas y actividades prenormativas.
- Alineación con otras arquitecturas de IoT.

2.3.3. Aproximación a las capas en arquitecturas de interoperabilidad IoT

La interoperabilidad es un concepto complejo con múltiples aspectos a considerar. La primera tarea, desde el punto de vista del análisis del material existente en la literatura, es acotar y clasificar los elementos que aborda la interoperabilidad. Existen diferentes clasificaciones de los distintos aspectos de la interoperabilidad, también llamadas niveles de interoperabilidad. La clasificación denominada Levels of Conceptual Interoperability Model (LCIM) [36] consta de siete niveles que van desde la no interoperabilidad hasta la interoperabilidad conceptual, anotados de Nivel 0 a Nivel 6. La clasificación, creada en el contexto de la teoría de la simulación, proporciona los siguientes niveles de interoperabilidad:

- Nivel 0 - No hay interoperabilidad: No hay conexión ni interoperabilidad.
- Nivel 1 - Técnico: Tienen conexión(es) técnica(s) y pueden intercambiar datos entre sistemas. La premisa son los protocolos de comunicación comunes (como HTTP, TCP/IP, UDP/IP, etc.) y la conectividad de red del dominio.

- Nivel 2 - Sintáctico: Tiene un protocolo acordado para intercambiar las formas correctas de datos en el orden correcto, pero el significado de los elementos de datos no está establecido. Los contenidos claramente definidos son el formato de la información intercambiada (por ejemplo: XML, SOAP; JSON, etc.).
- Nivel 3 - Semántico: Los sistemas interoperables intercambian un conjunto de términos que pueden analizar semánticamente. La información definida es el significado de los datos y el contenido de la información intercambiada.
- Nivel 4 - Pragmático: Los sistemas interoperables serán conscientes del contexto (estados y procesos del sistema) y del significado de la información que se intercambia. La información definida es el uso de los datos y el contexto de la información intercambiada.
- Nivel 5 - Dinámico: Los sistemas interoperables son capaces de reorientar la producción y el consumo de información en función de los cambios de significado que se entienden, debido al cambio de contexto a medida que pasa el tiempo. La información definida es el efecto de los datos y el efecto de la información intercambiada.
- Nivel 6 - Conceptual: Los sistemas interoperables de este nivel son completamente conscientes de la información, los procesos, los contextos y los supuestos de modelado de los demás. Se centra en la composición y el dominio de la abstracción del modelado. La información definida son los supuestos, las restricciones, etc. y los contenidos definidos son un modelo conceptual documentado.

El marco europeo de interoperabilidad para los servicios paneuropeos de administración electrónica [37] define tres niveles: interoperabilidad técnica, semántica y organizativa. Sin embargo, el ETSI y la AIOTI [29] ofrecen una clasificación más relacionada con el IoT. En dicho documento ambos organismos definen cuatro niveles: interoperabilidad técnica, sintáctica, semántica y organizativa.

- Interoperabilidad técnica: Suele estar asociada a los componentes de hardware/software, sistemas y plataformas que permiten la comunicación entre máquinas. Suele centrarse en los protocolos de comunicación y en la infraestructura necesaria para que esos protocolos funcionen. Algunos protocolos de uso común son: CoAP, HTTP, WebSockets, MQTT y AMQP.

- Interoperabilidad sintáctica: suele estar asociada a los formatos de datos. Los mensajes transferidos por los protocolos de comunicación deben tener una sintaxis y una codificación bien definidas. El contenido puede representarse mediante sintaxis de transferencia de alto nivel, como, por ejemplo, XML, JSON y RDF.
- Interoperabilidad semántica: Suele estar asociada al significado del contenido y se refiere a la interpretación humana, más que a la de la máquina, del contenido. Así, la interoperabilidad a este nivel significa que existe un entendimiento común entre las personas sobre el significado del contenido que se intercambia. Las descripciones interpretables por la máquina son aplicables a diferentes aspectos de la interoperabilidad semántica, por ejemplo:
  - Modelos y tipos de datos.
  - Modelos que describen cómo interactuar con las cosas.
  - Marcos para describir diferentes versiones de dispositivos y software.
  - Descripciones semánticas de las cosas, por ejemplo, una luz, un aire acondicionado, etc.
  - Descripciones semánticas del contexto, por ejemplo, las habitaciones de un edificio.
  - Políticas de privacidad que cubren el uso de datos personales.
  - Políticas de seguridad, por ejemplo, control de acceso y actualizaciones de seguridad.
  - Contratos inteligentes y términos y condiciones.

  Para minimizar las barreras de los servicios digitales que abarcan diferentes plataformas, es muy necesario fomentar la convergencia de los marcos y lenguajes de modelado. Algunos trabajos relevantes son:

  - La Web de las Cosas del W3C, que utiliza JavaScript Object Notation for Linked Data (JSON-LD) para describir las cosas como objetos con propiedades, acciones y eventos, utilizando JSON Schema para describir los tipos de datos.
  - El Marco de Descripción de Recursos (RDF) del W3C, que utiliza gráficos con arcos etiquetados dirigidos.
  - Lenguaje de ontología web del W3C (OWL) y esquema RDF.
  - Diagramas de relación de entidades.
  - Lenguaje de modelado unificado (UML).

  - Modelado de objetos y roles (ORM).

- Interoperabilidad organizativa: Es la capacidad de las organizaciones para comunicar y transferir eficazmente datos significativos (información) aunque utilicen distintos sistemas de información en infraestructuras muy diferentes, posiblemente en distintas regiones geográficas y culturas. La interoperabilidad de las organizaciones depende del éxito de la interoperabilidad técnica, sintáctica y semántica.

El anterior enfoque por capas es la clasificación más consensuada para la descripción de la interoperabilidad en los sistemas IoT. Sin embargo, otra clasificación interesante se basa en los diversos elementos que componen los sistemas IoT (dispositivos, comunicación, servicios, aplicaciones, etc.). Propone que la interoperabilidad de IoT puede verse desde diferentes perspectivas, como la interoperabilidad de dispositivos, la interoperabilidad de redes, la interoperabilidad sintáctica, la interoperabilidad semántica y la interoperabilidad de plataformas. Este enfoque por capas es interesante para definir claramente en qué partes de las arquitecturas IoT se centran las soluciones del mercado y cómo se resuelven sus requisitos de interoperabilidad [26].

Se presentan a continuación otras aproximaciones que han contribuido a categorizar la aproximación seguida por la presente Tesis Doctoral.

AIOTI ha desarrollado la Arquitectura de Alto Nivel que define tres capas y proporciona formas más completas de caracterizar y clasificar los estándares aplicables [38]. Las tres capas son una forma de estructurar un IoT con las siguientes características:

- La capa de aplicación contiene el intercambio de información y los métodos de interfaz utilizados en las comunicaciones de proceso a proceso.

- La capa de IoT agrupa funciones específicas de IoT, como el almacenamiento y el intercambio de datos, y las expone a la capa de aplicación a través de Interfaces de Programación de Aplicaciones (API). La capa de IoT hace uso de los servicios de la capa de red

- Los servicios de la capa de red se pueden agrupar en servicios de plano de datos, proporcionando conectividad de corto y largo alcance, transporte de datos entre entidades, y servicios de plano de control, como ubicación, activación de dispositivos o calidad de servicio.

El proyecto CREATE-IoT del programa de pilotos europeos de IoT a gran escala, surgió en 2018 con una arquitectura de referencia de IoT 3D independiente del dominio de aplicación. Esta se basa en capas, funciones transversales y propiedades no funcionales [32]. La Capa Física se compone de dispositivos

para la recogida de datos y para la toma de decisiones de actuación. La Capa de Comunicación de Red define las tecnologías y los protocolos para transportar los datos, incluidas las pasarelas. La Capa de Procesamiento, que incluye capacidades de computación de borde para analizar los flujos de datos, se encuentra separada de la Capa de Almacenamiento, que puede ser centralizada o descentralizada para el análisis a largo plazo. La Capa de Abstracción cubre las características semánticas de los datos, para crear modelos de nivel superior del mundo real. Las tres capas superiores están destinadas a desarrollar y orquestar servicios y aplicaciones de IoT, herramientas para la visualización avanzada, el análisis y la elaboración de informes, así como la integración con las soluciones existentes a nivel empresarial y los sistemas de terceros.

Otros proyectos como IoT-A [39] se enfocan en proporcionar una arquitectura de referencia, un conjunto de buenas prácticas, directrices y un punto de partida para generar arquitecturas específicas de IoT. Su punto de vista funcional sigue una estructura modular de Grupos Funcionales con un conjunto de componentes, como Gestión de Procesos IoT, Organización de Servicios, Entidad Virtual, Servicio IoT, Comunicación; y dos transversales a los demás, que son Seguridad y Gestión.

El proyecto ACTIVAGE clasifica los componentes funcionales de las plataformas IoT desde el punto de vista de la interoperabilidad [40]. Parte de que las plataformas IoT ofrecen la posibilidad de crear aplicaciones y servicios en un entorno estructurado que está formado por componentes funcionales. Los bloques suelen ser comunes y se repiten en muchas aplicaciones y servicios de IoT. Esta característica de las plataformas IoT contribuye al ciclo de desarrollo y al tiempo de comercialización, al tiempo que reduce el coste global de la implementación de IoT. La plataforma IoT es un middleware que proporciona servicios a los dispositivos y aplicaciones ubicados en la red IoT y permite la conectividad del sistema. Desde este punto de vista, una plataforma IoT de extremo a extremo consta de ocho importantes bloques arquitectónicos para el desarrollo de aplicaciones y la habilitación de servicios:

- Conectividad y normalización: elimina la heterogeneidad de los datos, ya que reúne diferentes protocolos y formatos de datos en una interfaz de software que garantiza la transmisión de datos y la interacción con todos los dispositivos.
- Gestión de dispositivos: garantiza que los dispositivos de red conectados funcionen correctamente, ejecutando sin problemas parches y actualizaciones para el software y las aplicaciones que se ejecutan en el dispositivo o las pasarelas de borde.

- Almacenamiento de datos: el almacenamiento escalable de los datos de los dispositivos lleva los requisitos de las bases de datos híbridas basadas en la nube a un nuevo nivel en términos de volumen, variedad, velocidad y veracidad de los datos.
- Procesamiento y gestión de acciones: los datos son procesados y gestionados con disparadores de acción basados en reglas que permiten la ejecución de acciones inteligentes basadas en datos de sensores específicos.
- Análisis: realiza una serie de análisis complejos, desde la agrupación básica de datos y el aprendizaje automático profundo hasta el análisis predictivo, extrayendo el máximo valor del flujo de datos del IoT.
- Visualización: permite a los usuarios ver patrones y observar tendencias a partir de paneles de visualización en los que los datos se representan a través de gráficos de líneas, apilados o circulares, y de modelos 2D o incluso 3D.
- Herramientas adicionales: permiten a los desarrolladores de IoT crear prototipos, probar y comercializar el caso de uso de IoT creando aplicaciones del ecosistema de la plataforma para visualizar, gestionar y controlar los dispositivos conectados.
- Interfaces externas: se integran con sistemas de terceros y con el resto del ecosistema a través de Interfaces de Programación de Aplicaciones (API), Software Development Kit (SDK) y pasarelas.

Finalmente, es importante resaltar que el trabajo realizado en la presente Tesis Doctoral esta totalmente alineado con las clasificaciones presentadas anteriormente y sigue la aproximación por capas definida en el proyecto INTER-IoT. Dicha aproximación está basada en: (i) interacción dispositivo a dispositivo basado en mecanismos multiprotocolo y acceso; (ii) módulos interoperables definidos por software para movilidad y encaminamiento; (iii) marco de gestión abierto para objetos inteligentes; (iv) virtualización y pasarela de servicios de aplicaciones inteligentes de IoT; y (v) una ontología común que facilita el acceso a los datos heterogéneos, datos que serán recolectados y administrados por plataformas IoT integradas [41]. Además, también sigue su modelo de referencia para la interoperabilidad de las plataformas de IoT. Está formado por una arquitectura de referencia y un sistema de interoperabilidad completo [42]. Esto se detalla en el Capítulo 3, donde se presenta la arquitectura y el diseño de la solución presentada en la Tesis Doctoral.

2.4. Estándares e iniciativas

Teniendo en cuenta el crecimiento estimado del ecosistema de la IoT y la creciente necesidad de interoperabilidad de las soluciones de la IoT, la normalización desempeña un papel importante en este contexto al promover las mejores prácticas, la integración y la interoperabilidad de los sistemas y los requisitos de privacidad y seguridad [28]. Las normas comunes aseguran la interoperabilidad, ya que garantizan que las tecnologías funcionen juntas de forma fluida y fiable y fomentan la investigación y la innovación. La interoperabilidad efectiva garantiza que los dispositivos conectados puedan comunicarse sin problemas entre sí, independientemente del fabricante, el sistema operativo u otros componentes técnicos [31].

Para mejorar el estado de la interoperabilidad del IoT, las investigaciones han aprovechado numerosos enfoques y tecnologías que denominan enfoques de gestión de la interoperabilidad [11]. Los principales enfoques son los siguientes

- Uso de adaptadores, pasarelas o conectores: Estos componentes se centran en el desarrollo de una herramienta intermedia, a veces denominada mediadores, para mejorar la interoperabilidad entre los dispositivos IoT. Su objetivo es actuar como puente entre diferentes especificaciones, datos, estándares y middleware, etc. Principalmente, realizando una conversión entre el protocolo del dispositivo emisor y el protocolo del dispositivo receptor [43][44].

- Tecnologías de red: Se pueden utilizar diferentes protocolos y tecnologías de red para proporcionar interoperabilidad de red en la IoT. Estos enfoques podrían ser enfoques basados en IP, Redes Definidas por Software (SDN), virtualización de funciones de red o computación en la niebla [45][46].

- APIs abiertas: Las APIs abiertas bien documentadas proporcionan a los desarrolladores un acceso claro a las funcionalidades y servicios [47].

- Arquitectura orientada al servicio: Facilita la interoperabilidad sintáctica entre dispositivos heterogéneos y a través de todos los sistemas, porque se construye sobre la capa de red y la información procesada puede gestionarse fácilmente a través de diferentes componentes de servicio [48] [49].

- Tecnologías de la web semántica: Las tecnologías de la web semántica desarrolladas, por ejemplo, por el W3C, como el Marco de Descripción de Recursos (RDF), SPARQL y el Lenguaje de Ontología Web (OWL), pueden utilizarse para describir recursos en la web. En la actualidad, los

mismos estándares se utilizan en muchos ámbitos diferentes, incluido el de IoT. Las ontologías en IoT son un conjunto de objetos y relaciones utilizados para definir y representar un área de interés. Representan una tecnología de abstracción cuyo objetivo es ocultar la heterogeneidad de las entidades, actuando como mediador entre el proveedor de aplicaciones y los consumidores y para apoyar su correspondencia semántica [50][51][52].

- Estándares abiertos: Un estándar es un marco de especificación que ha sido aprobado por una organización reconocida, o que es generalmente aceptado y ampliamente utilizado por la industria. Los estándares abiertos son un medio importante para proporcionar interoperabilidad entre y dentro de diferentes dominios [53][54][55][56].

La estandarización en el panorama de la IoT es importante porque reduce las diferencias entre los protocolos. Varias iniciativas están trabajando en el desarrollo de normas de IoT, principalmente impulsadas por agencias gubernamentales, organismos de normalización y empresas de la industria. Durante la última década se están aplicando cientos de normas a IoT. Varias son las organizaciones que las producen y mantienen. Según el análisis realizado por el presente autor para el proyecto ASSIST-IoT [57], las principales actividades de estándares IoT relacionados o interesantes para la presente Tesis Doctoral son los siguientes:

Organización	Descripción
ETSI [58]	ETSI, el Instituto Europeo de Estándares de Telecomunicaciones, produce estándares aplicables a nivel mundial para las Tecnologías de la Información y las Comunicaciones (TIC), incluidas las tecnologías fijas, móviles, de radio, convergentes, de transmisión e Internet. ETSI Smart M2M se centra actualmente en: (i) participación y contribución a las iniciativas de la UE en las áreas de M2M e IoT, (ii) estandarizar un marco para una ontología abierta (SAREF) que permita el intercambio de información entre dispositivos y servidores de IoT utilizando diferentes tecnologías y (iii) Apoyo a la Iniciativa AIOTI. Además, otras actividades de ETSI están relacionadas con IoT, en particular proporcionando estándares para la interoperabilidad de la red.

IEEE [59]	Las normas del IEEE establecen especificaciones y mejores prácticas basadas en los conocimientos científicos y tecnológicos actuales. Estas normas abarcan la conectividad por cable e inalámbrica, el cifrado, la seguridad de los datos, etc. La norma IEEE P2413 trabaja con un enfoque descendente y sigue las recomendaciones para la descripción de arquitecturas definidas en la norma ISO/IEC/IEEE 42010, que (i) Proporciona una ontología básica para la descripción de arquitecturas, (ii) Especifica las disposiciones que hacen cumplir las propiedades deseadas de los marcos de arquitectura, (iii) Puede utilizarse para establecer una práctica coherente para desarrollar marcos de arquitectura y (iv) Puede utilizarse para evaluar la conformidad de un marco de arquitectura.
IETF [60]	Su objetivo es hacer que Internet funcione mejor produciendo documentos técnicos relevantes y de alta calidad que influyan en la forma en que la gente diseña, utiliza y gestiona Internet. El IETF apoya el IoT desde 2005, cuando comenzó con 6LoWPAN. Los grupos de trabajo del IETF relacionados con el IoT son: 6Lo (IPv6 sobre redes de nodos con recursos limitados); ROLL (Routing Over Low-power and Lossy networks); CORE (Constrained RESTful Environments); ACE (Autenticación y Autorización para Entornos Restringidos); CBOR (Concise Binary Object Representation Maintenance and Extensions); 6tisch (IPv6 sobre el modo TSCH de IEEE 802.15.4e); IPWAVE (Acceso inalámbrico IP en entornos vehiculares); IPWAN (IPv6 sobre redes de área amplia de baja potencia); Detnet (Deterministic Networking); LWIG (Guía de Implementación Ligera)

ISO/IEC [61]	La ISO y la IEC tienen un comité técnico conjunto llamado JTC 1. El JTC 1 estableció un Grupo de Trabajo Especial (WG10) sobre IoT en 2012 que se convirtió en un WG formal en 2015 y pasó a ser un Subcomité formal en 2017. Los trabajos actuales son: ISO/IEC 30141, Arquitectura de referencia del Internet de las cosas; ISO/IEC 20924, Definición y vocabulario; ISO/IEC 21823-1, Interoperabilidad para los sistemas del Internet de las Cosas, Parte 1: Marco de trabajo; ISO/IEC PDTR 22417, Casos de uso de IoT.
ITU-T [62]	La ITU-T desarrolla normas internacionales que actúan como elementos definitorios de la infraestructura mundial de tecnologías de la información y la comunicación. Las actividades de normalización de la ITU-T se gestionan en Comisiones de Estudio (SG), dos de ellas de especial interés: SG17 sobre seguridad y SG20 sobre IoT y aplicaciones y Smart Cities.
oneM2M [63]	El propósito y el objetivo de oneM2M es desarrollar especificaciones técnicas que aborden la necesidad de una capa de servicio M2M común que pueda integrarse fácilmente en diversos equipos y programas informáticos y en la que se pueda confiar para conectar la miríada de dispositivos sobre el terreno con los servidores de aplicaciones M2M de todo el mundo. OneM2M trata de lograr la interoperabilidad a través de diferentes esfuerzos de estandarización. Los distintos grupos de trabajo elaboran especificaciones para una arquitectura de referencia (ARC WG), un protocolo de mensajería (PRO WG), una gestión de datos, abstracción y semántica (MAS WG), pero también pruebas de interoperabilidad (TST WG).

W3C [64]	W3C pretende contrarrestar la fragmentación de IoT a través de un marco de interoperabilidad semántica que desvincula las aplicaciones de los estándares, protocolos, formatos de datos y patrones de comunicación subyacentes de IoT, y permite el descubrimiento, la composición y la adaptación a las variaciones de los dispositivos de diferentes proveedores. El objetivo es reducir los costes y riesgos del desarrollo de soluciones IoT. La Web de las Cosas se basa en el trabajo del W3C sobre Linked Data y abarca el modelo de interacción expuesto a las aplicaciones en términos de propiedades, acciones y eventos para las cosas, los modelos semánticos que describen los tipos de cosas y sus relaciones, y los metadatos relativos a la seguridad, la confianza, la privacidad, los acuerdos de nivel de servicio y otros términos y condiciones.

Tabla 2.1: Organismos de estandarización

2.5. Interoperabilidad IoT en las capas Middleware y Aplicaciones y Servicios

Esta sección ofrece un análisis de las principales aproximaciones existentes que proporcionan mecanismos para facilitar la interoperabilidad en las capas objeto de estudio en la presente Tesis Doctoral. En primer lugar, se ofrecen soluciones existentes que pueden contribuir a la creación de una solución de abstracción en la capa middleware de modo que se puedan conectar diferentes sistemas o plataformas a dicha capa de abstracción. En segundo lugar, se ofrece una visión general de las plataformas IoT existentes. En tercer lugar, se presenta un análisis de las técnicas y tecnologías existentes que ofrecen funcionalidades que pueden facilitar la interoperabilidad de los servicios y aplicaciones de las plataformas IoT.

2.5.1. Aproximación en la capa Middleware

Una descripción general define al Middleware como el software que se sitúa entre un sistema operativo y las aplicaciones que se ejecutan en él [65]. Desde un punto de vista más específico, la capa middleware se refiere a la infraestructura de software que permite la comunicación entre los diferentes componentes de un sistema, generalmente en forma de solicitud y respuesta o una comunicación

de conexión sostenida para transmisión de datos [66]. Principalmente se puede encargar de gestionar los datos, los servicios de aplicaciones, la mensajería, la autenticación y la gestión de las API. Es el hilo conductor entre aplicaciones, datos y usuarios [67][68][69].

Se encuentran dentro de la definición de middleware desde servidores web hasta sistemas de autenticación y herramientas de mensajería [70]. Entre otras cosas las soluciones middleware permiten:

- Desarrollar nuevas aplicaciones.
- Optimizar aplicaciones existentes.
- Ofrecer funcionalidades de integración, permitiendo la conectividad de diferentes sistemas.
- Ofrecer APIs para acceder a ellos.
- Transmisión de datos en tiempo real.
- Trasmisión asíncrona de la información, replicando los datos en un almacén intermedio, que puede compartirlos con varias aplicaciones.

Desde el punto de vista de IoT, es una capa clave porque generalmente se describe como un sistema de software diseñado para ser el intermediario entre los dispositivos y las aplicaciones IoT [18]. Una plataforma de software definida como middleware tiene el objetivo de conectar aplicaciones y datos de las cosas que se comunican a través de interfaces heterogéneas utilizando diversas tecnologías. Este tipo de soluciones tienen que proporcionar una capa de abstracción, de adaptación o conexión y ofrecer los servicios mediante APIs [66].

Existen varios enfoques para el middleware, sobre todo en función del tipo de aplicaciones y servicios que va a conectar. Históricamente, se utilizaba el modelo Object Request Broker (ORB) [71], en el que un proceso podía realizar una Llamada a Procedimiento Remoto (RPC). Aunque existían mecanismos para facilitar este modelo, la comunicación era puramente punto a punto entre los dos procesos, y no proporcionaba ninguna entidad intermedia que pudiera facilitar los problemas de interoperabilidad y otras cuestiones relacionadas. Un problema común con el acceso directo a los servicios es que las solicitudes son enviadas al proveedor de servicios, independientemente de su estado actual o nivel de carga. Esto puede crear problemas de carga en los que un servidor ocupado está tratando de procesar varias peticiones que actualmente están haciendo uso de la mayor parte de sus recursos, pero debe atender la recepción de la nueva petición, cargando el servidor aún más, y potencialmente impidiendo

que complete con éxito cualquier trabajo. Además, al mismo tiempo se pierden nuevas peticiones de servicios.

Para evitar este tipo de problemas, se utiliza un enfoque de Middleware Queue, en el que se despliega una cola de mensajes para recibir las peticiones a uno o varios servicios, y los proveedores de servicios acuden a la cola para recoger una nueva petición siempre que juzguen tener los recursos necesarios para realizarla. Este método de Pull evita la sobrecarga del servidor, permite gestionar de forma transparente varias instancias de servicio en una sola cola simplifica mucho el traslado o la actualización de los servicios y puede escalarse mediante nuevas instancias de cola.

Una división general de la arquitectura de los middlewares es separar entre Middleware orientado a Mensajes (MoM) y Middleware orientado a Servicios (SoM). Cada uno tiene dos tipos de implementación: implementaciones de servidor que corresponden a la arquitectura cliente-servidor e implementaciones descentralizadas presentes en los sistemas peer-to-peer.

La siguiente sección se centrará en los middlewares orientados a servicios, mediante la aproximación a la interoperabilidad middleware mediante las plataformas IoT. Estas plataformas implementan e interconectan diversas tecnologías y datos IoT para ofrecer diversos servicios adicionales sobre una arquitectura middleware.

La implementación de Middleware orientado a mensajes propone que la funcionalidad principal de un middleware resida en uno o varios servidores centrales denominados brokers de mensajes. Este elemento software es responsable de la entrega consistente, fiable y tolerante a fallos de los mensajes a cualquier cliente dispuesto. Los brokers de mensajes pueden poner en cola, retrasar, almacenar, difundir, traducir, retransmitir o reintentar la entrega de mensajes, junto con otras funcionalidades orientadas a los mensajes. Las colas de mensajes son muy utilizadas en los middlewares de comunicación debido a su escalabilidad, modularidad y flexibilidad.

Existen varias implementaciones comerciales y de código abierto de colas de mensajes que proporcionan las características básicas y otras más avanzadas. Prácticamente, todas ellas permiten que los procesos se suscriban a una cola en concreto o a un tema, y que publiquen un mensaje en un tema de la cola. Además, los temas pueden configurarse para entregar un mensaje recibido a una y sólo una parte suscrita, o a todas [72].

Otras características que implementan algunas colas son:

- Acuse de recibo, en el que se informa a un publicador de que la cola ha recibido la solicitud.
- Política de entrega, como por ejemplo, al menos una vez o como máximo una vez.

- Persistencia, en la que los mensajes aceptados nunca se pierden, ni siquiera en caso de fallo del Middleware.
- Sincronía, en la que se pone a disposición un nuevo mensaje antes de que se responda al anterior.

La siguiente tabla muestra una lista de las principales implementaciones comerciales de colas de mensajes:

Tecnología	Descripción
RabbitMQ [73]	RabbitMQ es ligero y fácil de implementar en local y en la nube. Es compatible con múltiples protocolos de mensajería. RabbitMQ se puede implementar en configuraciones distribuidas y federadas para cumplir con los requisitos de alta disponibilidad y gran escala. Se ejecuta en muchos sistemas operativos. Proporciona una amplia gama de herramientas de desarrollo para los idiomas más populares. Es un bróker de mensajería tradicional y adecuado para muchos protocolos de mensajería. RabbitMQ es un sistema flexible y con un encaminamiento de mensajes más completo que otras alternativas como Apache Kafka.
Apache Kafka [74]	Apache Kafka es una plataforma de streaming de eventos distribuidos de código abierto para canalizaciones de datos de alto rendimiento, análisis de streaming, integración de datos y aplicaciones de misión crítica. Tiene un rendimiento y escalabilidad superior a RabbitMQ, por lo que tiende a ser usado en proyectos Big Data en los que se desea una alta tolerancia a fallos y procesamiento en tiempo real. Se encuentra regularmente entre los cinco proyectos más activos de la Apache Software Foundation.

ActiveMQ [75]	Apache ActiveMQ es un broker de mensajes de código abierto, multiprotocolo y basado en Java. Es compatible con los protocolos estándar del sector, por lo que los usuarios obtienen las ventajas de las opciones de los clientes en una amplia gama de lenguajes y plataformas. Permite integrar las aplicaciones multiplataforma mediante el protocolo AMQP, intercambiar mensajes entre las aplicaciones Web utilizando STOMP sobre Websockets y gestionar los dispositivos IoT mediante MQTT. Es una solución potente y flexible.
AmazonMQ [76]	Amazon MQ es un servicio de agentes de mensajes en la nube administrado para Apache ActiveMQ y RabbitMQ que facilita la configuración y la operación de agentes de mensajes en AWS. Amazon MQ reduce las responsabilidades operativas de los usuarios mediante el aprovisionamiento, la configuración y el mantenimiento de agentes de mensajes.
ZeroMQ [77]	ZeroMQ parece una biblioteca de redes incrustada, pero actúa como un marco de trabajo de concurrencia. Ofrece una biblioteca de mensajería asíncrona multiplataforma, desarrollada por una comunidad de colaboradores como proyecto de código abierto. Proporciona una cola de mensajes para un entorno distribuido, sin necesidad de un broker de mensajes. Está escrita en C++, pero proporciona enlaces en muchos lenguajes como Java, Python, Scala, PHP y otros. Este gestor de colas de mensajes soporta aplicaciones distribuidas, en un esfuerzo por proporcionar una biblioteca de mensajería empresarial rápida, descentralizada y de uso general.
Mosquitto [78]	Eclipse Mosquitto es un broker de mensajes de código abierto que implementa diferentes versiones del protocolo MQTT. Es ligero y se puede utilizar en todos los dispositivos, desde ordenadores de placa única de baja potencia hasta servidores completos. El protocolo MQTT proporciona un método ligero para llevar a cabo la mensajería utilizando un modelo de publicación/suscripción. Esto lo hace adecuado para la mensajería de IoT, por ejemplo, con sensores de baja potencia o dispositivos móviles, como teléfonos, ordenadores integrados o microcontroladores.

HiveMQ [79]	HiveMQ es un broker MQTT y una plataforma de mensajería basada en el cliente diseñado para el envío rápido, eficiente y fiable de datos hacia y desde dispositivos IoT conectados. Utiliza el protocolo MQTT para el envío instantáneo y bidireccional de datos entre el dispositivo y los sistemas.

Tabla 2.2: Implementaciones comerciales de soluciones de colas de mensajes

El rendimiento, la persistencia y las directivas son las características que principalmente varían en las diferentes implementaciones. Por lo tanto, según los diferentes requisitos de los escenarios donde aplicarse y para obtener el funcionamiento esperado, se tenderá a priorizar la selección de una implementación comercial del software de colas de mensajes respecto a las demás.

El bróker seleccionado en la Tesis Doctoral para implementar las colas de mensajes en la implementación del componente de interoperabilidad middleware entre plataformas IoT ha sido RabbitMQ. Esto es debido a su popularidad, a que es de código abierto, dispone de muy buena documentación, es mantenido por su comunidad, ligero, fácil de desplegar, con soporte a múltiples protocolos de mensajes y puede ser desplegado en configuraciones federadas y distribuidas. En contraposición, el componente que actúa como mediador semántico en el proyecto INTER-IoT ha utilizado para su implementación el software Apache Kafka.

2.5.2. Plataformas IoT

Este apartado pretende ofrecer, de una manera concisa, un análisis del estado del arte sobre las plataformas IoT existentes. Estas soluciones software son el pilar básico de la presente Tesis Doctoral. No obstante, es importante señalar que este análisis ha sido un trabajo periódico durante estos años que comenzó con el estudio previo realizado al principio de la Tesis y ha ido actualizándose con un análisis de las publicaciones científicas sobre el tema, la participación en proyectos de investigación relacionados (Capítulo 5) y la colaboración con iniciativas para la interoperabilidad. Desde un punto de vista técnico, se han desplegado y probado las plataformas IoT, se han configurado, se han conectado sensores a ellas y se han testeado los servicios y aplicaciones que contienen. Además de las de pruebas, las plataformas se han puesto en marcha en pilotos, casos de uso o demostradores, donde ha sido necesario analizar su funcionamiento desde el punto de vista de su puesta en marcha para producción. Este trabajo, tanto preliminar como constante, ha sido necesario porque era crítico

tener un alto grado de conocimiento de los detalles técnicos y funcionamiento de plataformas para ofrecer una serie de mecanismos de interoperabilidad sobre ellas. Finalmente, es importante indicar que durante este periodo han aparecido nuevas plataformas, algunas han dejado de ser mantenidas y otras han evolucionado constantemente.

Para elaborar esta sección se ha utilizado el trabajo realizado o en el que ha contribuido el presente autor. En primer lugar, el estado del arte de los proyectos INTER-IoT [80], ACTIVAGE [81], DataPorts [82] y ASSIST-IoT [57]. En segundo lugar, publicaciones científicas en las que ha participado el autor, todas han girado en mayor o menor grado alrededor de esta temática, y se encuentran listadas en Capítulo 1 de la presente Tesis Doctoral. En tercer lugar, las conclusiones obtenidas por las iniciativas europeas en las que se ha colaborado a través los proyectos de investigación, CREATE-IoT [83] y UNIFY-IoT [1], donde el análisis de las plataformas ofrece una visión general, pormenorizada y apoyada por una comunidad de expertos. Finalmente, también se han analizado publicaciones externas, tanto científicas como industriales [84][85][86][87][88][89][90]. De esta manera, se obtiene una visión general más allá de la relacionada con proyectos de investigación y se obtiene una visión desde el punto de vista empresarial.

Entrando en detalle, las plataformas IoT se consideran el componente más importante del ecosistema IoT. El IoT no puede funcionar sin este software middleware. Las plataformas llenan el vacío entre los sensores de los dispositivos y las redes de datos. En primer lugar, conectan los datos con el sistema de sensores. En segundo lugar, proporcionan información mediante aplicaciones back-end para dar sentido a la gran cantidad de datos ofrecidos por los numerosos sensores. Desde un punto de vista funcional, una plataforma IoT puede supervisar, gestionar y controlar varios tipos de puntos finales y, además, puede permitir la conectividad y la gestión de la red, la gestión, el procesamiento y el análisis de datos, el desarrollo de aplicaciones, la seguridad, el control de acceso, la supervisión, el procesamiento de eventos y la interfaz e integración.

Desde un punto de vista conceptual la definición más extendida por la IoT-EPI [26] es la siguiente: *Una plataforma de IoT se puede definir como una capa inteligente que conecta las cosas a la red y que abstrae las aplicaciones de las cosas con el objetivo de permitir el desarrollo de servicios. Las plataformas de IoT logran varios objetivos principales, como la flexibilidad (poder implementar cosas en diferentes contextos), la usabilidad (poder hacer que la experiencia del usuario sea fácil) y la productividad (permitiendo la creación de servicios para mejorar la eficiencia, pero también permitiendo el desarrollo de nuevos servicios). Una plataforma de IoT facilita la comunicación, el flujo de datos, la gestión de dispositivos y la funcionalidad de las aplicaciones. El objetivo es crear aplicaciones de IoT dentro de un marco de plataforma de IoT. Una*

plataforma de IoT permite que las aplicaciones conecten máquinas, dispositivos, aplicaciones y personas a centros de control y datos.

Las funcionalidades de las plataformas de IoT cubren la cadena de valor digital como sensores, actuadores, dispositivos hardware, elementos tecnológicos para garantizar la conectividad, despliegues en la nube y aplicaciones:

- Las plataformas de conectividad de hardware se utilizan para conectar los dispositivos periféricos y procesar los datos fuera del centro de datos, y programar los dispositivos para tomar decisiones sobre la marcha. Los beneficios son la seguridad, la interoperabilidad, la escalabilidad y la capacidad de gestión mediante el uso de análisis y gestión de datos avanzados desde el sensor hasta el centro de datos.
- Las plataformas software IoT incluyen la integración de sensores y actuadores heterogéneos, varios protocolos de comunicación abstraen todas esas complejidades y presentan a los desarrolladores APIs simples para comunicarse con cualquier sensor en cualquier red.
- Las plataformas IoT también ayudan con la ingesta, el almacenamiento y el análisis de datos, por lo que los desarrolladores pueden centrarse en la creación de aplicaciones y servicios, que es donde reside el valor real de IoT.
- Los proveedores de la nube ofrecen plataformas de IoT basadas en la nube para ayudar a los desarrolladores a crear soluciones de IoT en sus nubes. Los proveedores de infraestructura como servicio (IaaS) y los proveedores de plataforma como servicio (PaaS) tienen soluciones para desarrolladores de IoT que cubren diferentes áreas de aplicación.
- Las soluciones PaaS, abstraen la red subyacente, la computación y la infraestructura de almacenamiento, se enfocan en la funcionalidad móvil y de Big Data, mientras se mueven a dispositivos periféricos abstractos y agregan funciones para la ingesta y procesamiento de datos y servicios de análisis.

Otra aproximación es categorizar las plataformas según su naturaleza comercial:

- Plataformas Open Source
- Plataformas comerciales
    - Desarrolladas por pequeñas y medianas empresas y por start-ups.

- Desarrolladas por multinacionales: Empresas con experiencia que están centradas en ofrecer varios tipos de soluciones mediante plataformas cloud. Centradas en cubrir las necesidades de la industria, en facilitar las comunicaciones o en el acceso a los dispositivos.
- Plataformas globales de los líderes de la industria: Como son Microsoft, Amazon, Google o IBM.

A continuación, se resumen brevemente un conjunto de las plataformas que cumplen con estas funcionalidades de interoperabilidad [91][92]:

- Plataformas IoT centradas en la nube: Servicio totalmente gestionado e integrado en la oferta de la nube. Este tipo de plataformas permiten comunicaciones bidireccionales fiables y seguras entre millones de dispositivos IoT y una solución back-end. Algunas plataformas son Microsoft Azure IoT, la plataforma Amazon AWS IoT y la plataforma IBM Watson IoT.
- Plataformas de IoT centradas en la industria: La conectividad de IoT se extiende a máquinas, sensores, dispositivos y procesos en los sectores industriales. Los resultados empresariales producen un aumento de la eficiencia de la fabricación, una mejor utilización de los recursos y una transformación de los modelos de soporte que están impulsando su adopción. En este contexto, el desarrollo de las plataformas de IoT industrial está impulsado por las grandes empresas de fabricación. Algunas plataformas son PTC ThingWorx, Bosch IoT Platform y GE Predix.
- Plataformas de IoT centradas en la comunicación: Se centran en la gestión de productos y máquinas conectadas y en la implementación de aplicaciones IoT y M2M. Algunas plataformas son Cisco/Jasper y Axeda IoT.
- Centradas en el dispositivo: Se desarrollan como plataformas de software específicas de hardware impulsadas por empresas que comercializan componentes de dispositivos IoT y han construido un backend de software que se denomina plataforma IoT. Algunas plataformas son Open Hab, Nimbits, IoT ToolKit y Chimera IoT platform.
- Plataformas de pymes o startups: Algunas plataformas de este tipo son Xively, Thingspeak o Carriots.
- Plataformas de proyectos código abierto: Algunas plataformas de este tipo son Fiware, OpeIoT, Universaal y Kaa.

- Plataformas basadas en OneM2M (ETSI): Son implementaciones de plataformas M2M que siguen el estándar OneM2M, un estándar relacionado con el IoT cada vez más importante en el sector de las telecomunicaciones. Algunas plataformas son Eclipse OM2M y Open MTC.

Se procede a continuación, a listar detalladamente las principales plataformas involucradas en la presente Tesis Doctoral:

- **FIWARE** [93]

  Descripción: La idea detrás de FIWARE es que la tecnología debe ser accesible para todos, con un enfoque en la interoperabilidad, la modularidad y la personalización. Los datos de diferentes fuentes relevantes deben integrarse sin problemas. Por esa razón, los componentes de FIWARE brindan formatos de datos estandarizados abiertos y especificaciones API para simplificar esta integración y facilitar el desarrollo de soluciones inteligentes. La plataforma es de código abierto y los socios del ecosistema pueden integrarla fácilmente en el diseño de sus soluciones para reducir los riesgos de bloqueo de proveedores. Su componente llamado Orion Context Broker es el núcleo de la plataforma digital que permite la integración de datos recopilados, incluyendo conocimientos para una mayor explotación. Es importante tener en cuenta que, actualmente, Orion Context Broker es uno de los componentes básicos del catálogo digital Connecting Europe Facility (CEF). Los componentes básicos de CEF proporcionan capacidades básicas que se pueden reutilizar en cualquier proyecto europeo para garantizar la interoperabilidad entre los sistemas de las tecnologías de la información y facilitar la implementación de servicios a través de fronteras y sectores. La API estandarizada significa que los servicios pueden operar en diferentes plataformas de proveedores. FIWARE NGSI (Interfaz de Servicio de Próxima Generación) es la API expuesta por Orion Context Broker y se utiliza para la integración de componentes dentro de una plataforma compatible con FIWARE, y por aplicaciones para actualizar o consumir información de contexto. Dicha API define un modelo de datos, una interfaz para intercambiar información y una interfaz de consultas para obtener la información contextual. Finalmente, están los agentes que son los conectores que garantizan la transmisión de datos sin procesar a Orion Context Broker utilizando sus propios protocolos nativos. La plataforma FIWARE aparece en todos los casos de uso de la presente Tesis Doctoral y es clave en la implementación de los mecanismos de interoperabilidad middleware.

  Detalles: El Catálogo FIWARE es un conjunto de componentes de plataforma de código abierto que se pueden ensamblar junto con otros compo-

nentes de plataformas de terceros para acelerar el desarrollo de Soluciones Inteligentes. Sobre la base de Orion Context Broker, se encuentra disponible un amplio conjunto de habilitadores genéricos FIWARE de código abierto complementarios, que se ocupan de: (i) Proporcionar interfaz con IoT, robots y sistemas de terceros, para capturar actualizaciones sobre información de contexto y traducir las actuaciones requeridas. (ii) La gestión, publicación y monetización de datos de contexto/API, lo que brinda soporte para el control de uso y la oportunidad de publicar y monetizar parte de los datos de contexto gestionados. (iii) EL procesamiento, análisis y visualización de información de contexto implementando el comportamiento inteligente esperado de las aplicaciones y/o ayudando a los usuarios finales a tomar decisiones inteligentes. (iv) Impulsar la iniciativa Smart Data Models centrada en facilitar un enfoque basado en el ecosistema y comunidad de FIWARE para definir modelos de datos comunes a las áreas de interés. (v) Proporcionar tutoriales, cursos y demostraciones guiadas de los componentes paso a paso.

- **UniversAAL** [94]

  Descripción: UniversAAL es una plataforma de software distribuida y semántica diseñada para facilitar el desarrollo y la interoperabilidad de aplicaciones integradas de Ambient Assisted Living (AAL). Se basa en el proyecto PERSONA (Sexto Programa Marco Europeo FP6) y explota en gran medida la semántica para apoyar la interoperabilidad entre dispositivos, servicios y aplicaciones, en una escala sin precedentes. La naturaleza semántica de UniversAAL lo hace ideal para entornos altamente heterogéneos, que van desde IoT hasta dispositivos portátiles, Big Data y muchos más dominios. En UniversAAL se modelan semánticamente todos y cada uno de los componentes, además del mundo real o los servicios, que también se representan como ontologías. La mayoría de las ontologías se han construido completamente desde cero, ya que fueron discutidas y acordadas entre los socios del proyecto UniversAAL, pero no se basan en ningún estándar existente.

  Detalles: UniversAAL se ha desarrollado en el contexto de proyectos europeos sobre el uso de las tecnologias de la información para envejecer bien, como una plataforma para Ambient Assisted Living (AAL). Por lo tanto, la mayoría de las aplicaciones desarrolladas hasta ahora sobre universAAL tienen como objetivo apoyar la vida activa e independiente de las personas mayores. Sin embargo, universAAL apoya el desarrollo de otros tipos de aplicaciones más allá de AAL. La parte inferior de universAAL, conocida como middleware universAAL, es un facilitador para la integración de componentes distribuidos y la comunicación entre ellos,

ocultando la distribución y la heterogeneidad. Los casos de uso soportados por el middleware de universAAL son principalmente de envío y recepción de mensajes, permitiendo la interoperabilidad semántica sin ningún sesgo de dominio específico. Esta característica hace que el middleware universAAL sea apropiado para la integración de cualquier sistema distribuido abierto de sistemas. Más allá del middleware, hay un conjunto de componentes opcionales de gestión de universAAL. Estos se basan en la confianza en las tecnologías de la web semántica para la representación unificada de los datos independientemente del dominio y la tecnología de extracción, el lenguaje de consulta unificado entre dominios y los modelos de dominio externalizables y compartibles. UniversAAL también ofrece un marco de interacción con el usuario para abordar los principales retos de interacción. El intermediario encargado de dicha interacción es el bus de interfaz de usuario que, como el resto de componentes, está muy orientado a la semántica.

- **SOFIA2/ONESAIT** [95]

  Descripción: ONESAIT, antes SOFIA2, es una plataforma de IoT basada en la nube y con una aproximación semántica, que utiliza modelos ontológicos para la conexión entre la información del dispositivo de bajo nivel y las aplicaciones. La plataforma ONESAIT proporciona la flexibilidad necesaria para que los desarrolladores puedan construir sus propias soluciones de forma sólida y ágil utilizando tecnologías Open-Source. Proporciona una visión unificada de las entidades de negocio. El modelo describe el significado de las entidades, las relaciones y los datos.

  Detalles: ONESAIT Platform Community edition es la versión gratuita de la plataforma digital y de código abierto que cualquiera puede descargar y utilizar para crear soluciones completas. ONESAIT Platform proporciona compatibilidad nativa para aplicaciones desarrolladas sobre la plataforma FIWARE, permitiendo que las aplicaciones que utilizan el estándar NGSI se puedan conectar para interoperar entre sí, y además poder utilizar el resto de capacidades de la plataforma. El soporte nativo para el protocolo NGSI se consigue incorporando a la arquitectura de la ONESAIT Platform tres componentes de la arquitectura abierta FIWARE: el Orion Context Broker, el encargado de persistencia Cygnus y STH-Comet para almacenar información a corto plazo. Onesait es una plataforma tecnológica que acelera la transformación digital, permitiendo la construcción ágil de aplicaciones mediante el uso de componentes integrados de código abierto y la combinación de tecnologías centradas en el tratamiento, análisis inteligente y publicación de información. Cuenta con un amplio catálogo de componentes que permiten la construcción de

aplicaciones de gestión clásicas, así como sistemas avanzados que aplican inteligencia a los procesos, conectan el mundo físico con el virtual, o escalan inteligentemente para adaptarse a las necesidades del negocio.

- **OPENIOT** [96]

  Descripción: OpenIoT es una plataforma financiada por la UE que respalda el desarrollo de servicios de IoT en la nube en contextos donde la interoperabilidad semántica es un requisito importante. Para recopilar datos de múltiples dispositivos de detección, OpenIoT amplía Global SensorNetworks (GSN) con funciones RDF y datos vinculados, y proporciona un envoltorio GSN-CoAP para acceder a los datos del sensor mediante un enfoque RESTful. Para unificar semánticamente y componer dinámicamente y bajo demanda servicios de IoT heterogéneos siguiendo un paradigma basado en la nube. Open-IoT aprovecha tres componentes principales: la ontología W3C Semantic SensorNetworks (que proporciona un modelo común basado en estándares para representar sensores virtuales), un middleware de código abierto (que facilita la recopilación y la anotación semántica adecuada de datos sensoriales desde prácticamente cualquier dispositivo IoT) y un conjunto de herramientas visuales que fomentan el fácil desarrollo e implementación de aplicaciones IoT. Las sinergias con otros proyectos de código abierto como GSN hacen que OpenIoT actualmente esté siendo respaldado por una comunidad de investigadores de IoT.

  Detalles: OpenIoT permite la integración de servicios de IoT dispersos geográfica y administrativamente de forma semánticamente interoperable. OpenIoT también proporciona algunas herramientas visuales para construir aplicaciones IoT utilizando una interfaz de usuario Lo que Ves Es lo que Obtienes (WYSIWYG). Puede gestionar la conexión de sensores móviles a través de un servicio pub/subscribe. OpenIoT es pertinente para una amplia gama de áreas científicas y tecnológicas interrelacionadas: Redes de sensores, middleware para sensores, ontologías, modelos semánticos, anotaciones semánticas, representación de objetos conectados a Internet, técnicas de datos abiertos semánticos, computación de servicios en la nube, seguridad basada en servicios y esquemas de privacidad. OpenIoT permite la integración de servicios IoT dispersos geográfica y administrativamente de forma semánticamente interoperable. OpenIoT también proporciona algunas herramientas visuales para la construcción de aplicaciones IoT utilizando una interfaz de usuario WYSIWYG. Puede gestionar la conexión de sensores móviles a través de un servicio pub/subscribe. OpenIoT es pertinente para una amplia gama de áreas científicas y tecnológicas interrelacionadas: Redes de sensores, middlewa-

re para sensores, ontologías, modelos semánticos, anotaciones semánticas, representación de objetos conectados a Internet, técnicas de datos abiertos semánticos, computación de servicios en la nube, seguridad basada en servicios y esquemas de privacidad.

- **SENSINACT** [97]

  Descripción: Es una plataforma horizontal dedicada al IoT y utilizada especialmente en diversas aplicaciones de ciudades y hogares inteligentes. SENSINACT tiene como objetivo gestionar la heterogeneidad de los protocolos y dispositivos del IoT y proporciona acceso síncrono (bajo demanda) y asíncrono (periódico o basado en eventos) a los datos/acciones de los dispositivos del IoT, así como acceso a los datos históricos con una API genérica y fácil de usar. SENSINACT viene con dos marcos complementarios: (i) La plataforma SENSINACT interconecta los dispositivos IoT utilizando diferentes protocolos como Zigbee, EnOcean, LoRa, XBee, MQTT, XMPP, así como plataformas como FIWARE y permite acceder a ellos con varios protocolos como HTTP REST, MQTT, XMPP, JSON RPC y CDMI. La pasarela también puede alojar aplicaciones y gestionarlas mediante un módulo de gestión de aplicaciones. (ii) SENSINACT Studio ofrece un IDE (Entorno de Desarrollo Integrado) basado en Eclipse para gestionar los dispositivos existentes, además de desarrollar, desplegar y gestionar aplicaciones IoT.

  Detalles: La plataforma SENSINACT está compuesta por seis grupos funcionales y sus correspondientes interfaces: (i) El Adaptador de Protocolo del Dispositivo abstrae la tecnología de conectividad específica de las redes de sensores inalámbricos. Está compuesto por los puentes asociados a las pilas de protocolos. Todos los puentes se ajustan a una API genérica de acceso a dispositivos que se utiliza para interactuar con los servicios que ofrece SENSINACT. (ii) El Acceso y Control de Objetos Inteligentes implementa las funcionalidades principales de SENSINACT, tales como el descubrimiento de los dispositivos y los recursos, la seguridad de la comunicación entre los dispositivos y los consumidores de sus servicios, etc. (iii) La API del Consumidor es agnóstica al protocolo y expone los servicios del Acceso y Control de Objetos Inteligentes a los Consumidores. (iv) El Adaptador de Protocolo del Consumidor consiste en un conjunto de puentes de protocolo que traducen la interfaz de la API del Consumidor a protocolos de aplicación específicos. (v) El grupo funcional de Gestión de Pasarelas incluye todos los componentes necesarios para facilitar la gestión de los dispositivos conectados a SENSINACT, independientemente de sus tecnologías subyacentes. Para ello se utiliza una API de gestión de dispositivos. Este grupo funcional también contiene los componentes

que gestionan la caché, el directorio de recursos y los servicios de seguridad. Estas funciones de gestión se exponen mediante la API de gestión de pasarelas. (vi) Y, por último, el Adaptador de Protocolos del Gestor permite adaptar la API de Gestión de Pasarelas a los protocolos específicos utilizados por diferentes entidades de gestión externas.

- **WSO2** [98]

  Descripción: WSO2 es un middleware de arquitectura orientada a servicios (SOA) de código abierto. Está diseñado con componentes independientes, por lo que se puede adaptar para una solución dirigida a las aplicaciones empresariales. Toda la pila de middleware de WSO2 funciona en nubes privadas o públicas, administradas por WSO2 e híbridas, así como en las propias empresas.

  Detalles: WSO2 ofrece capacidades de administración de dispositivos de Internet de las cosas, administración de API, integración, análisis, administración de identidad y seguridad, desarrollo de servicios y aplicaciones, administración, gobernanza y aplicaciones móviles.

- **Plataformas basadas en el estándar OneM2M** [99]

  Descripción: OneM2M no es una plataforma, sino una comunidad global que desarrolla estandards para IoT. El objetivo de oneM2M es desarrollar especificaciones técnicas que aborden la necesidad de una capa de servicio M2M común que pueda integrarse fácilmente dentro de varios hardwares y softwares, y en la que se pueda confiar para conectar la gran cantidad de dispositivos existentes con servidores de aplicaciones M2M en todo el mundo.

  Detalles: Hay diferentes implementaciones de código abierto de oneM2M como Eclipse OM2M project o OpenMTC. (i) El proyecto Eclipse OM2M, es una implementación de código abierto del estándar oneM2M y SmartM2M. Proporciona una plataforma de servicios M2M horizontales para el desarrollo de servicios independientemente de la red subyacente, con el objetivo de facilitar el despliegue de aplicaciones verticales y dispositivos heterogéneos. OM2M ofrece una plataforma flexible basada en oneM2M para implementar servidores, pasarelas y dispositivos M2M horizontales. Presenta una arquitectura modular, que se ejecuta sobre un contenedor OSGi, que es altamente extensible a través de plugins. (ii) OpenMTC es un middleware de integración basado en el estándar oneM2M, para realizar investigación aplicada y desarrollar aplicaciones M2M e IoT innovadoras. Su enfoque de servicio horizontal integra fácilmente dispositivos de diferentes verticales de IoT industrial, independientemente del hardware subyacente o de la infraestructura de red. Se trata

de una iniciativa conjunta del Instituto Fraunhofer de Sistemas de Comunicación Abiertos y la Cátedra de Redes de Nueva Generación del Departamento de Sistemas de Telecomunicaciones de la Universidad Técnica de Berlín.

A continuación, se listan y describen las principales plataformas Cloud del mercado y potencialmente utiles para la presente Tesis Doctoral:

IBM Cloud/Bluemix [100]: IBM Cloud, anteriormente Bluemix, es una Plataforma como Servicio (PaaS) para construir, implementar, administrar y ejecutar rápidamente aplicaciones que se componen de forma incremental a partir de servicios. Las tecnologías de inteligencia artificial de IBM fomentan funcionalidades avanzadas como el intercambio de datos en tiempo real, la comunicación segura y los servicios cognitivos para unificar los dispositivos de IoT. IBM Cloud / Bluemix es relativamente fácil de usar y proporciona interfaces intuitivas para que los principiantes desarrollen servicios de IoT. Es compatible con varios lenguajes de programación y servicios, así como con DevOps integrado para crear, ejecutar, implementar y administrar aplicaciones en la nube. Además de los servicios comunes de IoT, IBM Cloud / Bluemix proporciona extensiones para Reglas comerciales, procesamiento Hadoop, capa de base de datos Cloudant y MongoDB NoSQL, diferentes herramientas DevOps, mensajería y acceso a los servicios Watson, particularmente para procesamiento de lenguaje natural.

Microsoft Azure IoT [101]: Microsoft Azure IoT, que forma parte de Azure Cloud Services, es una solución basada en la nube que proporciona un servicio de procesamiento de datos y comunicación de dispositivos IoT. Azure ofrece diferentes servicios (incluidas bases de datos, Hadoop, aprendizaje automático, etc.) pero también proporciona, al ser una solución a escala empresarial, una plataforma en la que se puede implementar una amplia gama de sistemas de IoT, comenzando desde cero o preconstruidos. componentes. Las características específicas de IoT incluyen la recopilación de datos y el análisis de transmisiones. Las transmisiones pueden ser preprocesos utilizando un lenguaje similar a SQL, lo que permite la definición de reglas y se canaliza a otros productos de Azure. El SDK de Azure IoT Gateway implementa el protocolo AllJoyn para Windows y Linux. Admite comunicación a través de HTTP, AMQP y MQTT. La función de la puerta de enlace es enviar o recibir datos desde y hacia Azure, el núcleo de cualquier solución de suite de IoT. En el lado del servidor en la nube, dicha funcionalidad, es decir, la comunicación bidireccional segura, la ofrece el servicio Azure. Azure ha ampliado su API para admitir el estándar abierto Recursos Rápidos de Interoperabilidad en Salud (FHIR). Este modelo de datos estandariza la semántica y el intercambio de datos para que tanto

los sistemas Azure como los sistemas de salud de terceros que utilizan FHIR puedan interoperar.

Google Cloud IoT [102]: Google Cloud IoT es una plataforma que explota los servicios, las tecnologías y la infraestructura de Google. Esto incluye los productos en la nube para computación, almacenamiento, redes, big data, aprendizaje automático, autenticación y seguridad de un extremo a otro, y una red de fibra global que permite entrega de datos de muy baja latencia hacia y desde dispositivos IoT. Cloud IoT Core es un servicio totalmente administrado para conectar, administrar y recoger datos y eventos que se transmiten a gran escala de manera fácil, segura y en tiempo real. Aprovechando Google Cloud Pub / Sub, se proporcionan funcionalidades confiables de procesamiento y consulta de datos de IoT. Una puerta de enlace de IoT gestiona el tráfico entre redes que utilizan diferentes protocolos, y también actúa como un proxy en el caso de dispositivos de IoT con recursos limitados. Otra característica interesante es Cloud Dataflow, un modelo de programación unificado para fuentes de datos por lotes y de transmisión. La seguridad también es una preocupación importante, las API de Google Cloud Platform son seguras de forma predeterminada con cifrado completo, respaldado por una seguridad integrada y generalizada en toda la infraestructura.

AWS IoT [103]: Amazon AWS es una plataforma de IoT genérica provista de un IaaS asociado que proporciona servicios basados en la nube bajo demanda, como computación, almacenamiento, bases de datos, análisis, redes, dispositivos móviles, herramientas de desarrollo, herramientas de administración, IoT, seguridad y aplicaciones empresariales. Compuesto por un conjunto de módulos independientes pero integrables como DynamoDB, AWS Lambda, Kinesis, AWS tiene como objetivo simplificar la gestión de dispositivos IoT, la recopilación y análisis de datos sensoriales en la nube y la interacción con aplicaciones IoT. Estas funcionalidades están aseguradas a través de un conjunto de autenticación y encriptación avanzada y también están disponibles sin conexión a través de las llamadas Device Shadows (básicamente, un documento JSON), que permiten almacenar y recuperar información del estado actual de un dispositivo a través de MQTT o HTTP.

Finalmente, se ofrece también un listado de otras plataformas IoT que no corresponden con plataformas principalmente destacadas en los casos de uso de la presente Tesis Doctoral, pero que, serán mencionadas durante los próximos capítulos:

SEAMS [104]: Como indica su título Smart Energy-efficient and Adaptive Management System (SEAMS) es capaz de monitorizar las máquinas y equipos que se están utilizando en una Terminal Portuaria de Contenedores.

I3WSN [105]: Plataforma para redes de sensores inalámbricos inteligentes industriales para entornos interiores.

BodyCloud [106]: Es una plataforma abierta para la integración de redes de sensores corporales con una infraestructura Cloud Platform-as-a-Service (PaaS) y actualmente se basa en Google App Engine.

KAA [107]: El Proyecto Kaa es una plataforma IoT de middleware multipropósito, de código abierto, para construir soluciones IoT inteligentes, conectadas y de un extremo a otras basadas en la nube al facilitar el intercambio de datos entre los dispositivos IoT, su análisis y visualización. Aprovechando la arquitectura de microservicios, KAA permite la gestión, el mantenimiento, la interacción y la integración, casi en tiempo real, de múltiples dispositivos IoT conectados a plataformas IoT heterogéneas a través de protocolos de código abierto como MQTT y CoAP. Se proporcionan funcionalidades de back-end para operar con soluciones de IoT a gran escala que comprenden la seguridad, la coherencia y la interoperabilidad de los datos. KAA está preintegrado con las soluciones de procesamiento de datos existentes (mongoDB, Hadoop, Oracle, etc.) y ya tiene muchas aplicaciones de consumo (por ejemplo, wearables, atención médica, entretenimiento) para Android, Raspberry Pi y otras plataformas.

Thingspeak [108]: Basado en tecnología de nube pública y canales abiertos, ThingSpeak permite recopilar datos de una variedad de sensores de plataforma (Arduino, Raspberry Pi, Beaglebone, etc.) a través de protocolos populares de IoT. Este agregador de nube proporciona interfaces web para acceder de forma transparente a los datos del sensor de IoT. Una vez almacenados en la nube, los datos pueden analizarse y visualizarse a través de las herramientas de análisis de MATLAB y explotarse a través de la API abierta para interactuar con diferentes aplicaciones de terceros (por ejemplo, Twitter, IFTTT, Twilio). De esta manera, Thingspeak permite la creación de prototipos y la construcción de sistemas IoT interoperables que requieren análisis (descubrir relaciones, patrones y tendencias, por ejemplo) sin configurar servidores o desarrollar software web desde cero.

Eclipse OM2M [109]: Citado anteriormente, el proyecto Eclipse OM2M es una implementación de código abierto del estándar oneM2M y SmartM2M.

AllJoyn [110]: Marco de software de código abierto que facilita a los desarrolladores escribir aplicaciones que puedan descubrir dispositivos cercanos y comunicarse entre sí, directamente independientemente de las marcas, categorías, transportes y sistemas operativos sin la necesidad de la nube. Extrae los detalles de los transportes físicos y proporciona una API fácil de usar. Se admiten múltiples topologías de sesión de conexión, incluidas las sesiones de punto a punto y de grupo. El marco de seguridad es flexible y admite muchos mecanismos y modelos de confianza. Los tipos de datos transferidos también son flexibles y admiten sockets sin procesar u objetos abstractos con interfaces, métodos, propiedades y señales bien definidos.

OpenHab [111]: OpenHAB no intenta reemplazar las soluciones existentes, por lo que puede considerarse como un sistema de sistemas. Por lo tanto, asume que los subsistemas se instalan y configuran independientemente de openHAB, ya que a menudo se trata de un asunto muy específico y complejo. En cambio, openHAB se centra en el lado del uso diario de las cosas y se abstrae de los propios dispositivos. Un concepto central de openHAB es la noción de elemento. Un elemento es un bloque de construcción atómico funcional centrado en datos, se puede pensar en él como una capacidad. OpenHAB no esta centrado en si un elemento está relacionado con un dispositivo físico o alguna fuente virtual como un servicio web o un resultado de cálculo. Todas las funciones que ofrece openHAB utilizan la abstracción de los elementos. El usuario no encontrará ninguna referencia a elementos específicos del dispositivo (como direcciones IP, ID, etc.) en las reglas de automatización, definiciones de UI, etc. Esto hace que sea perfectamente fácil reemplazar una tecnología por otra sin realizar ningún cambio en las reglas y las interfaces de usuario.

IoTivity [112]: IoTivity es una plataforma de código abierto que implementa los estándares OIC, los principios CoAP y REST para garantizar la conectividad de dispositivo a dispositivo en múltiples dominios como domótica, automotriz, atención médica y promoción de la interoperabilidad de dispositivos y aplicaciones en mercados y casos de uso. Este marco software funciona como middleware en todos los sistemas operativos y plataformas de conectividad, y admite operaciones como el descubrimiento de dispositivos y recursos, la transmisión de datos, la gestión de dispositivos y la gestión de datos. Basada en el marco de IoTivity, la plataforma de código abierto IoTivity Cloud se ha desarrollado para recopilar, analizar e interpretar en la nube una gran cantidad de datos provenientes del entorno de IoT que son fundamentales para desarrollar servicios avanzados de IoT.

ThingWorx [113]: Esta solución proporciona interfaces fáciles de usar a través de las cuales los usuarios pueden construir aplicaciones en poco tiempo,

con costes y riesgos reducidos. Las áreas industriales que utilizan ThingWorx incluyen agricultura, ciudades inteligentes, redes eléctricas, gestión del agua, eficiencia de edificios y telemática. A través de un componente de modelado de aplicaciones dedicado, llamado Composer, la plataforma ThingWorx IoT permite a los usuarios construir interfaces móviles funcionales sin necesidad de codificación, mediante el uso de paneles y espacios de trabajo en tiempo real. La plataforma permite la integración con una variedad de otras tecnologías, desde la realidad aumentada hasta la conectividad industrial, facilitando así su adopción.

2.5.3. Aproximación en capa de Aplicaciones y Servicios

Los servicios se encuentran en la parte superior de la pila de elementos de las plataformas de IoT. En muchos casos, las plataformas de IoT ofrecen habilitadores o herramientas que facilitan la interconexión entre sus propios componentes y aplicaciones. Las plataformas IoT proporcionan una arquitectura o pila de acciones en la que sus componentes interactúan con otros siguiendo las reglas internas definidas por la propia plataforma, como el uso de un formato específico en el intercambio de información. Usando este enfoque, es difícil para los desarrolladores crear nuevas interconexiones entre servicios, diferentes de las que ya existen. Como solución, algunas plataformas ofrecen Marketplace [114] o frameworks para interactuar con sus propios servicios, dando más posibilidades al desarrollador [92]. Estos mercados y marcos suelen estar estrechamente vinculados al formato de datos internos y al intermediario de sus plataformas. En estos casos, es necesario trabajar dentro de las reglas definidas por la plataforma IoT. Este enfoque general dificulta las posibilidades de interactuar con los servicios prestados por las otras plataformas.

Además, los principales problemas genéricos comunes para lograr la interoperabilidad entre servicios también se replicarán en un dominio enfocado en IoT. Existen múltiples tipos de servicios (procesador de eventos complejos, base de datos histórica, procesamiento de Big Data, visualización, análisis, etc.). Los servicios son heterogéneos y es difícil interconectar sus funcionalidades e información. Además, están muy vinculados al dominio de la solución.

Las plataformas de IoT no tienen la capacidad de interactuar con otras plataformas de IoT en la capa de aplicaciones y servicios [115]. Las nuevas soluciones de interoperabilidad se centran en resolver los problemas de interoperabilidad entre los dispositivos de diferentes plataformas, pero no en la interoperabilidad de los servicios de IoT. Es necesario desarrollar nuevas soluciones para este problema de falta de interoperabilidad. Otro aspecto a destacar es la falta de catálogos de aplicaciones IoT [17]. El catálogo facilita el descu-

brimiento de propiedades y funcionalidades sobre los servicios de plataforma disponibles [116].

La aproximación que sigue esta Tesis Doctoral en la capa de aplicación y servicios es ofrecer una capa de abstracción completamente funcional que garantice la interoperabilidad entre los servicios y aplicaciones que están disponibles en las plataformas IoT. Además de ofrecer una metodología para realizar conectores para acceder a los servicios de la plataforma IoT e implementar una arquitectura completa para interactuar con estos servicios y crear servicios compuestos.

Se han analizado diferentes enfoques que son útiles para lograr la interoperabilidad entre servicios y aplicaciones:

- La composición del servicio es el elemento principal que respalda este objetivo. La composición de servicios engloba todos aquellos procesos que crean servicios de valor agregado, llamados servicios compuestos, a partir de servicios existentes en las plataformas de IoT. Se han analizado en la bibliografía [117][118][119][120] métodos como la composición de servicios, orquestación, coreografía, mashup o programación orientada a flujos.

- Se debe considerar el acceso nativo como un método para acceder a las aplicaciones y servicios de las plataformas de IoT. Casi todas las plataformas de IoT proporcionan una API pública para acceder a sus servicios, aunque en algunas plataformas hay que manejar otras alternativas para acceder a ellos

- Faltan catálogos de aplicaciones, datos y dispositivos dedicados al IoT. Con una solución basada en un Catálogo de Servicios, se pueden registrar aplicaciones para hacerlas detectables. Además, se ofrece una descripción o información detallada sobre los servicios y aplicaciones. La utilización de descubrimiento de servicios permitirá descubrir información sobre los servicios de IoT. Se han analizado soluciones como UDDI [121], WSDL [122], HyperCat [123] o iServe [124]. Estas no han sido utilizadas, pero han ofrecido conocimientos de cómo desarrollar y trabajar con soluciones de este tipo.

- Es deseable considerar las posibilidades que brinda la virtualización de servicios y aplicaciones. Esto será desarrollado en la sección de los componentes habilitantes en la parte de la virtualización mediante contenedores, destacando el uso de Docker.

- Las ventajas que los envoltorios pueden aportar al IoT. El término envoltorio significa aquellos programas específicos capaces de extraer datos

de sitios o servicios de Internet y convertir la información en un formato estructurado.

Desde el punto de vista de la composición de servicios, las plataformas IoT existentes implementan este tipo de herramientas para componer servicios. Son dos ejemplos destacados:

- FIWARE: WireCloud [125] es un habilitador de Fiware basado en mashup de aplicaciones web centrada en el usuario destinada a aprovechar los servicios de IoT de la plataforma. Los mashups de aplicaciones web integran datos heterogéneos, lógica de aplicación y componentes gráficos (widgets) que se obtienen de la web para crear nuevas aplicaciones compuestas funcionales y de valor agregado. Esta herramienta aparecerá en la validación del proyecto relacionado con la validación del caso de uso DataPorts en el Capítulo 9.

- OpenIoT [126]: Esta plataforma proporciona OpenIoT IDE, un entorno de desarrollo integrado. Usado para crear, implementar y gestionar aplicaciones de IoT. Open-IoT IDE comprende una gama de herramientas visuales que permiten: (a) Definición visual de los servicios de IoT de una manera que evita la necesidad de dominar los detalles del lenguaje SPARQL; (b) Descubrimiento visual de sensores de acuerdo con su ubicación y tipo; (c) Configuración de los metadatos del sensor según sea necesario para su integración dentro del software intermedio; (d) Monitorización del estado de los diversos servicios de IoT, incluidos los volúmenes de datos que producen y el estado de los sensores que los componen; (e) Visualización de servicios de IoT sobre la base de mashups Web2.0, es decir, en mapas, gráficos de líneas, barras, paneles de control y otros elementos visuales. Estas herramientas aceleran el proceso de desarrollo de aplicaciones de IoT. En varios casos, se pueden desarrollar aplicaciones sencillas sin apenas programar.

El paradigma de composición de servicios seleccionado para implementar la composición de servicios en esta Tesis Doctoral es la aproximación llamada Programación Basada en Flujos [127]. Se extenderá en los Capítulos 3 y 4 de la presente Tesis Doctoral.

La programación basada en flujos es un paradigma que define las aplicaciones como procesos de caja negra, llamados nodos. Estos procesos intercambian datos a través de conexiones predefinidas entre los procesos mediante el paso de mensajes. Estos procesos de caja negra se pueden conectar entre sí para crear un flujo que implemente diferentes aplicaciones compuestas sin necesidad de modificar los nodos internamente [128].

Las plataformas de IoT proporcionan una interfaz para acceder a sus servicios. Principalmente, ofrecen una API REST u otras alternativas como SOAP Web Services, bibliotecas de programación o mediante envoltorios que proporcionan acceso a la información. Por esa razón, las herramientas que implementan la programación basada en flujo tienen que desarrollar piezas de código para envolver y acceder a las funcionalidades de estos servicios. Esta tarea se realiza aprovechando las interfaces disponibles. Los fragmentos de código para acceder a los servicios son los denominados nodos. Estos nodos proporcionan el mecanismo para acceder e interactuar con el servicio de IoT [117].

Un nodo necesita parámetros de entrada para ejecutarse y tras su ejecución ofrece una información de salida. Durante esta ejecución tienen lugar una serie de procesos internos para procesar la información y proporcionar los resultados. Estos procesos son iniciados por la llamada de otro nodo o de un usuario. La interacción entre los diferentes nodos estará definida por un flujo de ejecución. Este flujo de ejecución es el que define, gestiona y actúa como motor en este proceso de interoperabilidad entre los servicios.

Este paradigma de programación se está implementando mediante soluciones centradas en crear nuevas aplicaciones basadas en la interconexión de diferentes servicios, como las siguientes:

Organización	Descripción
Node-RED [129]	La herramienta basada en flujo más extendida relacionada con IoT. Una solución de código abierto creada por IBM. Es fácil de desplegar, usar y extender. Se pueden crear aplicaciones compuestas con los servicios disponibles de manera muy rápida y efectiva. La única desventaja es que algunas empresas, dado lo aparentemente sencillo que es su usabilidad, la consideran una herramienta más centrada a crear prototipos o la implementación de soluciones sencillas. Han tenido reticencia en usarla en aplicaciones de entornos de producción. No obstante, esta solución comenzó hace aproximadamente 10 años, va por la segunda versión completa y con una correcta configuración puede ser integrada en ecosistemas industriales sin ningún problema. Hay una gran cantidad de nodos y flujos disponibles para integrarse con tecnologías de la información de todos los ámbitos.

Apache Nifi [130]	Apache NiFi admite grafos dirigidos potentes y escalables de encaminamiento de datos, transformación y lógica de mediación del sistema. Ofrece una interfaz gráfica y un potente sistema para procesar y distribuir datos. Forma parte de un habilitador de Fiware llamado Draco que garantiza la persistencia y conexión de la información contextual obtenida. Esto indica que es una solución mantenida y utilizada por la comunidad. Aunque NiFi no es una herramienta complicada, otras herramientas como Node-RED, tienen la capacidad de brindar una solución más sencilla y en la que los primeros resultados se obtienen más rápidamente.
IFTTT [131]	Una herramienta que fue muy popular hace unos diez años es IFTTT. Crea flujos basados en el concepto *Si Esto Entonces Esto*. Su principal objetivo es crear un repositorio de tareas automáticas basado en este principio. Está enfocado a la interconexión de Servicios Web y la creación de nuevas tareas que no sean excesivamente complejas. El concepto que propone es interesante, pero queda lejos de cubrir los requisitos definidos en la presente Tesis Doctoral.

Flogo [132]	Project Flogo es un ecosistema de código abierto ultraligero basado en Go para crear aplicaciones impulsadas por eventos. Las aplicaciones están compuestas por activadores y acciones o actividades. Los disparadores reciben datos de fuentes externas, son gestionados por un modelo de subprocesamiento configurable y tienen una interfaz común que permite a cualquiera crear un disparador Flogo. Los manejadores envían eventos a las acciones. Las acciones procesan los eventos de manera adecuada con la implementación y tienen una interfaz común que permite las capacidades de procesamiento de eventos. Destaca por ser un software más ligero que las propuestas ofrecidas en Java o Node.js. Ofrece una interesante alternativa al diseño propuesto por Node-RED. No obstante, no ofrece un producto tan asentado en el mercado, ni tantos ejemplos, ni tanta documentación, ni una comunidad de programadores entorno a él tan amplia como la de Node-RED. Además, actualmente es una solución hecha por desarrolladores para desarrolladores, lo que hace que tenga una alta curva de aprendizaje.
Noflo [133]	NoFlo es una implementación en JavaScript de la programación basada en flujo. El flujo de control del software es separado de la lógica real del software, ayudándole a organizar grandes aplicaciones más fácilmente que los paradigmas orientados a objetos tradicionales, especialmente al importar y modificar grandes conjuntos de datos. NoFlo es más difícil de usar que Node-RED. La herramienta está dividida en varios componentes, lo que dificulta los primeros pasos, y el entorno gráfico no es tan amigable. No hay suficientes ejemplos, documentación e implementación disponibles sobre las soluciones de interoperabilidad de IoT.

Tabla 2.3: Soluciones software de programación basada en flujos

El enfoque seleccionado en el Capítulo 4, relacionado con la implementación de los mecanismos de interoperabilidad en la capa de aplicaciones y servicios, utiliza la solución tecnológica Node-RED, proporcionando más detalles sobre la tecnología que los presentados en la tabla. Hay que indicar que Node-RED

permite conectar servicios en línea, API, funciones de código y dispositivos de hardware relacionados con IoT. Las funcionalidades de estos elementos se definen como nodos y esta herramienta describe un gráfico de nodos, que intercambian mensajes que contienen datos a través de los bordes. Los nodos se utilizan principalmente para los siguientes propósitos: procesar eventos entrantes, enviar eventos salientes, manipular los mensajes y las cargas útiles o crear el código que adapta la información de datos de salida de un servicio a la entrada de otro servicio. Es importante resaltar la posibilidad de crear nodos que implementen funciones para actuar como nodos conversores que se colocan entre los servicios, para realizar las conversiones necesarias para comunicar sus salidas con sus entradas[134].

Esta herramienta ofrece un enfoque de programación visual para facilitar la creación de nuevos servicios compuestos. Nodo-RED ofrece el pegamento de backend entre las aplicaciones y los servicios. Los nodos se definen en archivos HTML y JavaScript y los flujos se representan en la notación de objetos JavaScript. Los flujos y nodos creados se pueden compartir fácilmente en línea para usarse en otros proyectos. Además, ya existe una comunidad activa que produce nuevos nodos con regularidad y la plataforma Node-RED es un proyecto de código abierto alojado en GitHub. Esta herramienta es sencilla de ampliar, se pueden agregar nuevas capacidades por encima de ella [135].

Hay varios proyectos que han introducido la herramienta Node-RED y este enfoque en su plataforma, aunque ninguno de ellos está enfocado a la interoperabilidad de servicios y aplicaciones de distintas plataformas IoT. A pesar de ello, todos ofrecen un punto de vista interesante en el uso de estas herramientas. Por ejemplo, el proyecto FRED ofrece la posibilidad de crear múltiples instancias de Node-RED, la gestión de usuarios que utilizan la herramienta y el uso de Docker para incrustar las soluciones en la nube [136]. También proporciona una gran cantidad de manuales sobre cómo crear soluciones utilizando la herramienta. La plataforma Sofia2 (actualmente llamada ONESAIT) presenta esta solución para crear interoperabilidad gráfica entre dispositivos y servicios de su plataforma, para crear flujos de interacción que se pueden mover e implementar en varias Raspberry Pi. El proyecto Sofia2 ha desarrollado nodos para sus propios componentes. El proyecto Glue.Things utiliza esta herramienta para desarrollar una solución software mashup para agregar, manipular y gestionar la entrada de datos de los dispositivos conectados, donde todos los dispositivos registrados se administran en un contenedor de dispositivos virtualizados [137].

Finalmente, es interesante indicar que Apache Flume [138] es una tecnología utilizada en el caso de uso DataPorts presentada en el Capítulo 8. Se hace uso de ella para garantizar la persistencia de los datos recogidos por la plataforma FIWARE y sus agentes con los componentes Big Data desarrollados en ese

proyecto principalmente por sus capacidades de distribuir y procesar datos de manera efectiva.

2.6. Proyectos de investigación

Las soluciones IoT representan para la Comisión Europea (EC) el siguiente paso hacia la digitalización de nuestra sociedad y economía. Los objetos y las personas están interconectados a través de redes de comunicación, e informan sobre su estado y el entorno que los rodea [41]. Se estima que IoT beneficiará a la economía europea generando crecimiento económico y empleo [139].

IoT es un área emergente que no sólo requiere el desarrollo de infraestructura, sino también el despliegue de nuevos servicios capaces de soportar aplicaciones múltiples, escalables e interoperables. El foco está hoy asociado a la eliminación de silos evitando la existencia de desarrollos específicos de dominio de aplicación (AIOTI y EC están presionando en esta línea). IoT no solo vincula dispositivos conectados a través de Internet, también es un intercambio de datos habilitado por la web para permitir que los sistemas con más capacidades se vuelvan inteligentes y accesibles, creando redes de objetos y permitiendo la integración de datos, servicios y componentes [26].

El ecosistema y la comunidad de IoT ha sido apoyado por diferentes acciones de investigación e innovación durante más de 10 años. Es por eso, que durante los últimos años en el marco de H2020 se han mantenido diferentes iniciativas relacionadas con proyectos de IoT y otras áreas. Dada la amplitud y profundidad potencial de un ecosistema de IoT, esto significa cultivar asociaciones y colaborar con toda la comunidad. En un panorama en constante evolución con nuevos protocolos, tecnologías y, nuevas plataformas, la única forma de garantizar un crecimiento favorable es asociándose con los últimos proveedores y desarrolladores de tecnología de vanguardia; así, como interactuando con otras comunidades de investigación como Big Data, Inteligencia Artificial, Robótica, Electrónica y 5G [57].

En el Capítulo 5 de la presente Tesis Doctoral se presentan con gran detalle una serie de proyectos europeos que han contribuido a la elaboración de la presente tesis doctoral. Los resultados obtenidos de estos proyectos se presentan en los Capítulos 6, 7 y 8.

Finalmente, la siguiente sección pone interés en dos iniciativas europeas. En primer lugar, la IoT-EPI, a la cual pertenece el proyecto INTER-IoT, descrito en el Capítulo 6, iniciativa totalmente centrada en la interoperabilidad y de la que se ha extraído un incalculable valor para el diseño e implementación de la presente tesis doctoral. En segundo lugar, la IoT-LSP, a la cual pertenece el proyecto ACTIVAGE, descrito en el Capítulo 7, que ha ofrecido gran valor

desde el punto de vista de la validación en diferentes entornos de las soluciones creadas.

2.6.1. Proyectos centrados en la interoperabilidad de las Plataformas IoT

La Iniciativa de Plataformas Europeas de Internet de las Cosas (IoT-EPI) se creó para construir un ecosistema de Internet de las Cosas vibrante y sostenible en Europa, maximizando las oportunidades de desarrollo de plataformas, interoperabilidad e intercambio de información. Está formado por siete proyectos de investigación e innovación (AGILE, bIoTope, BIG IoT, Inter-IoT, symbIoTe, TagItSmart, VICINITY) apoyados por dos proyectos de acción de coordinación y apoyo Be-IoT y UNIFY-IoT, que ponen su tecnología a disposición de terceros. Las áreas específicas en las que se centran las actividades de investigación son las arquitecturas y la interoperabilidad semántica, que cubren de forma fiable múltiples casos de uso. Su objetivo es ofrecer una infraestructura configurada dinámicamente y plataformas de integración para objetos inteligentes conectados que cubran múltiples tecnologías y múltiples soluciones inteligentes. La publicación con la descripción de la iniciativa ofrece una visión completa de los objetivos y metas de la misma [140].

La siguiente lista resume los resultados del análisis realizado en el marco de la iniciativa, que ofrece una visión sobre la interoperabilidad en las plataformas y ecosistemas IoT creados y utilizados por IoT-EPI [26]. El documento cubre los aspectos de interoperabilidad, los retos y los enfoques que hacen frente a la interoperabilidad en las actuales plataformas de IoT existentes.

- **AGILE** [141]

  Descripción: Construye una pasarela modular de hardware y software para el IoT centrada en las capas física, de comunicación de red, de procesamiento, de almacenamiento y de aplicación. Los módulos de software de AGILE abordan funciones como la gestión de dispositivos, las redes de comunicación como las redes de área y de sensores y la solución para el almacenamiento distribuido. El proyecto contempla todos los módulos necesarios para proporcionar una solución de gestión de la seguridad robusta.

  Plataformas IoT: Resin.io; Eclipse IoT; Node-RED

- **Big IoT** [142]

  Descripción: Desarrolla una API web genérica y unificada para las plataformas IoT implementadas. Como parte del proyecto, se integran plataformas IoT en el ecosistema. El proyecto se centra en las capas superiores

de la arquitectura de IoT abordando la gestión de la seguridad, las API, la integración de servicios, los servicios de sistemas externos, las aplicaciones y la empresa comercial.

Plataformas IoT: Smart Data Platform; Smart City Platform (Bosch); Wubby Platform; OpenIoT; Traffic Information Centre; BEZIRK Platform.

- **BioTope** [143]

  Descripción: Proporciona una arquitectura y recomendaciones para el uso de estándares abiertos e implementaciones de casos de uso que permiten a las partes interesadas crear fácilmente nuevos sistemas y servicios de IoT y aprovechar rápidamente la información disponible utilizando capacidades avanzadas de Sistemas para Objetos Inteligentes Conectados. bIoTope también desarrolla y proporciona APIs abiertas estandarizadas para permitir la interoperabilidad. El proyecto aborda las principales capas de la arquitectura del IoT y valida las soluciones de interoperabilidad en un entorno multidimensional.

  Plataformas IoT: DIALOG; Node-RED; Warp10; FIWARE; Open IoT; Mist; eAir Web.

- **INTER-IoT** [80]

  Descripción: El proyecto aborda un marco abierto de capas cruzadas, una metodología asociada y herramientas para permitir la interoperabilidad voluntaria entre plataformas heterogéneas de IoT centrándose en seis capas de la arquitectura de IoT definidas: dispositivo, red, middleware, aplicación, datos y transversal. Este proyecto se explica completamente en el Capítulo 6.

  Plataformas IoT: SEAMS, BodyCloud; Node-RED; OpenIoT; FIWARE; UniversAAL; Eclipse OM2M; WSO2; Microsoft Azure IoT suite; Amazon AWS IoT.

- **TagItSmart** [144]

  Descripción: Ofrece un conjunto de herramientas y tecnologías habilitadoras que pueden integrarse en diferentes plataformas de IoT utilizando las APIs proporcionadas para que los usuarios de toda la cadena de valor puedan explotar plenamente la potencia de los códigos funcionales dependientes de la condición para conectar los productos del mercado masivo con el mundo digital en múltiples sectores de aplicación.

  Plataformas IoT: SocioTal; FIWARE; EVRYTHNG; RunMyProcces; Microsfot Azure.

- **SymbioTe** [145]

  Descripción: El proyecto proporciona una capa de abstracción para una visión unificada de varias plataformas IoT y recursos de detección y actuación. Las aplicaciones pueden utilizar los servicios centrales de symbIoTe implementando un motor semántico de IoT para encontrar los recursos adecuados ofrecidos por las plataformas habilitadas para symbIoTe y posteriormente acceder a los recursos virtuales de la plataforma directamente para la adquisición de datos y la actuación [146]. El proyecto se centra en diferentes capas de la arquitectura IoT, desde la capa física hasta la de aplicación, y propone un conjunto completo de gestión de la seguridad.

  Plataformas IoT: OpenIoT; Symphony; MoBaaS; nAssist; Navigo Digitale IoT platform; KIOLA.

- **VICINITY** [147]

  Descripción: Se centra en una plataforma y un ecosistema que proporciona *"interoperabilidad como servicio"* para las infraestructuras en el IoT y aborda las principales capas superiores de la arquitectura del IoT. El trabajo considera la integración de servicios, la lógica empresarial, la virtualización, el almacenamiento, las API, las herramientas, los servicios de sistemas externos, las aplicaciones, la analítica de datos y los servicios en la nube.

  Plataformas IoT: LinkSmart; IoTivity; SiteWhrere; Eclipse Kura; TinyMesh; Gorenje Cloud Sevices

Los siete proyectos proporcionaron diferentes enfoques y se centraron en diferentes verticales, ampliando el ecosistema en diferentes direcciones, y proporcionando la semilla para nuevos desarrollos, como los que se llevaron a cabo en el Programa Europeo de Pilotos a Gran Escala (IoT-LSP) descrita en la siguiente sección.

2.6.2. Pilotos a gran escala centrados en la interoperabilidad IoT

IoT-LSP es un clúster formado por cinco pilotos a gran escala financiados en la convocatoria H2020 IoT1 y apoyados por dos CSA (CREATE-IoT y U4IoT). Los programas piloto y de experimentación a gran escala, aprovechan los ecosistemas de innovación y las plataformas tecnológicas. Se han encargado de activar relaciones que puedan estimular la generación de valor socioeconómico significativo para el conocimiento en el futuro. Los ecosistemas de IoT creados entorno a estos instrumentos ofrecen los mecanismos para fomentar y agilizar la participación de todos los actores en el proceso de innovación [148].

IoT-LSP ha incluido los consorcios de innovación que están colaborando para fomentar el despliegue de soluciones de IoT en Europa a través de la integración de tecnologías avanzadas de IoT en toda la cadena de valor, la demostración de múltiples aplicaciones de IoT a escala y en un contexto de uso, y lo más cerca posible de las condiciones operativas [57]. Los proyectos implicados son ACTIVAGE, MONICA, IoF2020, AUTOPILOT y SYNCHRONICITY.

En paralelo, se desarrolla la iniciativa CREATE-IoT, cuyo objetivo es estimular la colaboración entre las iniciativas de IoT. Se encarga de buscar una sincronización y alineación estratégica y operativa a través de intercambios frecuentes y multidireccionales entre las diversas actividades de las áreas de interés de IoT. Sin embargo, también requiere una fertilización cruzada de los diversos pilotos a gran escala de la IoT para cuestiones tecnológicas y de validación de interés común en los diferentes dominios de aplicación y casos de uso. El impacto de CREATE-IoT se produce por su interacción con los pilotos a gran escala (LSP), junto con otras actividades europeas de IoT, como las asociaciones de Participación Público-Privada (PPP) y las iniciativas tecnológicas conjuntas.

Por otro lado, U4IoT, se encarga de colaborar en la participación de los usuarios en proyectos piloto a gran escala en IoT. Esta CSA proporciona a los pilotos métodos de participación de los usuarios finales, como talleres de creación conjunta, crowdsourcing y Living Labs. Proporciona herramientas y recursos en línea para la participación de los usuarios finales, el crowdsourcing y la protección de datos personales.

Los ámbitos de interés LSP son muy variados y abarcan los entornos de vida para envejecer bien, la agricultura inteligente y la seguridad alimentaria, los wearables para los ecosistemas inteligentes, las zonas de referencia en las ciudades de la UE y los vehículos autónomos en un entorno conectado. Esta heterogeneidad significa que el enfoque de CREATE-IoT se centra en asegurar que haya un elemento sólido de análisis inicial realizado en los diferentes conceptos para identificar las actividades no contiguas que puedan beneficiarse de los enfoques compartidos, ya sean de naturaleza tecnológica, política o empresarial.

A continuación, se ofrece una visión general de cómo cada piloto cubre la interoperabilidad en su campo de intereses y las plataformas involucradas en los diferentes proyectos que forman parte de los LSP [83].

- **ACTIVAGE** [81]

  Descripción: ACTIVAGE ha desarrollado e implementado una Arquitectura de Referencia de Nivel Alto (HLA). Esta arquitectura está adaptada a las necesidades del Envejecimiento Activo y Saludable (AHA) y se basa en un modelo de capas para garantizar la intermediación entre las aplicaciones y la capa de dispositivos sensores (edge). Se trata de tres capas

principales (i) La capa de servicios, llamada AIOTES, es un conjunto de soluciones de software, herramientas y metodologías de apoyo a la interoperabilidad semántica, la seguridad, la privacidad y la protección de datos. (ii) La Capa de Interoperabilidad es una capa de abstracción encargada de garantizar la interoperabilidad a través de las plataformas ACTIVAGE. (iii) Capa de Plataforma IoT. El middleware IoT encargado de conectar todas las cosas involucradas en los casos de uso de ACTIVAGE es complejo y heterogéneo. La capa de la plataforma sirve como una capa de abstracción que garantizará que las diferentes plataformas pueden ser soportadas, y que un determinado servicio puede ser replicado a través de diferentes sitios piloto. Este proyecto será descrito en el Capítulo 7.

Plataformas y tecnologías involucradas: FIWARE; IoTivity; OpenIoT; SensiNact; SOFIA2; universAAL.

- **IoF2020** [149]

  Descripción: El proyecto está formado por 19 casos de uso agrupados en 5 ensayos con usuarios finales de los sectores verticales de los cultivos, los productos lácteos, las frutas, las verduras y la carne y con integradores de IoT que demuestran el caso de negocio de las soluciones innovadoras de IoT para un gran número de áreas de aplicación. Los principales puntos de interoperabilidad resultantes del proyecto son (i) un habilitador de conectividad para Dispositivos IoT y maquinaria agrícola, (ii) permite la exposición de los datos y servicios ofrecidos por los Dispositivos IoT a través de interfaces programáticas conocidas, (iii) permite la transformación, agregación, armonización y publicación, como información de contexto, de los datos armonizados procedentes de los Dispositivos IoT, la maquinaria agrícola u otras fuentes de información (portales de datos abiertos, servicios web que proporcionan datos contextuales, etc.). Por otro lado, expone una forma unificada de enviar comandos y mediar con los Dispositivos IoT o la maquinaria agrícola, independientemente de la interfaz expuesta por la Capa de Servicios IoT o la Maquinaria Física, (iv) proporciona acceso a todos los datos (datos en tiempo real e históricos o resultados de analíticas) de interés para las aplicaciones agrícolas inteligentes, (v) permite a las Capas de Aplicación y Mediación consumir Geo-Servicios públicos y (vi) un punto de interoperabilidad transversal que facilita el intercambio seguro de información entre las diferentes capas y actores.

  Plataformas y tecnologías involucradas: : 365FarmNet; AgroSense; Apache Cassandra; Apache Flink; Apache Spark; Arvalis IoT Platform;

Altland FMIS; Connecterra IoT; Cygnus; EBBITS; EPCIS; FIWARE; FIBSPACE; LinkSmart; MongoDB; OpenStack; Qlip (calibración y validación automatica); ThingWorx IoT; VIRTUS.

- **MONICA** [150]

  Descripción: MONICA presenta un ecosistema de IoT a gran escala que utiliza innovadores sensores y actuadores de IoT portátiles con servicios de back-end integrados en una plataforma interoperable basada en la nube capaz de ofrecer una multitud de aplicaciones simultáneas y específicas. Proporciona (i) una capa de API, (ii) una capa de servicios donde se implementa la inteligencia de la plataforma y se integran los módulos de procesamiento. (iii) una capa de IoT que se encarga de la interoperabilidad utilizando dos frameworks de código abierto: LinkSmart (middleware IoT) y SCRAL (capa de abstracción IoT). (iv) Capa de borde, un conjunto de módulos de procesamiento de datos en tiempo real directamente desde la capa de dispositivos, (v) Capa de red, que permite la comunicación efectiva entre los wearables heterogéneos de IoT, los dispositivos de IoT y los módulos de la plataforma de IoT y (vi) Capa de dispositivos, que incluye todos los wearables y sensores de IoT.

  Plataformas y tecnologías involucradas: LinkSmart middleware; SCRAL adaptation framework; GOST (Go-SensorThings) IoT; OGC SensorThings.

- **AUTOPILOT** [151]

  Descripción: AUTOPILOT desarrolla nuevos servicios sobre el IoT para implicar a los vehículos de conducción autónoma, como el uso compartido de vehículos autónomos, el aparcamiento automatizado o los mapas dinámicos digitales mejorados para permitir una conducción totalmente autónoma. Su marco de interoperabilidad se logra sobre la base de: (i) La plataforma de interoperabilidad OneM2M y las pasarelas de interoperabilidad. Las plataformas propietarias de IoT se interconectan mediante pasarelas de funcionamiento interconectado y la plataforma de interoperabilidad oneM2M. (ii) Modelos de datos IoT estandarizados, para especificar la sintaxis y la semántica de los datos.

  Plataformas y tecnologías involucradas: OneM2M; FIWARE; Huawei Platform; Watson IoT Platform.

- **SYNCHRONICITY** [152]

  Descripción: El objetivo de SynchroniCity es establecer una arquitectura de referencia para el mercado previsto de la ciudad habilitada para el

IoT con puntos de interoperabilidad identificados e interfaces y modelos de datos para diferentes verticales. Proporciona: (i) Una API estándar común para la gestión de la información de contexto. El gestor de datos de contexto (Context Data Broker) es un componente clave de la arquitectura SynchroniCity y la implementación de su API (conforme a la NGSI API) se considera un "punto de interoperabilidad"para permitir que las ciudades participen en la plataforma SynchroniCity. (ii) Un conjunto común de modelos de información: la interoperabilidad semántica, lograda a través de la adopción de modelos de datos comunes, se introduce en la arquitectura como requisito básico para permitir la reutilización de aplicaciones en diferentes ciudades y dominios. (iii) Un conjunto de plataformas de publicación de datos con estándares comunes (el papel de los datos es crucial en SynchroniCity). Por esta razón, la arquitectura de referencia incluye componentes específicos de gestión de datos que pretenden proporcionar, a través de interfaces estándar, todas las funcionalidades relacionadas con la gestión del ciclo de vida de los datos.

Plataformas y tecnologías involucradas: Mosquitto MQTT; Fiware Stack.

Como recapitulación, el folleto *Programa europeo de pilotos a gran escala de IoT* [153] y el libro interactivo [154] ofrecen una visión compartida y detallada de cada uno de los proyectos. Los resultados comunes recopilados por los grupos de trabajo de CREATE-IoT están disponibles en su página web [155]. El programa IoT-LSP ha ayudado a la comunidad investigadora de IoT a navegar por el entorno tecnológico, identificar prioridades y lagunas, y definir arquitecturas de referencia cada vez más importantes. Entre sus principales contribuciones se encuentran, por ejemplo [57]:

- El desarrollo colaborativo de un modelo de arquitectura de referencia en 3D que amplió el alcance de la especificación de la arquitectura y tuvo como objetivo contribuir a la estandarización.
- El desarrollo de los requisitos de un nuevo estándar para los enlaces de datos de tiempo crítico para los sensores IoT.
- Las contribuciones de los pilotos a la ontología SAREF que se está ampliando a entornos de aplicación IoT, como por ejemplo, las ciudades inteligentes y la alimentación agrícola inteligente, contribuyendo al desarrollo de un sólido ecosistema de normas de la UE.
- Las recomendaciones sobre la API abierta para IoT en las Ciudades Inteligentes y el Informe Técnico sobre la Inteligencia Artificial en el ecosistema de IoT y Ciudades Inteligentes.

2.7. Tecnologías habilitadoras de la interoperabilidad

Reciben el nombre de tecnologías habilitadoras de la interoperabilidad aquellas tecnologías que satisfacen los requisitos transversales de los diferentes mecanismos de interoperabilidad middleware y de aplicación y servicios. Estas características comunes son la seguridad, las interacciones de los diferentes mecanismos de interoperabilidad propuestos y la virtualización y la agrupación en clústeres de las soluciones de interoperabilidad propuestas.

Este tipo de tecnologías habilitadoras son el elemento de unión necesario para conectar los mecanismos de interoperabilidad con el marco de soporte (que se presentará en la siguiente sección) y ofrecer una solución software completa. La aceptación positiva por parte de los usuarios de la solución propuesta en esta Tesis Doctoral depende de estas tecnologías habilitadoras. Por ejemplo, no será exitosa una solución que no asegure la seguridad y la privacidad de la información y las comunicaciones. Tampoco será aceptada una solución que no ofrezca un despliegue sencillo y eficaz. En los congresos o presentaciones industriales en los que el autor de la Tesis Doctoral ha presentado los mecanismos de interoperabilidad, la seguridad y la virtualización de dichos mecanismos, siempre ha sido una pregunta recurrente. Se procede a detallar las principales conclusiones obtenidas en un análisis previo de estas tecnologías habilitadoras de la interoperabilidad.

2.7.1. Seguridad y privacidad

La seguridad es uno de los aspectos más relevantes en IoT. Constantemente aparecen nuevas tendencias, lo cuál implica una búsqueda continua de información y prueba de nuevas tecnologías.

Los principales desafíos de seguridad en IoT que han manejado de diferentes maneras las plataformas son [156] la seguridad e integridad de las comunicaciones, el control de acceso, cifrado de datos, autorización, autenticación y gestión de identidad.

Este análisis ofrecerá unas pinceladas de como las plataformas IoT están tratando con los aspectos relacionados con los requisitos de interoperabilidad. También se detallarán ciertas tecnologías específicas que permiten satisfacer las necesidades de seguridad de una solución como la propuesta.

Plataformas IoT

Se presenta a continuación la información destacada sobre gestión de la seguridad de una selección de plataformas IoT.

FIWARE [93]

FIWARE ofrece servicios y herramientas que le permiten administrar la autenticación y autorización en sus aplicaciones y servicios de backend. Keyrock es el componente de FIWARE responsable de la gestión de identidades. El uso de Keyrock junto con otros componentes de seguridad como WILMA PEP Proxy y Authzforce le permite agregar seguridad de autorización y autenticación basada en OAuth2 a sus servicios y aplicaciones. WILMA protege los recursos contra accesos no autorizados, ejerce el papel de actuar de proxy inverso delante de los servicios REST que de esta manera quedan protegidos. AuthForce proporciona una API para gestionar las políticas de autorización y hacer cumplir las decisiones basadas en ellas, cuando se solicite por parte de WILMA para autorizar o denegar las solicitudes de acceso a los servicios.

En la Sección 8.1, relacionado con el caso de uso PIXEL se proporcionan más detalles de la implementación de estos mecanismos de seguridad en FIWARE.

UniverAAL [94]

La seguridad de las comunicaciones entre los componentes y servicios internos se hace con el mecanismo RSA sobre el bus de comunicaciones. Externamente, se usa el protocolo SSL/TLS con autenticación de servidor basada en certificados.

La gestión de permisos de los miembros del bus se implementa en Java de forma especifica para UniversAAL. Las políticas adicionales de control de acceso pueden ser realizadas por el propietario de la aplicación con una interfaz de usuario.

La implementación de la gestión de autorizaciones se realiza mediante nombre de usuario/contraseña. El servicio de autenticación valida las credenciales introducidas frente a las credenciales de seguridad de lo usuarios almacenados en el servicio de perfil local de uAAL. La autenticación correcta da lugar a la creación de una sesión. La sesión de usuario autenticada permite a los servicios de aplicación tomar decisiones de autorización basadas en las políticas de control de acceso.

Los servicios de aplicación para la seguridad proporcionan un valor añadido a la plataforma universAAL, como la encriptación de documentos y la gestión del consentimiento. Al tratarse de una plataforma orientada a los datos sensibles, la información almacenada se encripta, además, según el estándar IHE DEN (Integrating the Healthcare Enterprise Document Encryption).

SOFIA2 [95]

Un procesador de conocimiento es el elemento permite el acceso para conectar cada aplicación con el espacio inteligente a través del broker de información semántica. La privacidad y seguridad sobre las comunicaciones entre algún cliente y SOFIA2 se determina por:

- Nombre del procesador del conocimiento, proporcionado por el administrador de SOFIA2.
- Identificador único del procesador de conocimiento. El propio cliente crea un identificador único para cada aplicación. Por ejemplo, se puede usar el número de serie de cada dispositivo como identificador único del KP.
- Token. Es una clave alfanumérica que permite el acceso al bróker de información semántica. Proporcionado por el administrador de SOFIA2.

Sofia2 define la privacidad y la seguridad sobre cualquier usuario, procesador del conocimiento, broker de información semántica y ontologías. Se ofrecen plugins para gestionar la seguridad de identidad del usuario, seguridad de consola y autorización backend basados en claves token de sesión.

OPENIOT [96]

La plataforma OpenIoT implementa la seguridad en forma de un módulo de seguridad, el Servicio de Autenticación Central (CAS), este se basa en Jasig CAS y proporciona autenticación y autorización OAuth2.0 para todos los demás módulos de OpenIoT.

Los roles son contenedores que contienen permisos y se implementan con el wildcard de Apache Shiro. Un acceso exitoso otorga al cliente un único token de sesión con un tiempo de expiración manejable.

OM2M [99]

El protocolo SSL/TLS se puede habilitar requiriendo un certificado de seguridad X.509 Diferentes recursos permiten el manejo del control de acceso:

- Política de control de acceso.
- Regla de control de acceso.
- Control de acceso basado en roles.

Hay una autenticación mutua con cada originador y receptor. Las entidades utilizan la autenticación mutua, ambas envían tokens de autenticación derivados de las credenciales maestras, ambas envían un desafío de autenticación que

puede o no ser aleatorio dependiendo de los parámetros de seguridad. La autenticación mutua se aplica a esquemas basados en claves simétricas y asimétricas. La autorización controla el acceso a los recursos y servicios alojados.

Azure IoT [101]

Esta solución en la nube añade funcionalidades para cada elemento de su implementación, incluidos los servicios en la nube y los dispositivos:

- Administración de eventos e información de seguridad para IoT integrado.
- Orquestación, automatización y respuesta de seguridad).
- Soluciones de detección y respuesta extendidas.
- Clave de identidad única para cada dispositivo que se utiliza para establecer un token de sesión.
- Gestión de claves implementada en la nube con SQL.
- Autenticación y autorización a nivel de canal contra la puerta de enlace.
- Validación de firmas contra registro de identidad y lista negra.
- Azure IoT Hub otorga acceso a los puntos finales al verificar un token con las políticas de acceso compartido y las credenciales de seguridad del registro de identidad.
- Mecanismos en la nube de Azure, incluidos Azure Active Directory y Key Vault.

AWS IOT [103]

Cada dispositivo o cliente conectado debe tener una credencial para interactuar con AWS IoT. Todo el tráfico hacia y desde AWS IoT se envía de forma segura a través de Transport Layer Security (TLS). Los mecanismos de seguridad de la nube de AWS protegen los datos mientras se mueven entre AWS IoT y otros servicios de AWS. Se ofrecen componentes para:

- Administración de Identidades (IAM) de Roles: Manejar usuarios, grupos y roles.
- IAM Policies: Reglas.
- AWS Security Credentials: Credenciales de seguridad.

WSO2 [98]

WSO2 es una solución middleware de arquitectura orientada a servicios de código abierto. Está diseñado con componentes independientes, por lo que se puede adaptar para una solución a medida:

- Ofrece un gestor de identidades: Es impulsado por APIs y diseñado para crear soluciones de gestión de acceso de identidades efectivas. Se basa en estándares abiertos como SAML, OAuth y OIDC con las opciones de implementación local, en la nube e híbrida. Admite requisitos complejos de gestión de identidades dada su alta extensibilidad.
- Ofrece capacidades de gestionar APIs: Expone las API de forma fácil y segura a consumidores internos y externos. Implementa flujos de autorización estándar de la industria, como OAuth, OpenID Connect y JWT, listos para usar e integrar con el gestor de identidades existente o herramientas de administración de claves.

Sus componentes de seguridad se pueden desplegar localmente, en servidores privados o en la nube. El uso del gestor de identidades y el gestor de APIs de WSO2 garantiza una mejor integración entre ambas tecnologías.

Conclusión clave

Las plataformas deben garantizar la seguridad de las comunicaciones, una política de control de acceso e implementación de la gestión de autorizaciones. También pueden proporcionar una serie de mecanismos adicionales.

Como se ha indicado, el Capítulo 6 sigue una aproximación principalmente basada en los componentes que ofrece la plataforma WSO2, y la sección 8.1 presenta un proyecto que utiliza la aproximación ofrecida por FIWARE. Los Capítulos 7 y 8 que tenían necesidades más complejas de integración con otros elementos han utilizado herramientas específicas de seguridad disponibles en repositorios abiertos y no desarrolladas en el marco del ecosistema de las plataformas IoT.

Herramientas existentes de seguridad

Además de analizar las principales implementaciones de la seguridad se han analizado una serie de herramientas que han sido de utilidad en el desarrollo de la presente Tesis Doctoral, como las que se encargan de:

- Seguridad en las comunicaciones: Todas las comunicaciones deben estar protegidas mediante la implementación de un cifrado punto a punto y

una autorización de extremo a extremo. Esto se implementa mediante el uso del protocolo HTTPS en las comunicaciones.

  - Por ejemplo, soluciones como auto-proxy funcionan sobre Docker y configura un contenedor que ejecuta un servidor http con compatibilidad integrada con Let's Encrypt para generar automáticamente certificados SSL/TLS y compatibilidad integrada con HTTPS.

- Gestión de identidades: Proporcionan soluciones para la gestión de identidades que podrían conectarse a los componentes desarrollados en la presente Tesis Doctoral. Estas soluciones deben proporcionar una forma de administrar usuarios, organizaciones, roles y aplicaciones.

  - Keycloack [157]: Esta solución admite "Single-Sign-On". Soporta varios protocolos como OpenID Connect, OAuth 2.0, SAML 2.0, inicio de sesión en redes sociales y admite LDAP y Active Directory. También admite políticas de contraseña personalizadas. Está diseñado para ser extensible para agregar nuevas funcionalidades personalizadas con la ayuda de un desarrollador experimentado. Keycloak tiene una gran comunidad de respaldo tras él.
  - WSO2 [98]: Aprovecha todas las facilidades que ofrece la plataforma WSO2 que permite desplegar sus componentes por separado. Proporciona una integración muy interesante con el componente de gestión de APIs de la misma plataforma. Ya se ha descrito con mayor detalle en el apartado anterior.
  - Keyrock [158]: También se ha descrito en el apartado anterior. Su uso es recomendable cuando se quiere dotar de gestión de identidades a soluciones desarrolladas sobre el ecosistema FIWARE.

- Gestión de las APIs: Proporcionan soluciones para administrar el acceso a las API, unificarlas en un punto de acceso común y protegerlas.

  - WSO2 [98]: El gestor de APIs de la plataforma, presentado en la sección anterior, ofrece un interesante numero de herramientas para configurar y manejar las APIs. Además, ofrece herramientas gráficas para realizar estas acciones. No obstante, su curva de aprendizaje es algo elevada.
  - Kong [159]: es una solución abierta de gestión de API gateway para microservicios muy popular y con un gran número de desarrolladores. Por ejemplo, esta comunidad ha desarrollado el componente

Konga que es un interfaz gráfico para manejarla. La curva de aprendizaje del manejo de Kong no es tan alta como la de WSO2. No obstante, en caso de necesitar programar algún plugin se deben programar con el lenguaje de programación LUA. Finalmente, destacar que existe una versión libre de Kong y otra de pago, con mayor número de funcionalidades.

- API Umbrella [160]: Es una plataforma de gestión de API de código abierto para exponer las API de servicios web. Actualmente, se está integrando en el entorno FIWARE, y parece una interesante alternativa a valorar en un futuro cercano.
- Express API Gateway [161]: Este API gateway de microservicios está construida sobre Express.js. Su curva de aprendizaje es rápida, y por tanto, se puede empezar a usar con facilidad. No ofrece un entorno gráfico de gestión. Tiene menos contribuidores que Kong.
- Traefik [162]: Proxy inverso de soluciones de contenedores, encaminador y edge router. Por tanto, además de funcionalidades de API Gateway también tiene funciones de encaminamiento para tareas más complejas.
- Istio [163]: Es una plataforma de red de servicios con tecnología open source, que permite controlar el intercambio de datos entre los microservicios. Tiene muchas más funcionalidades que las de una API Gateway. Tiene una curva de aprendizaje elevada. Istio está enfocada a Kubernetes, lo amplia para establecer una red programable y compatible con aplicaciones mediante el proxy de servicio Envoy. Al trabajar tanto con Kubernetes como con cargas de trabajo tradicionales, Istio ofrece seguridad, telemetría y administración de tráfico universal y estándar en implementaciones complejas.

Se ha mostrado un listado que corresponde con una pequeña parte de las muchas herramientas que se han probado o analizado durante estos años. Dado que los mecanismos de seguridad son mecanismos habilitadores, se han presentado aquellas soluciones que principalmente se han usado en los proyectos y prototipos o que se están planteando para futuros desarrollos.

2.7.2. Interacción entre capas

La aproximación a la interacción entre capas es una propuesta novel de la solución de interoperabilidad propuesta por el proyecto INTER-IoT, correspondiente al producto INTER-Layer mediante su componente Cross-Layer. La concepción de dicha aproximación ha acontecido y se ha realizado en paralelo

con la presente Tesis Doctoral. Este detalle se explicará detalladamente en los Capítulos 5 y 6 de este documento.

Hay diferencias entre la aproximación seguida por INTER-IoT y la seguida por la presente Tesis Doctoral. La aproximación de INTER-IoT contempla una interacción potencial entre las capas de interoperabilidad de dispositivo, red, middleware, aplicación y datos. La de la presente Tesis Doctoral está centrada y extiende con más detalle las interacciones ponteciales entre la capa de middleware y la de aplicación y servicios.

Ambas aproximaciones coinciden en que las llamadas a las APIs ofrecidas por un mecanismo de interoperabilidad son la pieza clave para lograr esta interacción entre capas. Estas llamadas pueden realizarse en los elementos del componente o en sus conectores. Permiten añadir las funcionalidades de una capa, como nuevas funcionalidades de una capa, o también permiten entender una capa como un agente externo conectado a dicha capa de interoperabilidad mediante un conector. Este tipo de llamadas conjuntas entre capas da un valor añadido al mecanismo de interoperabilidad ofrecida.

Finalmente, este tipo de interacciones también pueden ser orquestadas por el marco de interoperabilidad que se describirá en la siguiente sección. En este caso el marco ofrece las herramientas para que los usuarios y desarrolladores puedan implementar aplicaciones con interacciones entre capas a su medida.

2.7.3. Virtualización y Clusterización

La solución propuesta necesita aprovechar las herramientas de virtualización para mejorar la escalabilidad, portabilidad, aislamiento y flexibilidad. Además, una de las principales ventajas de la agrupación en un cluster es simplificar el despliegue y la gestión de grandes grupos de servidores. De esta manera, se obtiene acceso a un punto único de configuración para diferentes grupos de servidores. Para el administrador del sistema se obtiene una vista como si fuera una única estación de trabajo.

Se han analizado herramientas de virtualización de servidores como KVM, Xen, VMWARE, VirtualBox, Hyper-V o Proxmox, además de soluciones para proporcionar infraestructura como servicio, como OpenStack [164]. No obstante, se necesitaba una aproximación más cercana a los servicios y no a todo el servidor, como ofrecían las soluciones basadas en contenedores como LXC, RKT o Docker [165].

La tecnología seleccionada tras este análisis fue Docker [166]. Docker permite empaquetar una aplicación con todas sus dependencias en una unidad estandarizada para el desarrollo de software. Los contenedores Docker envuelven una pieza de software en un sistema de archivos completo que contiene todo lo que necesita para ejecutarse: código, tiempo de ejecución, herramientas

del sistema, bibliotecas del sistema. En resumen, contiene cualquier cosa que pueda instalarse en un servidor. Esto garantiza que siempre se ejecutará de la misma manera, independientemente del entorno en el que se esté ejecutando.

Docker tiene una comunidad muy extendida de desarrollos y soluciones [167] sobre ella para:

- Manejar operaciones con contenedores: Composición de contenedores, despliegue e infraestructura, monitorización, redes, orquestación, proxy inverso, funcionamiento, seguridad, descubrimiento de servicios, gestión de volúmenes de datos, interfaz de usuario, integraciones IDE, soluciones escritorio, soluciones terminal o herramientas consola.
- Crear imágenes de contenedores Docker: Herramientas de base, constructor de imágenes, linter, metadatos o registro de contendores.
- Desarrollar con Docker: Cliente API, CI/CD, entorno de desarrollo, recogida de basura, soluciones sin servidor, soluciones para pruebas o wrappers
- Servicios basados en Docker: Servicios de integración continua, de monitorización o de gestión.

Las principales utilidades que se han considerado durante la presente Tesis Doctoral son:

- Docker Registry: El conjunto de herramientas de Docker para empaquetar, enviar, almacenar y entregar contenido.
- Docker Compose: Definir y ejecutar aplicaciones multicontenedor con Docker-
- Docker Swarm: Un sistema de clustering nativo de Docker.
- Docker Portainer: Una interfaz de administración ligera para gestionar un host Docker o un clúster Docker Swarm cluster.

Desde el punto de vista de las necesidades de manejar clusters de servidores, la herramienta Docker Swarm proporciona una administración unitaria de un cluster Docker proporcionando también características de planificación y gestión. Docker Swarm permite una gestión centralizada de la propiedad desde un único punto de acceso a todo el entorno Docker. Estos servidores poseen su propia consola de gestión, pero al pertenecer todos a un mismo despliegue Swarm, son gestionados de forma centralizada por la consola de Docker Swarm, que unifica las características de cada contenedor Docker convirtiéndolos en un

único elemento software. Docker Portainer es una interfaz de gestión ligera y consiste en un único contenedor que puede ejecutarse en cualquier motor Docker. Permite una fácil gestión de ambos tipos de entornos Docker, hosts Docker y clusters Swarm. Su objetivo es ser una herramienta de gestión que ofrece una forma más cómoda de trabajar con el entorno Docker.

En parte de este trabajo también se ha utilizado Kubernetes. Esta tecnología consiste en un sistema de código abierto para automatizar el despliegue, el escalado y la gestión de aplicaciones en contenedores. Agrupa los contenedores que componen una aplicación en unidades lógicas para facilitar la gestión y el descubrimiento. Actualmente es una solución en auge y muy extendida, pero también tiene una alta curva de aprendizaje. No obstante, en este trabajo se ha priorizado el uso de Docker Swarm. Por un lado, debido a los recursos con los que se contaba para desplegar los mecanismos de interoperabilidad y, por otro lado, porque Kubernetes no estaba tan extendida en el periodo temporal en que se realizaron los primeros componentes del presente trabajo. Las partes en las que fue necesario desplegar Kubernetes, se realizaron mediante Minikube. Es una distribución de Kubernetes local, centrada en facilitar el aprendizaje y el desarrollo para Kubernetes. Para desplegarlo solo era necesario un entorno Docker o un entorno de máquina virtual.

En la actualidad, proyectos del grupo de investigación del presente autor (SATRD) están enfocando sus futuros desarrollos en tecnologías Kubernetes, como en el actual proyecto llamado ASSIST-IoT [57].

2.8. Marcos de soporte software para la Interoperabilidad IoT

El concepto de Marco es un término amplio y correspondiente a diferentes disciplinas de ingeniería del software. El proyecto INTER-IoT dispone de extensa documentación de análisis de la situación de este tipo de herramientas y su aplicación en IoT [168]. Esto es importante, porque el desarrollo del marco está ligado a las funcionalidades que se deseen ofrecer a los usuarios y se realizó en dicho proyecto un extenso trabajo previo de análisis. En el caso de la presente Tesis Doctoral, el marco de soporte corresponde con los elementos software que permiten el acceso, uso y gestión de los mecanismos de interoperabilidad IoT desarrollados. Es por eso que, en primer lugar, se ha procedido a analizar los elementos software que habitualmente han sido relacionados con soluciones de marco de desarrollo:

Librerías

Una librería es una colección de código relacionado con una tarea específica o un conjunto de tareas estrechamente relacionadas que operan en cada dominio. Las librerías están desarrolladas para ser llamadas por agentes externos, evitando cualquier control sobre las aplicaciones.

Conjunto de herramientas

Un conjunto de herramientas es una librería con un propósito definido y específico. Los conjuntos de herramientas generalmente operan en un nivel más alto de abstracción de una biblioteca, siendo muy a menudo consumidores de librerías.

Marco

Un marco es un conjunto de librería y módulos de software interrelacionados cuyo objetivo es resolver un problema o un conjunto de problemas unidos.

La definición de un marco suele estar vinculada a una de sus propiedades clave. Principalmente se basan en que el flujo del programa es asumido por el marco, aunque este sea un elemento preexistente y externo al programa. Esto es la solución opuesta al enfoque tradicional de la programación imperativa, donde el flujo del programa es controlado por el programa desarrollado. Los marcos también están dirigidos a un resultado específico, una aplicación para un sistema operativo específico o para un trabajo más general, como marcos de herramientas de programación.

De manera general, se puede indicar que los marcos contienen librerías. Un código desarrollado llama a las librerías para funcionar. En contraposición, los marcos son los encargados de llamar al código para ponerlo en marcha [169].

Kit de desarrollo de software (SDK)

Un SDK es una colección de herramientas para ayudar al programador a crear e implementar código y contenido que está diseñado específicamente para ejecutarse en una plataforma muy particular o de una manera muy particular.

Motor

Un motor software se refiere a una parte central y necesaria de un sistema o subsistema de software. No obstante, en este análisis la acepción de motor más acertada es la de un software que facilita procesos automatizados, en los que diferentes elementos de software interactúan para minimizar la intervención humana. Ejemplos de motores de software incluyen motores de bases de datos

relacionales, motores de flujo de trabajo, motores de inferencia y motores de búsqueda.

Interfaz de programación de aplicaciones (API)

Una API [170] ofrece una interfaz externa para ser llamada por otra pieza de software. Más formalmente, una API es un conjunto de reglas y especificaciones que los programas de software pueden seguir para comunicarse entre sí. Sirve como interfaz entre diferentes programas de software y facilita su interacción, al igual que la forma en que la interfaz de usuario facilita la interacción entre humanos y computadoras. Las APIs son extremadamente populares en los últimos tiempos como un patrón para superar las limitaciones de interoperabilidad de la creación y expansión continua de servicios de las tecnologías de la información. Otro impulsor importante de la popularidad de las APIs es la creación de ecosistemas centrados en aplicaciones, lo que permite a terceros crear clientes, usar datos y agregar valor a las características de los sistemas existentes [171].

En general, las APIs dependen del lengauje, ya que cada interfaz es un contrato especificado en un protocolo determinado. Sin embargo, las API pueden hacer uso de estándares o protocolos de comunicación (por ejemplo, HTTP, SOAP, COAP ...) para superar esta limitación y ser independientes del lenguaje de programación, aunque manteniendo la dependencia de la sintaxis con el protocolo seleccionado. Así, la clasificación de las APIs se puede hacer de la siguiente manera:

- Lenguaje API: APIs que suelen formar parte de un SDK y están compuestas por bibliotecas y anotaciones para ser invocadas en un lenguaje de programación concreto como Java, .NET, Ruby, JavaScript, Python, etc.
- Protocolo API: estas APIs, normalmente accesibles dentro de una red (principalmente Internet) se ejecutan en los sistemas de alojamiento. También pueden denominarse Servicios Web, ya que suelen estar expuestos a través de la Web para ofrecer servicios a integradores y terceros. El consorcio W3C define los servicios web como sistemas de software diseñados para admitir una interacción interoperable de máquina a máquina a través de una red. Los tres estilos arquitectónicos más comunes en los servicios web son:
    - RPC (Llamadas a procedimientos remotos): los servicios web basados en RPC presentan una interfaz de llamada para funciones y procedimientos distribuidos. Normalmente, la unidad básica de este tipo de servicios es la operación WSDL (descriptor de servicio).

- SOAP (Protocolo simple de acceso a objetos): Los servicios web se pueden implementar siguiendo los conceptos de la arquitectura SOA (Arquitectura orientada a servicios), donde la unidad básica de comunicación es el mensaje, más que la operación. Esto se conoce normalmente como servicios orientados a mensajes.
- REST (Transferencia de estado de representación): los servicios web basados en REST intentan emular el protocolo HTTP o protocolos similares restringiendo la interfaz a un conjunto conocido de operaciones estándar (por ejemplo, GET, PUT, etc.). Por tanto, este estilo está más enfocado a interactuar con recursos con estado, que con mensajes y operaciones.

Las APIs constituyen una poderosa herramienta de interoperabilidad que permite la comunicación dentro de una infraestructura heterogénea que da lugar a componentes poco acoplados. Los servicios RESTful se ajustan a los principios arquitectónicos de REST, que incluyen restricciones relativas a la comunicación cliente-servidor, la ausencia de estado de la solicitud y el uso de una interfaz uniforme. Además, estos servicios se caracterizan por el desacoplamiento recurso-representación, de manera que el contenido del recurso puede manifestarse en diferentes formatos (es decir, JSON, HTML, XML, etc.). Además, la mayoría de las APIs de la Web se basan en URIs para la identificación e interacción de los recursos y en el protocolo HTTP para la transmisión de mensajes, lo que da lugar a una pila tecnológica sencilla que proporciona acceso a terceros, para que puedan consumir y reutilizar los datos que se originan en diversos servicios en composiciones de servicios orientados a datos denominadas mashups [172].

La evolución de las arquitecturas monolíticas orientadas a servicios (SOA) ha partido de la gestión de la complejidad de los sistemas distribuidos en el ámbito de la integración de diferentes aplicaciones de software, y ha evolucionado a través de los microservicios hacia las arquitecturas sin servidor. En las arquitecturas orientadas a los servicios, un servicio proporciona funcionalidades a otros servicios principalmente mediante el paso de mensajes. Con la modularización de estas arquitecturas en ecosistemas de microservicios, los diferentes servicios se desarrollan y escalan de forma independiente entre sí según sus requisitos específicos y los estímulos reales de las solicitudes, lo que lleva a la localización de las decisiones por servicio en cuanto a lenguajes de programación, bibliotecas, marcos de trabajo, etc. Sin embargo, el auge de la computación en la nube ha dado lugar a arquitecturas sin servidor que soportan la asignación dinámica de recursos y la correspondiente gestión de la infraestructura con el fin de permitir el autoescalado basado en el estímulo de eventos y minimizar los costes operativos [173]. En ese contexto, las APIs de la

Web deben considerarse la piedra angular de la evolución explosiva de la creación de valor de los servicios. Sin embargo, desde el inicio de la prevalencia de las APIs sobre las tecnologías tradicionales de servicios web, las primeras han evolucionado de forma autónoma, careciendo de un lenguaje de definición de interfaces establecido. De ahí que, en términos de infraestructuras sin servidor, haya surgido la consiguiente necesidad de homogeneidad en el diseño y desarrollo de aplicaciones. El denominador común en el desarrollo de las funciones sin servidor es su capacidad para soportar diferentes funcionalidades de forma escalable y sin estado [174].

OpenAPI

La solución a la falta de un lenguaje de descripción de interfaces estandarizado y agnóstico al lenguaje de programación se cumple con la especificación OpenAPI (OAS). La iniciativa OpenAPI fue fundada en noviembre de 2015 por la colaboración de SmartBear, 3Scale, Apigee, Capital One, Google, IBM, Intuit, Microsoft, PayPal y Restlet. Esta iniciativa se formó como un proyecto de código abierto en el marco de la Fundación Linux y fue diseñada para permitir que tanto los humanos como los ordenadores exploren y entiendan las funcionalidades de un servicio RESTful sin requerir acceso al código fuente, documentación adicional o inspección del tráfico de red. OAS permite la comprensión y la interacción con el servicio remoto con una cantidad mínima de lógica de implementación, de acuerdo con un formato de descripción neutral para el proveedor.

Las ventajas más importantes de la Especificación OpenAPI son dobles. Por un lado, los beneficios empresariales que conlleva el reconocimiento de esta estandarización como un medio útil para que muchos desarrolladores creen repositorios de código abierto de herramientas que aprovechen esta habilitación. Además, cuenta con el apoyo de un grupo de líderes de la industria que contribuyen con su gran conocimiento y mentalidad, a la vez que indican estabilidad en una base de código diversa. Por otra parte, está registrado como una poderosa herramienta técnica que es, sobre todo, agnóstica en cuanto a lenguaje, y proporciona la comprensión de una API sin la participación de la implementación del servidor. Su documentación es actualizada regularmente por una amplia comunidad que proporciona, además, implementaciones de ejemplo, fragmentos de código y respuestas a consultas. De acuerdo con lo anterior, la importancia de la estandarización que ofrece OpenAPI racionaliza su adopción como lenguaje de descripción de la interfaz de la solución propuesta. El pluralismo de las diferentes fuentes de datos que necesitan ser integradas dentro de los mecanismos de interoperabilidad, en combinación con la existencia de diferentes marcos y tecnologías de las APIs existentes que ya son utilizadas por

los sistemas existentes, introducen la necesidad de una descripción uniforme de las interfaces expuestas. Además, en el intercambio de mensajes entre las diferentes Plataformas IoT juegan un papel crucial las APIs que facilitan la comunicación entre los componentes, por lo que la estandarización de su interfaz es de suma importancia para la escalabilidad e interoperabilidad de la plataforma. La arquitectura puede construirse a partir de la especificación OpenAPI, lo que permite el desarrollo de servicios complementarios y la creación de valor añadido de los datos disponibles. Además, la implementación de la plataforma dentro de un marco de arquitectura sin servidor subraya la importancia de la especificación OpenAPI como un poderoso estándar para la descripción de la interfaz de todos los servicios contemplados y expuestos por el mecanismo de interoperabilidad.

Swagger

Swagger comenzó como una especificación simple de código abierto para diseñar API RESTful en 2010. En 2015, SmartBear Software adquirió el proyecto Swagger. La especificación Swagger se donó a la fundación Linux y se renombró como OpenAPI. Actualmente Swagger es un conjunto de herramientas de desarrollo de API que permite el desarrollo a lo largo de todo el ciclo de vida de la API, desde el diseño y la documentación, hasta la prueba y la implementación. La Iniciativa OpenAPI fue creada para orientar el desarrollo de manera abierta y transparente. Desde entonces, Swagger se ha convertido en el conjunto de herramientas más popular para manejar el ciclo de vida de la API [175].

Funcionalidades marcos de interoperabilidad IoT

Basadas en los elementos anteriores se listan las conclusiones obtenidas y necesarias para crear un marco que cubra las funcionales requeridas en la presente Tesis Doctoral:

- El uso de API REST es el mecanismo principal para exponer las API de los diferentes mecanismos de interoperabilidad y la API global del marco común de interoperabilidad.
- La especificación y documentación de las APIs es un factor clave para la adopción de la tecnología, independientemente de que el software se utilice internamente o por terceros. La herramienta más madura que se ha analizado es Swagger con su especificación OpenAPI y contando con la comunidad de desarrollo más grande.
- Cualquier implementación de los mecanismos de interoperabilidad se distribuirá y se ejecutará en un servidor o un conjunto de servidores. Por

lo tanto, el marco de interoperabilidad debería proporcionar algunos mecanismos flexibles de alto nivel para definir, manejar y acceder a dichos despliegues de los mecanismos.

- Los marcos se pueden desplegar en un servidor local. No obstante, el despliegue de marcos en la nube permite administrar las diferentes plataformas conectados a los mecanismos de interoperabilidad aprovechando las características de despliegue que ofrecen los servicios ofrecidos por los proveedores cloud. Además, el uso de marcos basados en la nube permite integrar sistemas basados en despliegues en la nube con la infraestructura ofrecida por los mecanismos de interoperabilidad.
- El marco de interoperabilidad debe ofrecer ayuda, soporte y visualización a los desarrolladores y usuarios para crear aplicaciones con un valor añadido o manejar sus plataformas.
- El marco de interoperabilidad debe permitir un despliegue modular de los mecanismos y que se puedan añadir nuevos mecanismos existentes en el futuro.

Capítulo 3

Arquitectura y diseño de los mecanismos de interoperabilidad

El objetivo de este capítulo, es presentar el diseño propuesto en la Tesis Doctoral de los mecanismos de interoperabilidad en las capas Middleware y Aplicación y Servicios. Se centra en abordar el difícil objetivo de crear un ecosistema interoperable de la Internet de las Cosas (IoT) que permita la colaboración de las plataformas verticales de IoT para la creación de aplicaciones compatibles con múltiples dominios. Por lo tanto, en este capítulo se diseñan los mecanismos de interoperabilidad para permitir el descubrimiento y el intercambio de dispositivos, aplicaciones y servicios conectados a través de las plataformas IoT existentes y futuras. Sobre esta solución de interoperabilidad se ofrecerá un marco de desarrollo para el rápido desarrollo de aplicaciones IoT que tendrán compatibilidad multiplataforma. La Figura 3.1 muestra una visión general de la arquitectura propuesta.

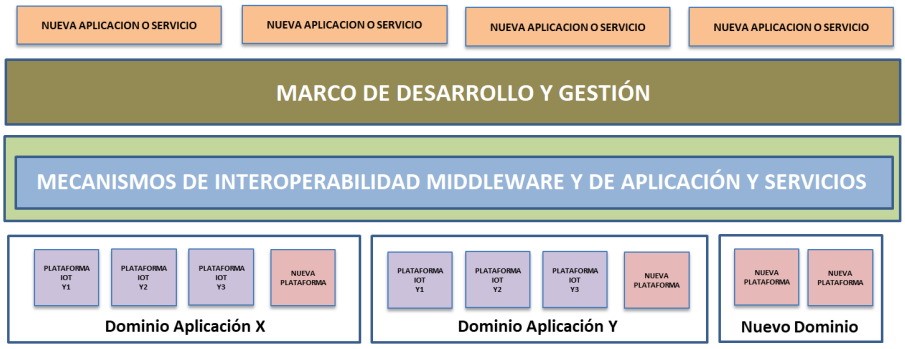


Figura 3.1: Visión general de los mecanismos de interoperabilidad diseñados.

El enfoque propuesto en la presente Tesis Doctoral es escalable. Por tanto, permite una interoperabilidad flexible y voluntaria entre diferentes capas. Este enfoque por módulos puede lograrse introduciendo un despliegue incremental de los mecanismos de interoperabilidad propuestos. Dependiendo de los componentes que se desplieguen habrá un diferente nivel de colaboración y cooperación de cada plataforma con las otras plataformas.

Esta solución propuesta no pretende crear una nueva plataforma IoT, sino una estructura de interoperabilidad para interconectar diferentes plataformas IoT, dispositivos, aplicaciones y otros elementos IoT. La interoperabilidad en las capas Middleware y Aplicación y Servicios representan los mecanismos esenciales de interoperabilidad de la presente Tesis Doctoral. Desde el punto de vista de la interoperabilidad organizativa o empresarial tiene diferentes estructuras para permitir a los proveedores de plataformas elegir un modelo de interoperabilidad adecuado para sus necesidades empresariales. Esta solución se apoyará en un marco de interoperabilidad que puede permitir el desarrollo de nuevas aplicaciones y servicios sobre de la solución propuesta.

3.1. Fundamentos de la arquitectura propuesta

El presente capitulo diseña un marco de interoperabilidad, que se fundamenta en una serie de mecanismos de interoperabilidad, entre diferentes plataformas de IoT. La mayoría de los desarrollos actuales de IoT se basan en conceptos endogámicos, centrados en un propósito específico y aislados del resto del mundo. La integración entre elementos heterogéneos suele hacerse a nivel de dispositivo o de red, y se limita a la recogida de datos. El presente diseño se centra en un enfoque multicapa de la integración de los diferentes dispositivos, plataformas,

servicios y aplicaciones de las plataformas o sistemas IoT. Permitirá una continuidad global de datos, infraestructuras y servicios. Además, se facilitará la reutilización e integración de los sistemas de IoT existentes y futuros, lo que permitirá la creación de un ecosistema global de facto de plataformas de IoT interoperables.

Cada mecanismo de interoperabilidad ofrece una API para que pueda ser consumido por los usuarios o por un marco de interoperabilidad común para todos los mecanismos. Esto ofrece un enfoque incremental y que existentes o nuevas aplicaciones y servicios puedan utilizar los mecanismos de interoperabilidad desarrollados.

La arquitectura presentada se fundamenta en la arquitectura global que se ofrece en el caso de uso INTER-IOT [41]. La definición de esta solución de interoperabilidad es la siguiente:

"*INTER-IoT presenta una solución de interoperabilidad orientada a las capas, para proporcionar interoperabilidad en cualquier capa y a través de las capas entre diferentes sistemas y plataformas de IoT. A diferencia de un enfoque global más general, un enfoque INTER-IoT tiene un mayor potencial para proporcionar interoperabilidad. Facilita una estrecha integración bidireccional, un mayor rendimiento, una completa modularidad, una gran adaptabilidad y flexibilidad, y presenta una mayor fiabilidad.*

Esta solución orientada a las capas se consigue mediante INTER-LAYER. INTER-LAYER incluye varias soluciones de interoperabilidad dedicadas a capas específicas (tal como se muestra en la Figura 3.2): Dispositivo a Dispositivo (D2D), Red a Red (N2N), Middleware a Middleware (MW2MW), Aplicación y Servicios a Aplicación y Servicios (AS2AS), Datos y Semántica a Datos y Semántica (DS2DS).

Cada capa de infraestructura de interoperabilidad tiene un fuerte acoplamiento con las capas adyacentes y proporciona una interfaz. Las interfaces serán controladas por un marco de metaniveles para proporcionar una interoperabilidad global. Se puede acceder a cada mecanismo de interoperabilidad a través de una API. Las capas de la infraestructura de interoperabilidad pueden comunicarse e interoperar a través de las interfaces. Esta estratificación cruzada permite lograr una integración más profunda y completa."

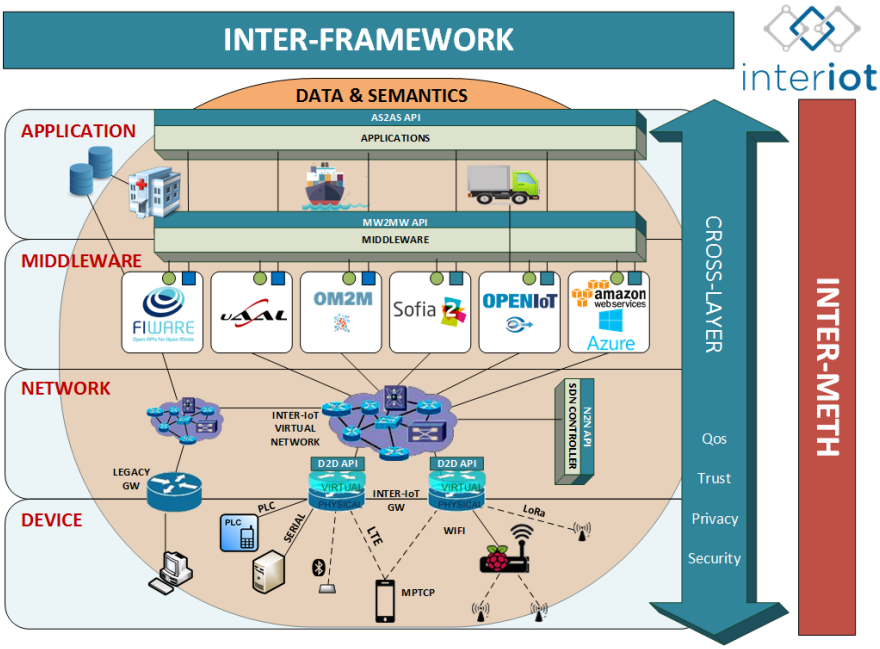


Figura 3.2: Arquitectura general del proyecto INTER-IoT.

Junto a la arquitectura global, en la presente Tesis Doctoral, se ha considerado la Arquitectura de Referencia de INTER-IoT [42], que fue diseñada para la interoperabilidad de las plataformas IoT. Una arquitectura de referencia modela los elementos arquitectónicos abstractos en el dominio de interés, independientemente de las tecnologías, protocolos y productos que se utilizan para implementar una solución específica para el dominio. Una arquitectura de referencia no es una arquitectura concreta; es decir, dependiendo de los requisitos que aborde la arquitectura de referencia, generalmente no especificará completamente todas las tecnologías, componentes y sus relaciones con suficiente detalle como para permitir la implementación directa.

El punto de partida para la instanciación de dicha arquitectura de referencia en el presente trabajo ha sido la vista funcional de la Arquitectura de Referencia de INTER-IoT que se representa en la Figura 3.3.

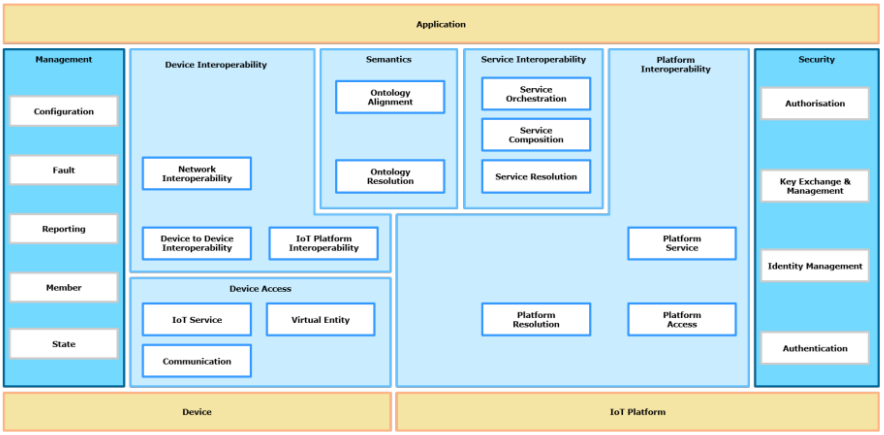


Figura 3.3: Arquitectura de referencia del proyecto INTER-IoT.

A partir de estos grupos funcionales y los componentes funcionales internos de cada grupo se han identificado algunos de ellos como relevantes para la presente Tesis Doctoral, de acuerdo con sus requisitos definidos. Los grupos funcionales que intervienen en la instanciación de la arquitectura en INTER-IoT pueden verse resaltados en la Figura 3.4.

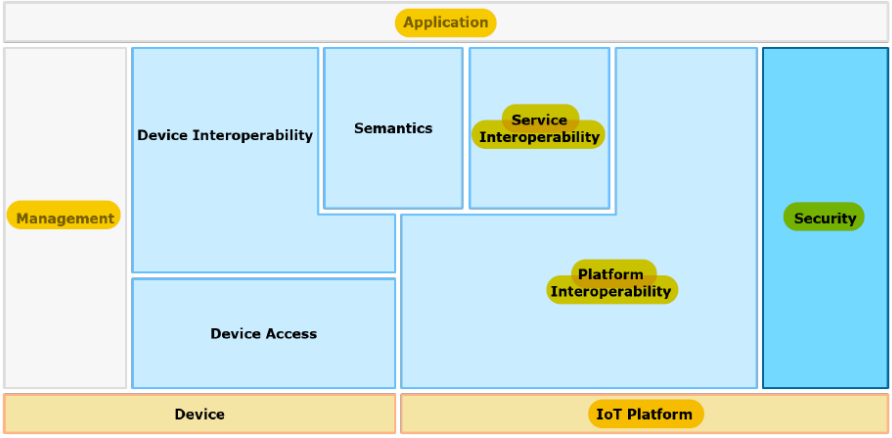


Figura 3.4: Grupos funcionales del proyecto INTER-IoT relacionados con la tesis doctoral.

La información principal de estos grupos funcionales es la siguiente:

El grupo funcional de interoperabilidad de la plataforma se utilizará para proporcionar acceso a las diferentes plataformas de IoT y realizar interacciones con la plataforma. Este grupo funcional se instanciará para interactuar con las diferentes Plataformas IoT que se interconecten. Es el responsable de acceder a las Plataformas IoT, pero no de implementar ninguna de las características que las Plataformas IoT proporcionan. Este grupo funcional tiene tres componentes funcionales:

- Resolución de la plataforma.
- Acceso a la plataforma.
- Servicio de plataforma.

El grupo funcional de Interoperabilidad de Servicios proporciona las características necesarias para acceder a los servicios de las Plataformas IoT y componer y orquestar nuevos servicios derivados entre ellas. Este grupo funcional consta de tres componentes funcionales:

- Resolución del servicio.
- Composición de servicios.
- Orquestación de servicios.

El grupo funcional de Seguridad se encarga de garantizar todos los aspectos de seguridad implicados en la interoperabilidad de las plataformas IoT. La seguridad tiene dos caras:

- La gestión de los aspectos de seguridad relacionados con la conexión con las Plataformas IoT subyacentes. Esto implica cumplir con las diferentes características de seguridad que las plataformas requieren. La solución propuesta tendrá que abordar la autenticación del usuario para conectarse a una plataforma, la autorización (por ejemplo, el uso de tokens de autenticación) y el cifrado de algunas comunicaciones.
- Gestión de la seguridad interna de la solución. La conexión a la solución debe ser segura, con capacidades de autenticación adecuadas, y la gestión de la autorización.

El grupo funcional de seguridad consta de cuatro componentes:

- Autorización
- Intercambio y gestión de claves
- Gestión de la identidad
- Autenticación

El grupo funcional de gestión se encarga de gestionar los fallos, la configuración, la administración, el rendimiento y la seguridad.

3.1.1. Visión general de la arquitectura propuesta

Partiendo de los fundamentos presentados, la presente Tesis Doctoral se centra en crear una solución extendida enmarcada en estas dos capas presentadas anteriormente en la arquitectura del proyecto INTER-IoT:

"***Capa de middleware (MW2MW):*** *En el nivel de middleware, la solución INTER-IoT permitirá un sistema de descubrimiento y gestión de recursos sin fisuras para los dispositivos IoT en plataformas IoT heterogéneas. La interoperabilidad en la capa de middleware se consigue mediante el establecimiento de una capa de abstracción y la vinculación de las plataformas IoT a la misma. Los diferentes módulos incluidos en este nivel proporcionarán servicios para gestionar la representación virtual de los objetos, creando la capa de abstracción para acceder a todas sus características e información. Entre los servicios ofrecidos, se encuentran las soluciones de interoperabilidad basadas en componentes dentro del middleware basado en la comunicación mediante mediadores, puentes y brokers. Los brokers son accesibles a través de una API general. La interoperabilidad en esta capa permitirá una explotación global de los objetos inteligentes en sistemas IoT multiplataforma a gran escala.*

Capa de aplicaciones y servicios (AS2AS)*: Este nivel de interoperabilidad pretende permitir el uso de servicios heterogéneos entre diferentes plataformas IoT. Este enfoque permitirá descubrir, catalogar y componer servicios de diferentes plataformas. AS2AS también proporcionará una API como caja de herramientas de integración para facilitar el desarrollo de nuevas aplicaciones que integren los servicios heterogéneos de IoT existentes.*"

En la presente Tesis Doctoral, estos mecanismos han sido extendidos, ampliados y desarrollados con detalle. El resultando ha dado lugar a cuatro mecanismos de interoperabilidad (Figura 3.5):

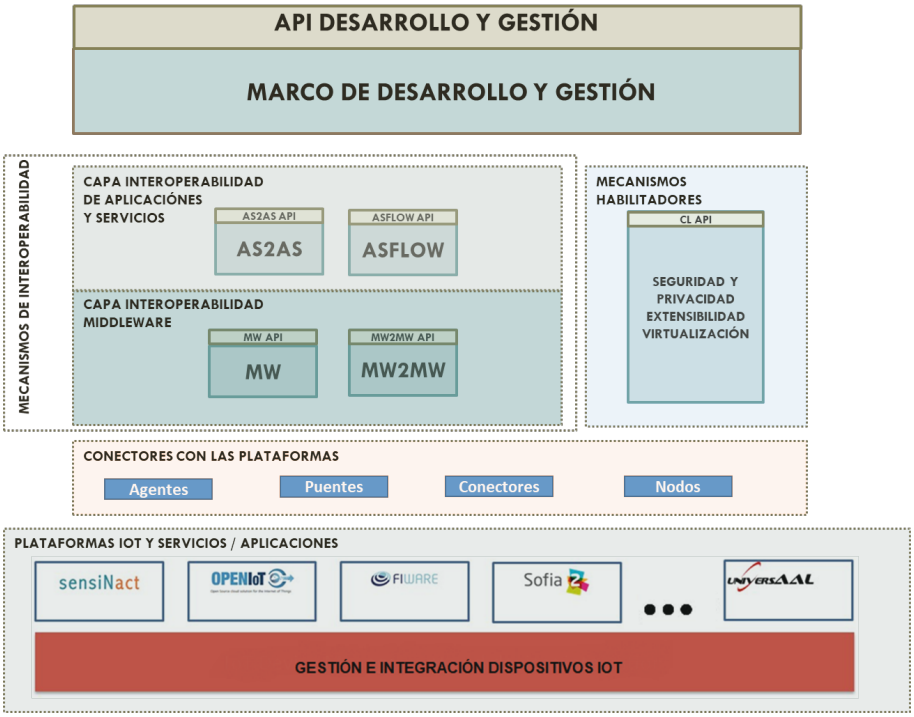


Figura 3.5: Mecanismos de interoperabilidad propuestos.

- ***Interoperabilidad Middleware (MW)***: Este mecanismo de interoperabilidad se apoya en una plataforma middleware de referencia como mecanismo principal de interoperabilidad. Unos adaptadores llamados agentes son el elemento clave para conectar las diferentes plataformas y dispositivos a este mecanismo.

- ***Interoperabilidad de Plataformas Middleware (MW2MW)***: Este mecanismo de interoperabilidad se apoya en una capa de abstracción específicamente desarrollada como mecanismo de referencia para proporcionar el componente middleware que abstrae las funcionalidades de interoperabilidad. Unos adaptadores llamados puentes son el elemento clave para conectar las diferentes plataformas y dispositivos a este mecanismo.

- ***Interoperabilidad de Aplicaciones y Servicios (AS2AS)***: Este mecanismo de interoperabilidad se puede apoyar en cualquiera de los otros mecanismos (tanto los de interoperabilidad middleware como el

de aplicaciones y servicios orientados a flujos) de referencia para ofrecer una API común que garantice definir las llamadas a los servicios y aplicaciones que hay en las plataformas conectados a este mecanismo. En este caso tanto los agentes, los puentes o los nodos (según el mecanismo básico elegido) pueden ser el elemento clave para conectar las diferentes aplicaciones y servicios disponibles en las plataformas IoT a este mecanismo de interoperabilidad.

- ***Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos (ASFLOW)***: En este caso el mecanismo de interoperabilidad se fundamenta en un motor gráfico de diseño y ejecución de flujos de servicios conectados. Se sigue el paradigma llamado programación orientada a flujos. Para acceder a los servicios y aplicaciones, se utilizan unos adaptadores llamados nodos. La conexión entre sí de estos nodos define los flujos de ejecución entre los servicios. Los nodos, además, facilitan la descripción de las funcionalidades de los servicios y aplicaciones accedidos.

Opcionalmente, estos mecanismos pueden interactuar con el mediador semántico ofrecido en el proyecto INTER-IoT para obtener facilidades de interoperabilidad semántica. Este componente se describe como:

"*La solución INTER-IoT para la capa DS2DS permitirá una interpretación común de los datos y la información entre diferentes sistemas IoT y fuentes de datos heterogéneas, logrando la interoperabilidad semántica. Se basará en la traducción semántica de las ontologías de las plataformas IoT a/desde una ontología modular IPSM común. El componente Inter Platform Semantic Mediator (IPSM) se encargará de realizar las traducciones de ontología a ontología de la información utilizando alineaciones ontológicas. Será necesario definir una semántica explícita demarcada por OWL para cada artefacto IoT que quiera interoperar, comunicarse y colaborar.*"

Además, la solución propuesta cubrirá y garantizará, mediante una serie de componentes habilitantes, los aspectos no funcionales que deben estar presentes en todos los mecanismos, como la confianza, seguridad, privacidad, extensibilidad o escalabilidad. Finalmente, se describirá un marco común de interoperabilidad que tiene como objetivo proporcionar mecanismos, herramientas y contenidos de ayuda para hacer un uso adecuado de los mecanismos de interoperabilidad aquí presentados y las APIs que estos ofrecen.

3.2. Dominios de aplicación

Aunque muchos proyectos han abordado o están abordando el desarrollo de arquitecturas de IoT en diversos ámbitos de aplicación, no son muchos los proyectos/trabajos que han abordado cuestiones de interoperabilidad/integración. Además, cuando se inició la presente Tesis Doctoral no existían propuestas para ofrecer un enfoque general, totalmente reutilizable y sistemático para resolver los múltiples problemas de interoperabilidad existentes en la tecnología de las plataformas IoT. Actualmente hay algunas iniciativas en curso, como se ha indicado en el Capítulo 2, que incluye el estado del arte presentado.

La solución propuesta permite desarrollar, de forma eficaz y eficiente, aplicaciones inteligentes de IoT, sobre diferentes plataformas heterogéneas de IoT, que abarcan dominios de aplicación únicos y/o múltiples (Figura 3.6). El objetivo general de la Tesis Doctoral es proporcionar una arquitectura marco interoperable para la integración de diferentes arquitecturas IoT presentes en diferentes dominios de aplicación. La interoperabilidad se proporcionará a diferentes niveles: middleware, servicios/aplicaciones y datos. La interoperabilidad IoT, incluso a gran escala, se creará mediante un enfoque ascendente.

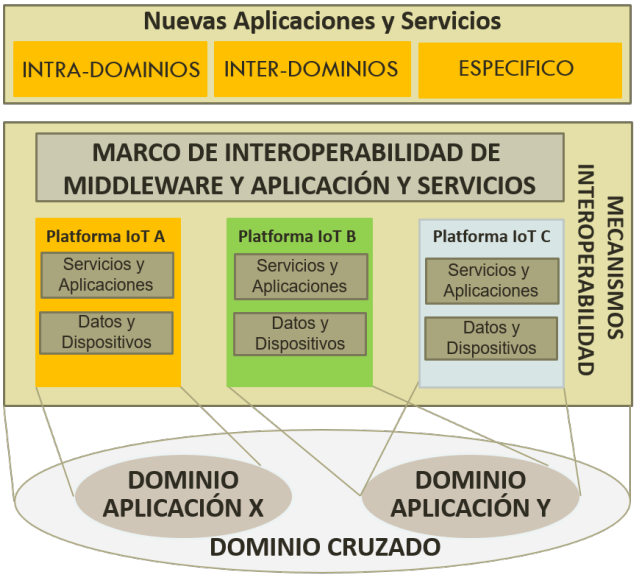


Figura 3.6: Visión general de dominios de aplicación de la solución propuesta.

En el capítulo 5 se exponen cuatro casos de uso para validar este diseño y la implementación. Estos casos de uso cubren diversos escenarios, correspondientes a dominios de aplicación como el sector portuario o de la salud. Lo más importante en este punto es destacar que la solución puede extenderse a cualquier ámbito en que sea necesario interconectar diferentes arquitecturas de IoT ya desplegadas. Es decir, el enfoque es completamente general y puede aplicarse a cualquier dominio de aplicación y a través de los dominios, en los que hay una necesidad de interconectar sistemas IoT diversificados o añadir otros nuevos. El reto es la formación de ecosistemas IoT de plataformas interoperables. Dos de los principales resultados de la Tesis Doctoral son una metodología validada para integrar, desplegar y utilizar el mecanismo de interoperabilidad desarrollado en los diferentes dominios de aplicación y una API para desarrollar nuevas aplicaciones en los dos dominios de aplicación considerados.

3.3. Requisitos

La solución propuesta está centrada en cubrir una serie de requisitos identificados organizados en diferentes grupos [176]. Estos requisitos están basados en los análisis realizados en los casos de uso definidos en los posteriores capítulos y se resumen a continuación:

Mecanismos Interoperabilidad de Plataformas IoT

Alineamiento con otras arquitecturas de IoT

Un requisito clave para la arquitectura del sistema es la alineación con los modelos arquitectónicos de referencia de otros proyectos de IoT. La finalidad es utilizar sus resultados para evitar volver a inventar un nuevo modelo arquitectónico desde cero, y estar alineado y ser compatible con esos proyectos. Por ejemplo, se debe satisfacer mediante una alineación clara con la visión de la arquitectura de AIOTI y la arquitectura utilizada en otros proyectos y organismos de estandarización citados en el estado del arte.

Independencia de la plataforma

Se debe ofrecer soporte a las plataformas actuales y futuras. La solución debe ser capaz de adherir y conectarse o comunicarse con cualquier plataforma mediante un protocolo estandarizado que pueda ser soportado por cualquier plataforma.

Compatibilidad con los protocolos comunes de comunicación de IoT

Deben admitirse los protocolos comunes de comunicación del IoT. Podrían utilizarse muchos tipos diferentes de dispositivos, por lo que no debe haber una limitación de los protocolos de comunicación admitidos.

Soporte en tiempo real

Los sensores en tiempo real deben ser soportados por el sistema. La solución debe soportar la transferencia y el procesamiento de datos en tiempo real para permitir la detección de eventos en tiempo real.

Salida en tiempo real

Las salidas del sistema deben funcionar sin retrasos apreciables. Los retrasos en tiempo real no deben interferir en el buen funcionamiento del sistema. Para que el sistema funcione en tiempo real, se permite un pequeño retraso. Sin embargo, lo ideal es que el retraso no sea perceptible.

Análisis de datos de plataformas heterogéneas

El usuario final debe poder consultar el ecosistema IoT mediante herramientas dedicadas y obtiene datos de resultados que fueron recogidos por diferentes plataformas IoT. La solución debe proporcionar medios para analizar datos de plataformas heterogéneas con diferentes representaciones de datos. El usuario final debe tener la posibilidad de consultar los datos procedentes de diferentes

plataformas IoT de forma homogénea, independientemente de la representación de datos implementada en la plataforma subyacente.

Específicos de interoperabilidad Middleware

Proporcionar conectores a los estándares de middleware

Es difícil definir y aplicar un estándar común entre todas las diversas plataformas IOT. Diferentes plataformas utilizan diferentes estándares, sin embargo, ETSI M2M, One-M2M y SSN W3C están empezando a surgir como estándares interesantes a considerar. Por lo tanto, el sistema debería soportar estos estándares.

Apoyo a la modelización semántica en la capa de middleware

La interoperabilidad semántica define las reglas para entender el significado del contenido de la información, y crea un modelo de información específico del dominio, conocido como modelo semántico.

Virtualización de objetos comunes

La virtualización de los objetos comunes del middleware puede proporcionar herramientas para facilitar la interoperabilidad en la capa de middleware.

Compatibilidad con las principales plataformas de Internet de las cosas

El componente de abstracción Middleware requiere conectores a las diferentes plataformas IoT utilizando sus APIs y su principal objetivo es hacer transparente el acceso a los servicios/plataformas. El sistema requiere conectores a diferentes plataformas IoT (Fiware, OpenIoT, OM2M, Sofia2...) para acceder a sus servicios como descubrimiento, acceso, asignación de tareas, localización, etc. La plataforma debe asegurar la conexión con las principales plataformas IoT para lograr la interoperabilidad entre ellas.

Identificador único

Cada plataforma y dispositivo conectado a la solución debe ser reconocido para poder procesar los datos hacia y desde las plataformas y sus dispositivos. No debe haber una limitación en el número de dispositivos y plataformas que pueden conectarse.

Definición semántica

Cada dispositivo, aplicación o servicio puede definirse (atributos, metadatos, etc.) mediante una ontología semántica.

Comunicación con protocolos eficientes en cuanto al tamaño de los mensajes

La comunicación debe realizarse utilizando protocolos que sean eficientes en términos de cantidad de información intercambiada sobre la cantidad de datos intercambiados medidos en bytes.

La interacción entre los puntos finales puede seguir el concepto de comunicación M2M

La comunicación puede seguir el enfoque M2M con una interacción humana mínima o nula. M2M se refiere a las soluciones que permiten a las máquinas comunicarse con los sistemas de información back-end y/o directamente con otras máquinas, con el fin de proporcionar datos en tiempo real. Las aplicaciones M2M contemplan las siguientes etapas: recogida de datos, transmisión de datos, validación de datos y respuesta a la información disponible.

Interoperabilidad entre cosas de diferentes dominios administrativos/de gestión.

La interoperabilidad puede lograrse a todos los niveles entre plataformas IoT de diferentes dominios y con diferentes políticas administrativas.

Compatibilidad con un mediador semántico para apoyar la comunicación entre plataformas

El Mediador Semántico de Plataformas IoT soporta la comunicación de plataforma a plataforma y la comunicación entre plataformas y un actor externo. Un mediador semántico de plataformas IoT debe proporcionar funcionalidad para lograr un entendimiento común de la comunicación (traducción de la semántica entre las partes de la comunicación). Dos plataformas IoT deben ser capaces de comunicarse, y un actor externo puede acceder a los datos de diferentes plataformas IoT de manera uniforme.

Interoperabilidad semántica y sintáctica

La interoperabilidad entre plataformas IoT requiere la traducción de la comunicación de los formatos de datos y la semántica. Para permitir la interoperabilidad semántica es necesario proporcionar primero la interoperabilidad sintáctica. La combinación de datos recogidos por muchas aplicaciones de IoT añade valor a los datos recogidos en su conjunto y, para facilitar estos intercambios de datos, las aplicaciones de IoT necesitan formatos de datos y APIs comunes para poder acceder a los datos y combinarlos según sea necesario. Un mediador semántico puede recibir solicitudes en formatos de datos y semántica seleccionados y traducirlos a los formatos de datos y semántica de la plataforma IoT de destino. El mediador semántico de plataforma IoT proporciona interfaces para formatos de intercambio de datos, como mínimo, como OWL, RDF, XML y JSON.

Supervisión y prestación de servicios de suscripción entre diferentes plataformas

Con el fin de prepararse para todo tipo de servicios, el sistema necesita ser capaz de monitorizar entidades físicas pertenecientes a plataformas heterogéneas de IoT y proporcionarles servicios de suscripción. El sistema puede necesitar un modelo de editor/suscriptor para algunos de los servicios ofrecidos. Por ejemplo, los sistemas de notificación o alerta deben tener un funcionamiento push para poder actuar en tiempo real.

Específicos de interoperabilidad de Aplicaciones y Servicios

Descubrimiento de servicios IoT

Los diferentes servicios puestos a disposición de los usuarios deberán ser descubribles a través de la solución, y los usuarios podrán hacer uso de las funciones de descubrimiento para conectar los servicios con los datos.

Apoyo a la coreografía de servicios y a la orquestación de servicios

La coreografía de servicios es una forma de composición de servicios en la que los servicios participantes interactúan sin ser coordinados por un componente central. El esquema de mensajería entre los servicios elementales se define desde un punto de vista global fuera de los servicios implicados. Una coreografía de servicio define el protocolo de interacción entre los servicios. Lo ideal es que las capacidades de especificación de la coreografía sean necesarias para construir soluciones distribuidas sobre el sistema. La Orquestación de Servicios es una forma de Composición de Servicios en la que la coordinación de los servicios involucrados es realizada por un componente central. Este componente se denomina orquestador y define el esquema de mensajería, necesario para cumplir con el servicio compuesto. En la Orquestación de Servicios, se crea un nuevo servicio combinando varios servicios existentes en un flujo de procesos. Por lo tanto, las capacidades de orquestación son necesarias para construir soluciones flexibles sobre el sistema.

Soporte a Mashup de servicios

Mashup introduce una forma de componer un nuevo servicio a partir de servicios existentes. Se espera que el Mashup juegue un gran papel en los entornos de IoT. Las herramientas Mashup se han propuesto como una forma sencilla de desarrollar aplicaciones mediante la composición de servicios existentes en la Web.

Soporte de servicios nativos

La plataforma diseñada debe ser capaz de trabajar y facilitar un acceso directo a los servicios nativos proporcionados por diferentes plataformas IoT (Fiware, OpenIoT, etc.).

Mapeo de servicios de sensores

Para lograr la interoperabilidad entre las plataformas IoT, es necesaria la interoperabilidad entre los estándares más destacados Las diferentes plataformas IoT se adhieren a plataformas IoT específicas sobre la ontología. Para conseguir una interoperabilidad real entre las plataformas existentes, es necesario proporcionar características de interoperabilidad que permitan el mapeo entre las diferentes provisiones de servicio utilizadas por los diferentes estándares. Como el número de estándares es demasiado grande, se presenta una lista de los mínimos esperados. Según la EPI-TF02 de IoT [140], estos estándares son:

- W3C SSN (Semantic Sensor Network Ontology).

- ETSI SAREF (Smart Appliance REFerence Ontology)
- One M2M
- Dado que W3C SSN se basa en SensorML de OGC, debería ser aconsejable apoyar también el estándar de OGC.
- FIWARE debería ser integrable, permitiendo el mapeo de servicios de sensores entre plataformas IoT y FIWARE.

Este requisito se puede lograr a través de una API para ofrecer interoperabilidad de servicios de sensores entre W3C SSN, ETSI SAREF y One M2M.

Mecanismos habilitadores

Seguridad

Se deben proporcionar las funciones de control pertinentes, como las funciones de control de los recursos de acceso y transporte: Autenticación, Autorización y Contabilidad (AAA). Los niveles de seguridad no deben permitir que terceras partes tomen el control de un sistema privado que esté funcionando sobre IoT.

Privacidad

Se debe proporcionar protección de la privacidad para acceder a la información sobre entidades físicas, servicios o plataformas conectadas o integradas en el sistema. Para mantener esta privacidad, no es posible el acceso de terceros a los datos privados o al sistema.

Seguridad del canal de comunicación

La solución debe garantizar la seguridad e integridad de los datos, mediante protocolos seguros de comunicación/transmisión (por ejemplo, SSL, SSH) que garanticen la seguridad e integridad de los datos para todas las conexiones, incluidas las de máquina a máquina.

Procedencia de los datos

Los datos proporcionados por el sistema deben ser fiables. Los metadatos de procedencia de los datos deben permitir identificar cuál es el origen de los datos (por ejemplo, qué sistema/plataforma recoge los datos). El acceso a los datos en los ecosistemas de las plataformas IoT debe permitir rastrear qué dispositivo/sistema/plataforma recogió los datos.

Escalabilidad

El sistema debe estar diseñado para ser escalable y permitir un crecimiento futuro sin problemas. La escalabilidad está relacionada con la capacidad de los sistemas para atender sin problemas una mayor demanda de recursos informáticos de datos, dispositivos, personas y aplicaciones. Por ejemplo, la escalabilidad

se realizará normalmente a través de introducir los enfoques de escalabilidad disponibles actualmente en los servicios en la nube (Amazon, Azure...).

Extensibilidad

El middleware debe ser capaz de recibir los datos de múltiples tipos de sensores, físicos o emulados. Hay que tener en cuenta la extensibilidad de todos los componentes del sistema, desde el suministro de plataformas de hardware para integrar múltiples sensores hasta el software del middleware y las infraestructuras de los bancos de pruebas. Finalmente, la solución deberá ser capaz de soportar fácilmente extensiones, actualizaciones e inclusión de nuevos módulos a medida que se van integrando.

Marco de desarrollo y gestión:

Eficiencia en el tratamiento de la información

Se debe optimizar los costes (energía, computación, tiempo, etc.). El sistema y los dispositivos deben optimizar el procesamiento de la información con respecto a una función de costes (por ejemplo, de comunicación, computación, energía, etc.).

Supervisión de los dispositivos y del sistema

El sistema debe recoger evidencias de sus elementos y sus dispositivos/plataformas conectadas para comprobar que realmente están funcionando.

API Middleware para la interoperabilidad entre diferentes plataformas

El middleware debe proporcionar una API para las comunicaciones de la capa física y los servicios necesarios para las aplicaciones. La API de acceso común es necesaria para interoperar con las diferentes plataformas de IoT utilizando sus APIs.

3.4. Componentes de la arquitectura

Las siguientes subsecciones presentan el diseño de los principales componentes de la arquitectura propuesta. En primer lugar, se presentan dos mecanismos de interoperabilidad que actúan a nivel Middleware de las plataformas. La principal diferencia entre ellos es que uno toma una plataforma existente como referencia para lograr la interoperabilidad, lo que descarga el peso del acceso en unos adaptadores llamados agentes, mientras que el otro ofrece una capa de abstracción que no proviene de ninguna solución presentada en el estado del arte y el acceso se realiza mediante unos adaptadores llamados puentes. En segundo lugar, se presentan dos mecanismos de interoperabilidad que actúan a nivel Aplicación y Servicios de las plataformas. Dichos mecanismos se diferencian en que uno pone énfasis en ofrecer una API común de abstracción para

acceder a los servicios y aplicaciones de las diferentes plataformas IoT conectadas a ellas, mientras que el otro ofrece una solución gráfica que permite diseñar flujos de interoperabilidad entre los diferentes servicios. Por último, los componentes habilitantes son los encargados de ofrecer mecanismos de seguridad, de tener en cuenta la extensibilidad y la escalabilidad y, por último, considerar las relaciones entre los componentes previamente descritos. La figura 3.7 muestra de forma visual todas las relaciones entre estos componentes y mecanismos.

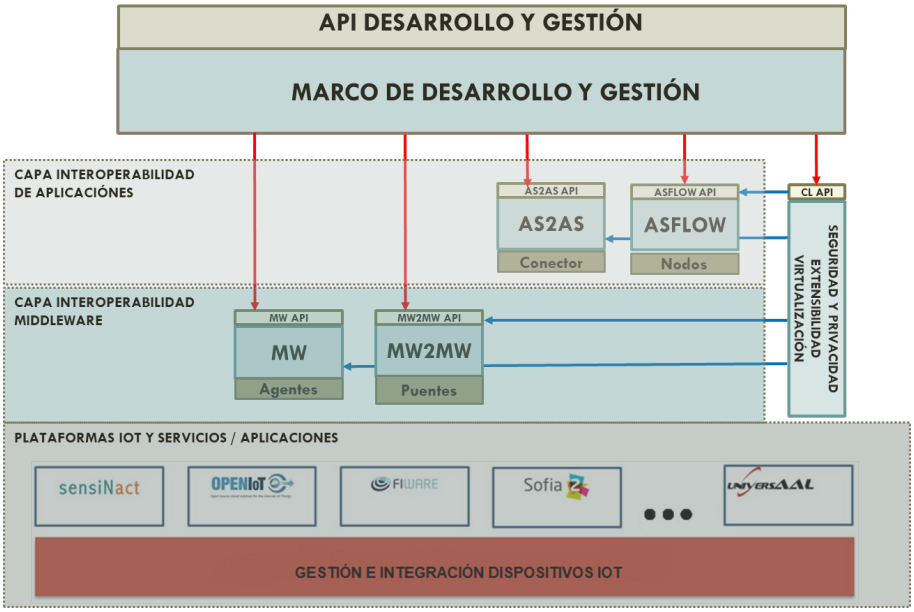


Figura 3.7: Visión general de la relación de los componentes y mecanismos de la solución propuesta.

3.4.1. Interoperabilidad Middleware

El middleware se refiere a la infraestructura de software y hardware que permite la comunicación entre los diferentes componentes del sistema, normalmente en forma de solicitud/respuesta o una comunicación de conexión sostenida para el flujo de datos.

Así, el middleware abstrae varios aspectos de una comunicación de extremo a extremo, como el nombre, la dirección y la ubicación del servicio, el protocolo de transporte de mensajes, la instancia del servicio, las características de interoperabilidad, etc. Por ejemplo, un cliente puede emitir una solicitud a un

servicio sin saber con qué instancia de ese servicio se comunicará, ocultando así algunas de las complejidades de la escalabilidad del servicio. El middleware es también una capa conveniente para la colocación de metaservicios adicionales en todo el sistema, como la seguridad, la anonimización, la auditoría y la supervisión.

En el ámbito de la IoT, se necesitan tecnologías de middleware específicas y potentes que asuman el papel fundamental de interconectar el ecosistema heterogéneo de aplicaciones que se comunican a través de varias interfaces utilizando y operando con tecnologías diversificadas. La interoperabilidad, el conocimiento del contexto, el descubrimiento y la gestión de dispositivos, la recopilación/almacenamiento/procesamiento/visualización de datos, la escalabilidad, la privacidad y la seguridad son algunos de los aspectos más importantes que deben abordar estas soluciones. El desarrollo de middlewares en el ámbito de IoT es un área activa de investigación científica e industrial, y hasta la fecha se han desarrollado varias soluciones interesantes.

Debido a la dificultad intrínseca de definir y aplicar un estándar común entre todos estos complejos escenarios, las plataformas de IoT tienen que proporcionar una capa de abstracción y adaptación a las aplicaciones de las cosas y ofrecer múltiples servicios mediante APIs fáciles de usar, pero potentes. Sin embargo, no hay una línea de división clara entre una solución de middleware y las plataformas de IoT. La mayoría de las plataformas de IoT ofrecen algunas funcionalidades de middleware, aunque se centran en ofrecer servicios eficientes de plataforma de IoT y no en resolver problemas de interoperabilidad.

Este mecanismo de interoperabilidad propuesto (visión general en la Figura 3.8) se apoya en una plataforma middleware de referencia como mecanismo de interoperabilidad. Se compone de dos partes: la capa de acceso a datos y la capa de interoperabilidad middleware. Unos adaptadores llamados agentes son el elemento clave para conectar las diferentes plataformas y dispositivos a este mecanismo.

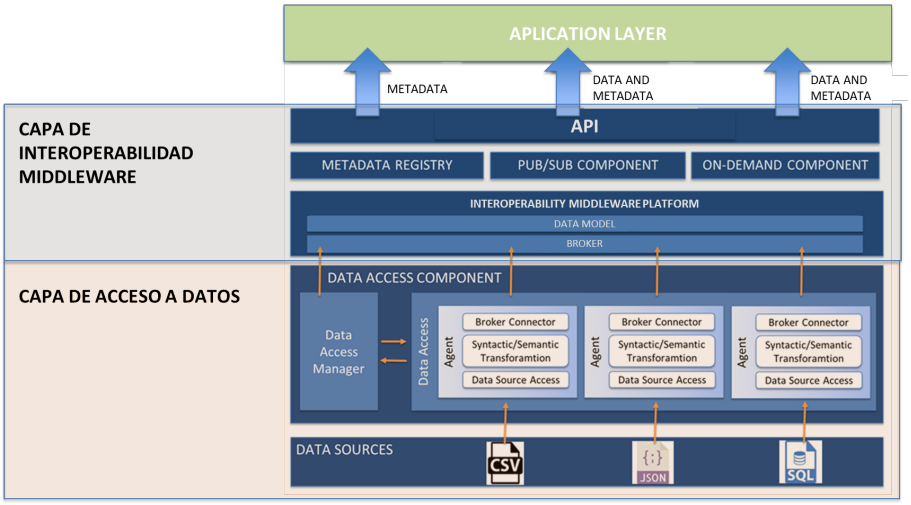


Figura 3.8: Visión general del mecanismo interoperabilidad Middleware propuesto.

El Componente de Acceso a Datos proporciona las herramientas para construir conexiones y obtener datos de diferentes fuentes de datos y plataformas. Estas herramientas están basadas en el desarrollo del agente o conector, para incluirlos a las fuentes de datos disponibles para el resto de la plataforma. Para ello, existe un asistente que ayuda a las organizaciones a conectar sus datos a través de las plantillas de agentes proporcionadas o también a crear nuevas gracias a un SDK. El componente también permite a las organizaciones administrar estas conexiones a través de la interfaz de usuario, para que puedan monitorear el estado y los eventos que ocurren en tiempo real. Para poder obtener estos datos y enviarlos a la solución, el Componente de Acceso a Datos implementará un agente para cada una de las fuentes, especificando todo lo necesario para extraer la información y hacer que los datos estén disponibles para el resto de la plataforma en un formato común. El Componente de Acceso a Datos trabaja en conjunto con el Componente de Interoperabilidad Middleware para posibilitar que se pueda acceder a los datos sin conocer el formato o la tecnología en los que fueron almacenados en cada una de las fuentes originales. El Componente de Acceso a Datos conectará y obtendrá los datos de las fuentes y luego los transformará siguiendo los modelos y reglas definidos por la interoperabilidad middleware.

El Componente de Interoperabilidad Middleware expone una API unificada para acceder a los datos de las diferentes fuentes de datos conectadas a la

solución, proporcionando tanto datos históricos o por lotes, como datos en tiempo real. Los datos pueden ser consumidos mediante peticiones a la API o mediante la creación de suscripciones a diferentes eventos.

Este componente en colaboración con el Componente de Acceso a Datos, proporciona una solución para la interoperabilidad de diversas fuentes de datos utilizando un modelo de datos común, que se basa en elementos de modelos de datos y ontologías existentes. Para garantizar que el consumidor de datos pueda comprender la estructura y el significado de los datos, los datos de salida siguen el formato común definido, independientemente de la fuente de datos que los genere.

En cuanto a los metadatos, el Componente de Interoperabilidad Middleware obtiene información sobre las diferentes fuentes de datos del Componente de Acceso a Datos y la almacena en un registro de metadatos para que puedan ser consultados mediante la API.

3.4.2. Interoperabilidad de Plataformas Middleware

En este mecanismo de interoperabilidad, la capa de abstracción no proviene de ninguna solución IoT presentada en el Capítulo 2. En este caso está compuesta por una serie de componentes que se apoyan en un broker (basado en los analizados en el Capítulo 2) para intercambiar los diferentes mensajes. Por tanto, esta solución no está enmarcada dentro de las funcionalidades proporcionadas por una plataforma concreta, sino que intenta ser un mecanismo que abstrae una serie de funcionalidades comunes. Dichas funcionalidades fueron identificadas en el estado del arte y obtenidas a través del análisis de los requisitos identificados.

El reto principal es unificar diferentes plataformas de IoT a través de servicios de middleware extensibles. La capa de abstracción común unifica la visión de todas las plataformas interconectadas, dispositivos y servicios (Figura 3.9). No importa qué dispositivo pertenece a qué plataforma, o qué servicio está en qué plataforma. La implementación de una interfaz REST API amplía aún más la usabilidad de esta capa de abstracción al exponer esta funcionalidad a través de una tecnología ampliamente utilizada

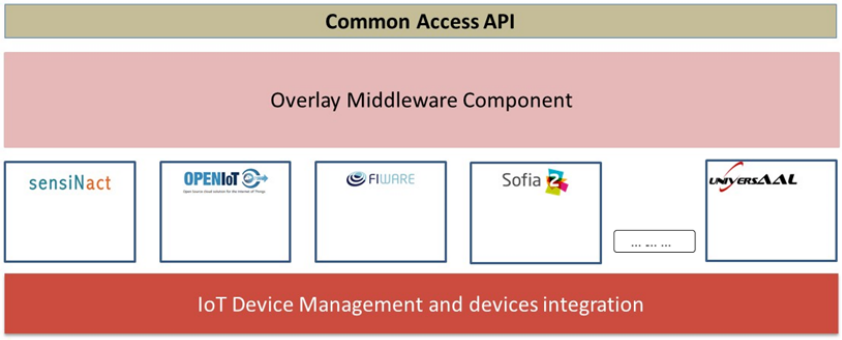


Figura 3.9: Visión general del mecanismo de interoperabilidad de plataformas Middleware propuesto.

Partiendo de un escenario en el que se ha creado un sistema compuesto por soluciones ad-hoc, este enfoque aumenta exponencialmente la complejidad (y, por tanto, los costes de desarrollo y funcionamiento) del sistema con la adición de cada nueva aplicación o plataforma. A medida que el sistema crece, se crea una compleja red de diferentes canales de comunicación entre aplicaciones y plataformas. El principal reto consiste en interconectar sistemas que nunca fueron concebidos para trabajar juntos, de una manera fácil de usar, rentable, transparente y segura. Esto plantea muchos problemas, introduce riesgos técnicos, legales, de privacidad y de seguridad, y, a menudo, supone un obstáculo insuperable para la entrega de estos sistemas innovadores. Como resultado, se puede afirmar que el desarrollo de servicios de valor añadido sobre las plataformas de la IoT es caro y requiere mucho tiempo.

La interoperabilidad de plataformas en la capa de middleware se consigue mediante el establecimiento de una capa de abstracción y el posterior acoplamiento de todas las plataformas a ella. Estas uniones se establecen utilizando puentes de capa de abstracción. De este modo, se evita la necesidad de interconectar todas las plataformas entre sí, sino que se conectan directamente a la capa de abstracción y se proporciona un mecanismo para su comunicación dentro de esta capa.

La comunicación en la capa de middleware se basa en un broker de mensajes, al que se accede en cada comunicación realizada entre plataformas. Esto permite tanto el desacoplamiento completo entre componentes como el aislamiento de la responsabilidad de la comunicación en un único elemento, lo que a su vez facilita la creación de perfiles, el escalado y la adaptación a las infra-

estructuras empresariales. Además, se accede al broker a través de una API general que expone las operaciones básicas comunes (pub/sub de mensajes, creación de temas, gestión de recursos básicos...), permitiendo el soporte de diferentes implementaciones de broker de mensajes.

Se usa un modelo de datos común para todos los componentes internos de la solución, lo que mejora la eficiencia de la transferencia interna de datos, así como también permite a los componentes hacer suposiciones sobre la estructura y el contenido de los datos, de modo que las funcionalidades específicas del dominio IoT pueden ser implementadas y ofrecidas en un modelo de datos común.

Sin embargo, no se exige el cumplimiento del modelo de datos a las plataformas conectadas. Los modelos de las plataformas que participan en la comunicación a través de la solución se elevan a ontologías y se traducen semánticamente mediante un mediador semántico (no implementado en la presente Tesis Doctoral, se ha seleccionado una herramienta desarrollada en paralelo durante uno de los proyectos como principal herramienta de soporte de las funcionalidades semánticas). Como resultado, los puntos comunes entre los modelos de datos de las plataformas IoT pueden expresarse de forma unificada (es decir, en un modelo común), a pesar de la posibilidad de tener una semántica diferente.

Por tanto, este mecanismo de interoperabilidad proporciona funcionalidades básicas relacionadas con la facilitación de la interoperabilidad entre las plataformas de middleware de IoT, así como la provisión de una capa de abstracción común para proporcionar acceso a las características e información de la plataforma de IoT. La arquitectura actúa como una capa de abstracción que unifica la visión de todas las plataformas, dispositivos y servicios interconectados. Además, su escalabilidad permite añadir fácilmente nuevas plataformas IoT. La arquitectura (Figura 3.10) de este mecanismo está compuesta por cinco componentes principales:

- Ontología: Ontología común para representar todos los mensajes encaminados a través del sistema. Desarrollada en paralelo a la presente Tesis Doctoral.
- Puentes: Actúan como intermediarios entre los mecanismos y las plataformas IoT.
- Comunicación y control: Orquesta todas las comunicaciones que se producen entre los diferentes componentes de la arquitectura.
- Servicios: Servicios comunes ofrecidos por el mecanismo de interoperabilidad para facilitar la interorabilidad entre plataformas.

- Interfaz REST: Extiende la usabilidad de la capa de abstracción exponiendo esta funcionalidad a través de una tecnología ampliamente utilizada.

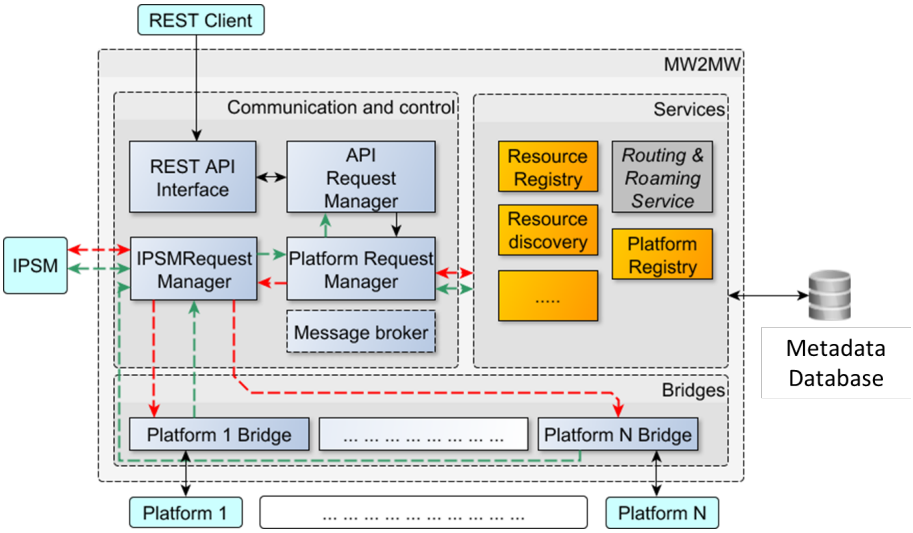


Figura 3.10: Visión detallada del mecanismo de interoperabilidad de plataformas Middleware propuesto.

La arquitectura se centra en la extensibilidad y escalabilidad de las capacidades y características. Para ello, se separan lógicamente los componentes que conforman la comunicación y el control de flujos, los servicios de middleware y los puentes a las plataformas de middleware. En la parte de comunicación, el gestor de peticiones a la API es responsable de gestionar las solicitudes recibidas del proxy de la API, lo que incluye: el mantenimiento de las sesiones activas y sus respectivas devoluciones de llamada; el reenvío de las solicitudes al Gestor de Peticiones a las Plataformas para su posterior procesamiento; y el suministro de información al llamante. El Gestor de Peticiones a las Plataformas prepara y envía las solicitudes a plataformas específicas a través de puentes, utilizando flujos de datos permanentes ya establecidos, que crea durante el arranque con la ayuda del Gestor de Flujos de Datos, o crea nuevos flujos de datos. Todos los flujos de datos que van hacia la parte inferior de la arquitectura, desde el Gestor de Peticiones de la Plataforma hasta los puentes, pasan por flujos de datos permanentes, que pueden ser encaminados a través de un mediador semántico (cuando se necesita una traducción ontológica y/o semántica, según se decida consultando el Registro y Capacidades de la Plata-

forma), o bien obviarlo y conectarse directamente a los puentes (eliminando así la sobrecarga). Todos los flujos de datos que van hacia la parte superior de la arquitectura, desde los puentes hasta el gestor de solicitudes de la plataforma, deben crearse según sea necesario.

Durante el pre-procesamiento de las solicitudes, el gestor de solicitudes de la plataforma cuenta con la ayuda potencial de algunos servicios de middleware, como el encaminamiento o el registro de dispositivos. Este envía las solicitudes a las plataformas subyacentes cuando es necesario. El Gestor de Flujos de Datos actúa como orquestador de los flujos de datos desde las plataformas (puentes) hasta el usuario que ha realizado originalmente la llamada, utilizando flujos de datos permanentes ya establecidos o creando otros nuevos y asegurándose de que todos los intermediarios están incluidos en la ruta. Por último, la Cola de Mensajes, sólo recibe y proporciona los mensajes a los componentes correspondientes, incluidos los temas temporales ad-hoc para solicitudes únicas, y los canales fijos de las plataformas.

En la parte inferior del bloque de comunicación y control, los puentes gestionan la comunicación con las plataformas subyacentes traduciendo las peticiones y respuestas desde y hacia los mensajes para la cola. Los diferentes puentes pueden necesitar utilizar HTTP, REST, sockets u otras tecnologías para comunicarse con las plataformas, pero éstas se traducirán hacia la parte superior de la arquitectura propuesta en mensajes. También pasan el contenido de los mensajes al Mediador Semántico (como se ha indicado previamente, un servicio externo a esta solución implementada), que permitirá la traducción ontológica y de formato entre las plataformas y un lenguaje común.

En el grupo de componentes de servicios, los más importantes son el Registro de Plataformas y Capacidades, que contiene la información de todas las plataformas conectadas incluyendo su tipo y capacidades de servicio; el Descubrimiento de Recursos, que crea peticiones para obtener la información necesaria de las plataformas; y el Registro de Recursos, que contiene una lista de recursos (por ejemplo, dispositivos) y sus propiedades que pueden ser consultadas rápidamente.

Finalmente, resaltar que la conversión sintáctica y la traducción semántica de los mensajes específicos de la plataforma se han definido de una forma desacoplada. La implementación de un puente para proporcionar interoperabilidad para una nueva plataforma significa, para un desarrollador de puentes, la implementación de un protocolo de comunicación con la plataforma y la traducción sintáctica del mensaje entre el formato específico de la plataforma y el formato común del mecanismo de interoperabilidad. La definición de reglas para la traducción semántica es necesaria, pero no a nivel de puente. Esa parte del proceso se define en el Mediador Semántico externo [177].

3.4.3. Interoperabilidad de Aplicaciones y Servicios

La interoperabilidad entre sistemas aplicaciones y servicios dispares es la capacidad de consumir servicios y datos mediante una capa de interoperabilidad común. Cada plataforma IoT proporciona su propia infraestructura, dispositivos, API y formatos de datos en sus servicios, lo que da lugar a problemas de compatibilidad y, por tanto, a la necesidad de especificaciones en cuanto a la interacción con otros sistemas de software. La interoperabilidad, como concepto complejo, conlleva múltiples aspectos que abordan la comunicación y coordinación efectivas entre los componentes y sistemas que pueden constituir una plataforma uniforme a mayor escala. Las APIs constituyen una herramienta de interoperabilidad que documenta todos los servicios disponibles que son expuestos por un sistema de software, así como la información sobre los respectivos protocolos de comunicación. Por lo tanto, las API se consideran un paso importante hacia la interoperabilidad de un sistema mediante la estandarización de la comunicación de los servicios. La presente sección describe la evolución de un modelo de arquitectura de interoperabilidad de servicios y aplicaciones mediante la estandarización a través de unas APIs que engloban una serie de funcionalidades comunes.

Este mecanismo se centra en un mecanismo de interoperabilidad que permite ejecutar llamadas a diferentes fuentes de datos proporcionadas por servicios o aplicaciones (como el acceso a bases de datos, APIs o archivos). Este componente se centra en las llamadas a servicios y aplicaciones de Plataformas IoT, los cuales, principalmente, devuelven como resultado una gran cantidad de datos heterogéneos. Las llamadas a aplicaciones que ejecutan una serie de acciones serán tratadas por el componente diseñado en la próxima subsección. La diferencia es que dicho componente ofrece una interoperabilidad centrada en flujos de ejecución mientras que el de esta subsección está centrado en una API de interoperabilidad común. Por tanto, volviendo al presente componente, el diseño realizado expone una API REST que permite ejecutar una serie de llamadas mediante una interfaz común a los servicios de las plataformas IoT. Esta llamada a la interfaz común, se encarga de procesar dicha llamada para poder ejecutar los diferentes servicios con los parámetros apropiados, interactuar con ellos y transformar su resultado, para, finalmente, ofrecer el resultado de la llamada en un formato estandarizado y esperado por el usuario.

El usuario dispone de un mecanismo (Figura 3.11) para abstraer las llamadas a todos los servicios y aplicaciones conectados al mecanismo de interoperabilidad, sin conocer los mecanismos propios de cada servicio y aplicación.

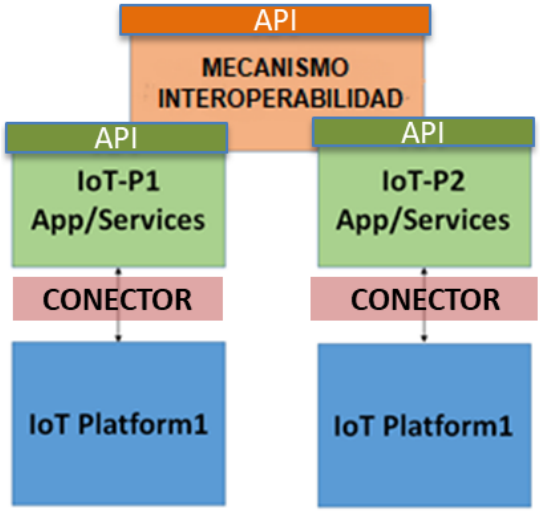


Figura 3.11: Visión general del mecanismo de interoperabilidad de Aplicaciones y Servicios propuesto.

Un Componente de Acceso a Datos resuelve el problema del acceso a diferentes fuentes u orígenes de datos de forma segura(Figura 3.12). El mecanismo tendrá que hacer frente a la variedad de fuentes de datos existente. Es necesario analizar las interfaces proporcionadas por cada fuente de datos para acceder a sus datos, entender la forma de recibir los datos, reconocer el volumen de datos, analizar el formato de los datos, considerar la existencia de una ontología en la fuente de datos y la velocidad y la veracidad de sus datos. Para hacer frente a esta heterogeneidad de fuentes de datos, es necesario definir un conector de acceso a los datos para cada fuente de datos integrada en el componente. Un conector es la pieza de software encargada de adquirir los datos de una fuente en determinadas condiciones. A continuación, los agentes transforman estos datos adquiridos de las fuentes en datos del modelo de datos común. Finalmente, estos datos se envían a los demás componentes de la plataforma. Además, el gestor de peticiones gestiona y almacena la descripción de los metadatos de los servicios procesados por los conectores. Finalmente, el gestor de servicios es el responsable de acceder a los diferentes servicios y obtener la descripción de estos, las llamadas disponibles, sus datos compartidos y su descripción mediante metadatos, con el fin de hacerlos comprensibles y disponibles para quien vaya a ejecutar la llamada. La información accedida por el gestor de servicios, es

almacenada en el registro de servicios y pasará a estar disponible en el gestor de peticiones.

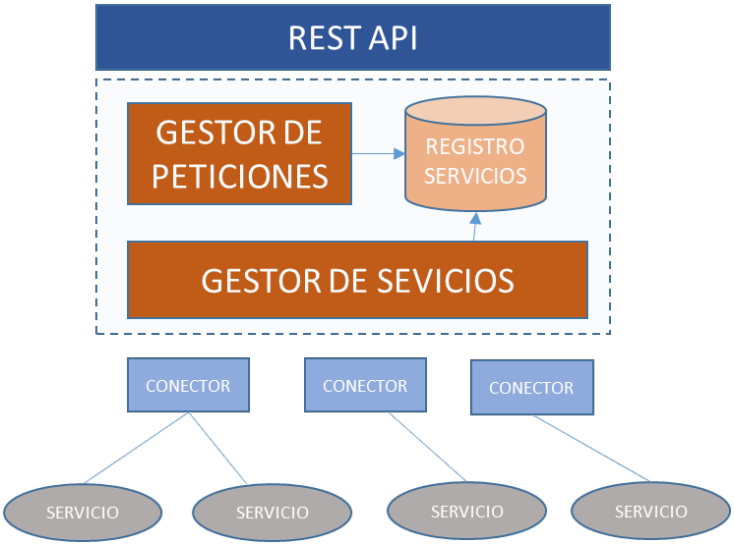


Figura 3.12: Visión detallada del mecanismo de interoperabilidad de Aplicaciones y Servicios propuesto.

En resumen, el componente de interoperabilidad proporciona una API unificada para acceder a los datos de las diferentes fuentes de datos conectadas al mecanismo de interoperabilidad (por ejemplo, proporcionando datos en tiempo real, históricos o por lotes a los consumidores de datos). En cuanto a los metadatos, el mecanismo de interoperabilidad obtiene la información sobre las distintas fuentes de datos y la almacena en el registro de servicios para que estén disponibles y accesibles para los usuarios que realizarán las peticiones.

3.4.4. Interoperabilidad de Aplicaciones y Servicios mediante Programación Orientada a Flujos

Las aplicaciones y los servicios se encuentran en la parte superior de los despliegues de IoT y representan una gran parte de la pila de IoT. Cada plataforma tiene varios servicios IoT y estos suelen estar orientados a un dominio y ser muy heterogéneos. Esta heterogeneidad dificulta la interoperabilidad entre servicios y aplicaciones de diferentes Plataformas IoT. La programación basada en flujos es un paradigma que permite la interconexión de servicios y

la creación de un flujo de ejecución entre ellos. Existen varias soluciones que utilizan este paradigma dentro de las plataformas IoT, pero ninguna se centra principalmente en el objetivo de conectar servicios de diferentes plataformas IoT. Para abordar esto, se describe un mecanismo de interoperabilidad para desarrollar funcionalidades de acceso a los servicios, y se proporciona una arquitectura con diferentes componentes para ofrecer una solución a este problema de interoperabilidad (Figura 3.13). Esta solución ofrece ventajas en el registro, catalogación y descubrimiento de servicios, y en la creación y gestión de servicios IoT compuestos.

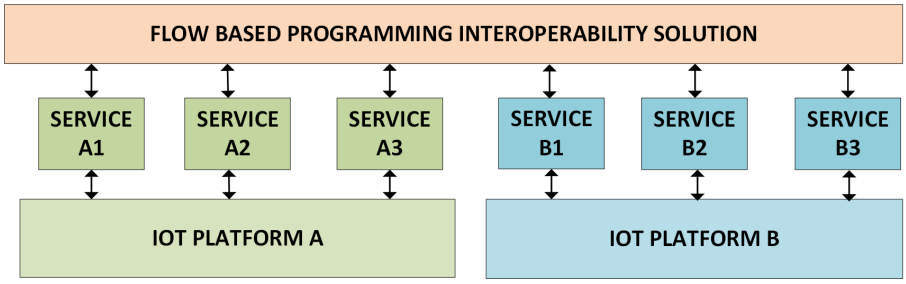


Figura 3.13: Visión general del mecanismo de interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos propuesto.

Por tanto, en primer lugar, el objetivo de este mecanismo de interoperabilidad es conseguir la identificación, registro y acceso a los servicios y aplicaciones nativas de las diferentes plataformas IoT. A continuación, se debe obtener una descripción o información detallada sobre estos servicios y aplicaciones y las especificaciones sobre cómo acceder a ellos. Además se debe ofrecer un listado de todos los servicios conectados a la capa de abstracción, permitiendo consultas o filtros de búsqueda sobre ellos.

En segundo lugar, otro objetivo de este mecanismo es hacer interoperables los servicios de aplicaciones proporcionados por plataformas heterogéneas de IoT. Debe hacer posible la reutilización y el intercambio de servicios heterogéneos de las diferentes plataformas IoT y debe permitir a los desarrolladores de aplicaciones producir nuevos servicios de valor añadido a partir de los servicios IoT existentes.

Por ello, el enfoque propuesto para de este mecanismo (Figura 3.14) de interoperabilidad se basa en:

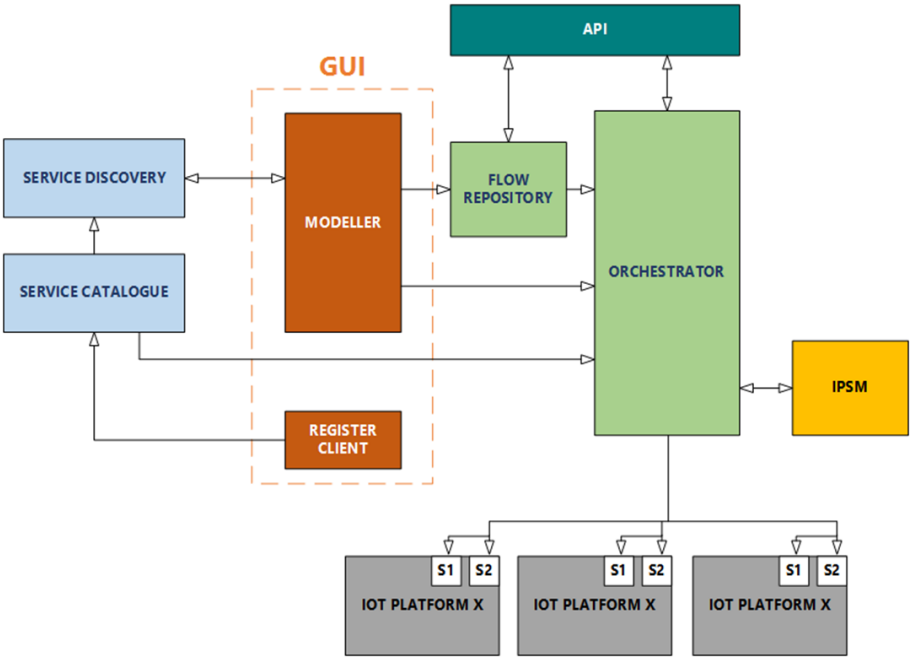


Figura 3.14: Visión detallada del mecanismo interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos propuesto.

- Acceder a las APIs proporcionadas por las plataformas IoT: Un gran número de las plataformas IoT existentes proporcionan una API pública para acceder a sus servicios. Las APIs suelen estar basadas en principios RESTful, pero en el caso de que las plataformas IoT no proporcionen esta API RESTful, es necesario crear una solución alternativa para acceder a sus servicios.
- Catálogo de servicios: Registrar los servicios/aplicaciones con su descripción o información detallada para hacerlos descubribles. Con una solución basada en el Catálogo de Servicios se podrán utilizar las mismas anotaciones de metadatos (creando así un punto de interoperabilidad) y tener un catálogo de datos uniforme.
- Descubrimiento de servicios: Crea peticiones al Catálogo de Servicios para obtener los servicios necesarios de las Plataformas IoT.

- Composición de Servicios: Engloba todos aquellos procesos que crean servicios de valor añadido, llamados servicios compuestos, a partir de los servicios existentes.

Se utilizan las siguientes soluciones para lograr el objetivo principal:

- Orquestador: La composición de servicios se realiza desde una perspectiva centralizada, expresando cómo tiene que actuar la composición para integrar los componentes.
- Flujos de datos: En una composición, la entrada a un componente viene de la salida de otro componente.
- Programación basada en flujos (FBP): FBP describe un grafo de nodos, que intercambian mensajes que contienen datos a través de las aristas. Las aristas se definen fuera de los nodos, es decir, los nodos no tienen control sobre el origen y el destino de los datos.
- Modelador: Permite arrastrar los servicios IoT y los servicios internos del catálogo y conectarlos en un entorno gráfico para luego crear flujos.

Por último, esta solución de interoperabilidad debe proporcionar una API de interoperabilidad y un marco de integración. Los usuarios y las organizaciones pueden desarrollar nuevas aplicaciones sobre la integración de sus infraestructuras heterogéneas de IoT existentes. Desde el punto de vista de la arquitectura propuesta, el Catálogo de Servicios y el Descubrimiento de Servicios, se encargan de almacenar y gestionar la información y descripción de los servicios disponibles en las plataformas IoT. Para interactuar con estos componentes, se cuenta con los módulos que forman parte del entorno gráfico (GUI): Modelador y Cliente de Registros.

El Cliente de Registros proporciona a los usuarios una herramienta para registrar nuevos servicios nativos de la plataforma IoT y nuevos servicios compuestos. Durante el registro de estos elementos es posible añadir una descripción sobre sus características. Una vez que se produce el registro del servicio, éste se almacenará en el Catálogo de Servicios.

El Modelador es un entorno gráfico que tiene acceso a los servicios que se han registrado (mediante el módulo Service Discovery, a través del cual llama al Catálogo de Servicios) y a funciones internas para ejecutar un determinado proceso (por ejemplo, funciones para realizar transformaciones en los datos resultantes de la ejecución de un servicio, para mostrar información, para determinar un tiempo de espera entre llamadas, para repetir una llamada a un servicio 'x' número de veces...). Con esta herramienta los usuarios pueden diseñar una solución basada en la composición de servicios. El editor visual

permite al usuario arrastrar y soltar los servicios (representados visualmente como nodos) en la superficie de diseño y luego unirlos arrastrando líneas entre ellos.

Una vez validado el diseño realizado por el Modelador, el flujo generado se almacena en el Repositorio de Flujos. Este componente gestiona la información de todos los flujos creados. El Orquestador es el motor de la solución. Se encarga de cargar los flujos creados con el Modelador y almacenados en el Repositorio de Flujos. Una vez cargado el diseño, realiza las llamadas necesarias a las APIs de servicios de las plataformas IoT y ejecuta las funciones internas, en el orden indicado en el modelo para ejecutar la composición de servicios. Colabora con el IPSM, que se encarga de realizar la traducción semántica de los datos intercambiados entre los servicios.

Por último, la API se encarga de poner a disposición del usuario las herramientas de interacción para gestionar el Orquestador y los flujos almacenados en el Repositorio de Flujos. Por ejemplo, sería como un gestor de procesos, donde un usuario puede iniciar/detener un flujo de ejecución, ver su estado, cargar un flujo en el orquestador, etc.

3.4.5. Componentes habilitantes

Una vez descritos los diferentes mecanismos de interoperabilidad ofrecidos, la visión pasa en estos momentos a los aspectos que quedan fuera del ámbito de estas soluciones específicas. Estos aspectos se pueden definir como elementos transversales que afectan a más de una capa y se pueden dividir en tres áreas principales: seguridad, interacciones entre capas y virtualización y clusterización de las soluciones. En primer lugar, la seguridad se refiere a los componentes ofrecidos para proteger el sistema y los datos que se manejan en ellos. Esto puede implementarse directamente dentro de las soluciones, juntas como una pieza más en el rompecabezas del software, o de forma externa y cruzada dando más resistencia a la solución desde un punto de trabajo externo. En segundo lugar, las interacciones entre capas se refieren a los módulos creados para comunicar los componentes de interoperabilidad entre sí en caso de que un despliegue necesite la información de una capa para alimentar o actuar sobre otra. En tercer lugar, la clusterización se define como la creación de conjuntos de sistemas que trabajan juntos con el mismo objetivo. Este atributo en particular es muy interesante para cuestiones de escalabilidad y para proporcionar las mismas soluciones ubicadas en el mismo cluster a diferentes inquilinos que comparten recursos. En este caso, la solución definida se apoya en la virtualización mediante contenedores para dar soporte a la clusterización.

Seguridad en capa Middleware

La solución debe tener en cuenta la seguridad a varios niveles, ya que actúa como mediador entre las plataformas y las aplicaciones. Así, la seguridad debe considerarse de la siguiente manera:

- Los componentes del middleware se comunican a través de un broker de mensajes o una plataforma IoT (según el mecanismo elegido). Esto significa que, independientemente de la implementación concreta del componente middleware de mensajes que se utilice, la comunicación entre cada componente y el componente middleware debe ser segura. Para ello, se utilizará el back-end de seguridad para generar certificados digitales de confianza para una comunicación segura.
- La seguridad de la API REST se consigue mediante el despliegue de un gestor de peticiones de la API. El gestor de peticiones de la API admite la integración con diferentes servidores de identidad/autenticación. En este caso, el gestor de solicitudes de API se integrará con el back-end de seguridad de capa cruzada.
- La autenticación de la plataforma IoT se analizará caso por caso y se implementará en puentes o agentes. Durante este proceso, cuando sea apropiado, se utilizarán mecanismos de seguridad de capa cruzada.

Seguridad en capa Aplicación y Servicios

La solución necesita aplicar la seguridad a los siguientes niveles:

- Nivel del modelador y del orquestador: Es necesario aplicar uno de estos tipos de autenticación disponibles a dicho componente gráfico:
  - Autenticación basada en credenciales de nombre de usuario/contraseña.
  - Utilizar una fuente de autenticación externa.

  Además, cada instancia debe cifrar el tráfico entre el navegador del cliente y el modelador/orquestador.
- Nivel de la API: La API está protegida por un token de acceso. Para acceder a las API, los usuarios pueden obtener un token de acceso y todas las llamadas posteriores a la API deben proporcionar este token en la cabecera de autorización.

- Nivel de nodo: Los nodos deben gestionar los mecanismos de seguridad internos para acceder a los servicios. Estos mecanismos serán los que el servicio proporcione para acceder a él y deben ser tenidos en cuenta en la creación del nodo. Por ejemplo, si se requiere un parámetro de autenticación en un servicio para acceder a él, este parámetro debe proporcionarse como parámetro de entrada del nodo. Los nodos de configuración permiten realizar estas tareas una sola vez para cada servicio.
- Nivel de flujo: También hay nodos que se desarrollan para proporcionar funciones de seguridad en la comunicación entre los elementos del flujo. Por ejemplo, nodos que encriptan y desencriptan mensajes, para crear un punto final HTTP seguro o para interactuar con las funciones de seguridad disponibles en las plataformas IoT.

Interacciones entre capas

El mecanismo de interoperabilidad del middleware expone toda su funcionalidad a través de la interfaz REST API. Cualquier punto final REST podría ser expuesto como un nodo AS2AS. El candidato más adecuado es el flujo de suscripción. Una suscripción, que proporciona una serie de observaciones de un conjunto de sensores, puede proporcionar información a los nodos consumidores, como cuadros de mando de visualización o Procesadores de Eventos Complejos (CEP) de otras plataformas IoT.

Virtualización y uso de contenedores para envolver los componentes y servicios

El el Capítulo 2 se ha descrito e indicado que se utilizarán soluciones concretas basadas en contendores de servicios para satisfacer las cuestiones relacionadas con la virtualización. Esto ofrece las siguientes ventajas:

- Despliegue rápido: Poner en marcha un nuevo recurso de plataforma o servicio IoT ocupa un tiempo considerable. Cuando las plataformas o servicios ofrecen un despliegue en un contenedor este lapso de tiempo se reduce a minutos.
- Entorno consistente: La naturaleza inmutable de las imágenes que ofrecen los contenedores proporciona un entorno inmutable para la aplicación, desde el desarrollo hasta la producción.
- Mejora de la productividad de los desarrolladores: En las fases de desarrollo de las soluciones de interoperabilidad, los programadores tienen

condiciones cercanas a trabajar con plataformas y servicios reales parecidos a los que se ofrecerán en producción.

- Portabilidad: Los contenedores se pueden mover fácilmente entre diferentes servidores destinados a tal fin.
- Múltiple tenencia: este atributo permite tener diferentes soluciones de los diferentes mecanismos de interoperabilidad. Estos inquilinos, sin embargo, no se dan cuenta de que están compartiendo los recursos y parece que tienen el sistema para ellos solos, en un contexto específico configurado.
- Aislamiento: una buena práctica de seguridad es la creación de diferentes contextos para diferentes aplicaciones. Así, si algo falla en una de ellas no se contagia a las demás haciendo que todo el sistema se caiga. Con el mecanismo de aislamiento ofrecido por los contenedores esto es posible y útil para que las aplicaciones tengan su propio espacio de datos y recursos para trabajar de forma segura y adecuada.
- Múltiples instancias de un servicio: al tener un servicio correctamente configurado y funcionando, esta propiedad permite el despliegue ágil de múltiples instancias del mismo servicio como réplica convirtiendo la solución en más escalable y resiliente (en caso de fallo de una de las instancias la solicitud puede ser redirigida a otra), además, si es necesario, las diferentes réplicas pueden ser configuradas con diversas características asociadas a cada usuario propietario.

3.5. Marco común de interoperabilidad

Los mecanismos de interoperabilidad presentados en el apartado anterior son independientes y adaptados a una solución concreta. El marco común de interoperabilidad proporciona una forma de homogeneizar los diferentes mecanismos de interoperabilidad presentados. Mediante el uso de este marco, cualquier plataforma IoT puede hacerse interoperable con respecto a sus dispositivo, middleware y capa de servicios. Esta herramienta ofrece un marco visual completo para configurar y gestionar de forma segura y desarrollar nuevas aplicaciones de software aprovechando los datos de múltiples plataformas IoT heterogéneas. Facilita la creación de un ecosistema de plataformas IoT interoperables y abiertas. Así, el tiempo de desarrollo de nuevos servicios y aplicaciones IoT puede acortarse, y estos servicios pueden proporcionarse sobre plataformas IoT interoperables. Esto se refleja en un menor esfuerzo de desarrollo y en unos costes más económicos para los propietarios de los productos, los usuarios, los desarrolladores y los integradores de plataformas. El

marco incluye componentes para abordar varios requisitos, como la seguridad, la gestión de APIs, la visualización de datos, la escalabilidad y la extensibilidad.

Los grupos funcionales cubiertos por esta herramienta son:

- Gestión de los mecanismos mediante aplicación web: Configurar, monitorizar y gestionar las diferentes capas desde una única vista. Esta solución sigue el patrón de diseño clásico modelo-vista-controlador. Incluye todos los componentes relacionados con el almacenamiento, la organización y la visualización de datos para el usuario final.
- Seguridad: Los componentes pretenden evitar el acceso a usuarios anónimos o no registrados. También evita el acceso no autorizado a los datos.
- Gestión de APIs: Proporcionar y gestionar una API de tipo REST para permitir el uso de las plataformas interoperables y las características de los mecanismos de interoperabilidad.
- Extensibilidad y escalabilidad: La finalidad es ofrecer soporte de aplicaciones actuales y futuras o escenarios más exigentes. La escalabilidad está basada en tecnologías de virtualización mediante contenedores. Esto contribuye a posibilitar el crecimiento de dispositivos y usuarios conectados a la solución. La extensibilidad se fundamenta en mecanismos que permiten agregar nuevas funcionalidades o mejoras de los servicios existentes.

Estos objetivos se abordan en los siguientes componentes funcionales diseñados:

- Marco de desarrollo de software: una combinación de resultados de software, documentación, plantillas y ejemplos que permiten la extensibilidad mediante la aplicación de los patrones de interoperabilidad de INTER-IoT en nuevas plataformas, dispositivos, servicios, etc.
- Marco y conjunto de herramientas de instanciación, configuración y gestión: una aplicación basada en la web que unifica en un único entorno la supervisión, la gestión y la configuración de los diferentes componentes del IoT (sensores, puentes, plataformas, servicios...) en cada mecanismo de interoperabilidad. Este elemento también incluye la gestión de elementos clave entre capas, como las configuraciones de autenticación o autorización.
- API común: Una API global y unificada para ofrecer un único punto de entrada a los desarrolladores de aplicaciones y servicios, ya sean propietarios de plataformas o terceros.

Capítulo 4

Implementación de los mecanismos de interoperabilidad

El objetivo del presente capítulo es describir el proceso de implementación del prototipo de la arquitectura presentada en el capítulo anterior. Se parte de una visión general de las tecnologías seleccionadas para, a continuación, ofrecer detalles específicos de cada uno de los mecanismos y elementos habilitadores necesarios.

4.1. Visión Global y Tecnologías seleccionadas para la implementación

A continuación, se listan brevemente las tecnologías seleccionadas para la implementación del prototipo de la solución propuesta en esta Tesis Doctoral. Esta implementación será instanciada, seleccionando los mecanismos y elementos habilitadores que convengan en cada caso, dentro de los casos de uso descritos en los próximos capítulos. Por tanto, se procede a presentar la implementación genérica compuesta por los siguientes elementos:

- **Mecanismos habilitadores de interoperabilidad:** Todos los componentes de cada mecanismo están Dockerizados. Además, todos los componentes ofrecen un API de acceso a sus funcionalidades. Cada API esta descrita mediante Swagger. Todas las API se ponen en común, para ser ac-

cedidas y aseguradas en único punto mediante un API Gateway (WSO2). Para cada uno de los mecanismos definidos en el capítulo anterior las tecnologías claves son las siguientes:

- **Interoperabilidad Middleware:** Se ha utilizado la plataforma IoT FIWARE como principal elemento habilitador de la interoperabilidad, concretamente su componente Orion Context Broker. En algunas implementaciones concretas se puede hacer uso de diferentes componentes suyos como STH, para almacenar información a corto plazo, o Cygnus, para garantizar la persistencia en bases de datos. Se utiliza MongoDB como base de datos para almacenar la información contextual y los metadatos y el estándar NGSI para conectar los componentes internamente e interactuar con Orion. Se han desarrollo agentes de acceso a datos mediante piezas de código en Python. Se han desarrollado diferentes componentes específicos para complementar las necesidades del mecanismo de interoperabilidad que no son cubiertas por Orion en Java, Node.js y Vue.js. Finalmente, se hace uso de un repositorio de código (Gogs) para almacenar el modelo de datos diseñado.
- **Interoperabilidad de Plataformas Middleware:** Se han implementado los componentes del mecanismo de interoperabilidad, la interfaz REST API que expone las funcionalidades y los puntos de acceso a las plataformas, mediante piezas de software en el lenguaje de programación Java. Para el intercambio de mensajes se utiliza el formato JSON-LD y un modelado de mensajes definido en el proyecto INTER-IoT. Estos componentes se apoyan en un broker para la gestión de mensajes, llamado RabbitMQ, y una base de datos semántica llamada Parliament Triple Store.
- **Interoperabilidad de Aplicaciones y Servicios:** Esta solucion necesita partir del resto de mecanismos (Fiware, solución MW2MW con RabbitMQ o Node-RED), utilizar sus respectivos registros (MongoDB, Parliament o Node-RED) y conectores (agentes, puentes o nodos). Finalmente, se implementan específicamente, y según el caso a instanciar, determinados componentes en Java o Node.js para actuar como gestor de servicios y peticiones y traductor de peticiones.
- **Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos:** Se utiliza Node-RED como motor de composición de servicios heterogéneos de plataformas IoT siguiendo el patrón de programación orientada a flujos. Se han desarrollado nodos como piezas de código HTML y Javascript. Se ha

definido los flujos con ficheros JSON. Se ha utilizado Gogs como repositorio de código de los nodos y flujos a compartir entre diferentes instancias de la solución. Se han desarrollado componentes Backend para ampliar la solución, mediante piezas de código en Java y Node.Js.

- **Componentes habilitantes:** la implementación propuesta se ha centrado en los componentes que ofrece la plataforma WSO2 relacionadas con la gestión de identidades, para tratar con los diferentes roles de los usuarios, y la centralización y gestión de las APIs ofrecidas por cada uno de los componentes. Se ha seleccionado Docker como mecanismo para el uso de contenedores para envolver componentes y servicios. Finalmente, las interacciones entre los distintos componentes, principalmente, son manejadas por sus conectores de acceso (agentes, puentes o nodos) mediante la implementación de mecanismos específicos para acceder a las APIs y realizar las transformaciones necesarias.

- **Marco de interoperabilidad común:** se ha desarrollado a través de una aplicación web que implementa la interfaz gráfica mediante Vue.js y el backend en Node.js, utilizando MongoDB como base de datos no relacional.

- **Extensibilidad de la solución:** En primer lugar, principalmente, se consigue mediante el desarrollo de nuevos conectores a nuevas plataformas, permitiendo una compatibilidad mayor con las soluciones existentes y las futuras tecnologías destacadas. En segundo lugar, mediante nuevas aplicaciones desarrolladas sobre la API, proporcionada por los mecanismos de interoperabilidad, que añaden valor añadido. Finalmente, se han aprovechado las ventajas de Docker y su API de gestión para permitir desplegar y gestionar múltiples instancias de los componentes desarrollados. Otras herramientas como Docker Swarm además permiten desplegar las soluciones en diferentes servidores huésped.

4.2. Mecanismos de interoperabilidad desarrollados

A continuación, se presentan, con todo detalle, las tecnologías seleccionadas para implementar los componentes de interoperabilidad indicados en el apartado anterior.

4.2.1. Interoperabilidad Middleware

Partiendo de la lista de plataformas presentadas en el Capítulo 2 y de un enfoque para la interoperabilidad de plataformas a través de una plataforma de IoT, se ha decidido seleccionar FIWARE como plataforma central para esta implementación.

La selección de una plataforma como eje central de la interoperabilidad es la cara opuesta a la que se presentará en la sección siguiente (que no tiene una plataforma como eje de la interoperabilidad), puesto que aprovecha todas las ventajas de una plataforma usada en diferentes empresas y proyectos, pero, por el contrario, se mantienen las carencia o desventajas que esta plataforma tenga frente a otras. No obstante, para ciertos casos de uso, en los que no se exigen ciertos requisitos, como una alta dependencia de la semántica o el uso de actuadores, adaptar una plataforma existente a las necesidades del caso de uso ahorra costes, disminuye la curva de aprendizaje y se aprovechan las ventajas de disponer de una comunidad abierta de desarrolladores manteniéndola.

Se ha seleccionado FIWARE, y concretamente su componente Orion Context Broker, por las principales funcionalidades que este ofrece. DIcho un componente permite la integración de los datos recopilados, incluidos los conocimientos para una mayor explotación. Es importante tener en cuenta que ahora Orion Context Broker es uno de los componentes básicos del catálogo digital del Connecting Europe Facility (CEF). Los componentes básicos del CEF proporcionan capacidades básicas que pueden reutilizarse en cualquier proyecto europeo para garantizar la interoperabilidad entre los sistemas de Tecnologías de la Información y facilitar la implementación de servicios a través de fronteras y sectores. Además, las especificaciones de Fiware NGSI v2, que se implementan en Orion Context Broker, son compatibles con la API estándar de administración de información de contexto (NGSI-LD) establecido por el ETSI.

Desde el punto de vista de la implementación, Orion administra los flujos de datos obtenidos de los agentes mediante su interfaz NGSI, así como también la información de los propios agentes que acceden a los datos. Los datos administrados por Orion se almacenan en MongoDB. Es importante tener en cuenta que sólo se almacena el último valor de cada entidad. Orion proporciona una interfaz de publicación / suscripción. Además, Orion gestiona el Registro de metadatos, que almacena localmente los metadatos de las fuentes de datos conectadas a él. Los datos proporcionados por Orion siguen el modelo de datos definido según el estándar de la iniciativa Smart Data Models.

Una API REST unificada expone las funciones de Orion Context Broker Esta interfaz expone una descripción Swagger integrada de la API del componente de interoperabilidad. Las interfaces cubiertas por la API común son las siguientes:

- Relacionado con el Registro de Metadatos, la API proporciona métodos para la gestión de los metadatos de las fuentes y agentes de datos. Esto incluye métodos para registrar agentes y fuentes de datos, así como métodos para enumerar / consultar / obtener los detalles de un agente o una fuente de datos, y métodos para actualizar o eliminar los metadatos.
- Relacionado con el componente bajo demanda, la API proporciona métodos para obtener los valores actuales de las entidades.
- En relación con el componente Publicar / Suscribir, la API proporciona métodos para crear suscripciones a las entidades, así como métodos de gestión de suscripciones (enumerar suscripciones, recuperar información de una suscripción, actualizar / eliminar suscripción).

En el Capítulo 8 relacionado con el caso de uso de DataPorts proporciona un ejemplo de instanciación completa de esta implementación. En la Sección 8.1, relacionado con el caso de uso Pixel, se proporciona una instanciación centrada principalmente en la plataforma Middleware de interoperabilidad.

4.2.2. Interoperabilidad de Plataformas Middleware

La funcionalidad principal de este componente de interoperabilidad es facilitar la interoperabilidad entre las plataformas de IoT Middleware (Figura 4.1), así como la provisión de una capa de abstracción común para proporcionar acceso a las características e información de la plataforma.

Para implementar este mecanismo de interoperabilidad se han tomado dos importantes decisiones de diseño:

- Modelado de datos común: El componente utiliza un modelo de datos común (definido en el proyecto INTER-IoT) para representar todos los mensajes que se dirigen a través del sistema. El formato de estos mensajes es JSON-LD.
- Abstracción del middleware. Unifica el trabajo con las plataformas IoT a través de servicios de middleware extensibles. La capa de abstracción común unifica la visión de todas las plataformas, dispositivos y servicios interconectados. No importa qué dispositivo pertenece a qué plataforma, o qué servicio está en qué plataforma. La implementación de una interfaz REST API amplía aún más la usabilidad de esta capa de abstracción al exponer esta funcionalidad a través de una tecnología ampliamente utilizada.

Para llevar a cabo las decisiones de diseño mencionadas anteriormente, se ha desarrollado una interfaz API REST por encima del Gestor de Solicitudes API. Esta interfaz API REST se encarga de todas las interfaces del mecanismo de interoperabilidad con los componentes de la capa de aplicación.

El concepto lógico del Gestor de Flujo de Datos, se ha implementado mediante la introducción de conversaciones y, por consiguiente, del control del flujo de datos. Un grupo de mensajes pertenece a la misma conversación si comparten el mismo identificador de conversación (ConversationID). Por ejemplo, en una sola conversación se tiene, típicamente, primero, un mensaje, que se suscribe a un grupo particular de sensores, y, luego, los mensajes con las lecturas de los sensores, que van hacia arriba, desde los sensores a la aplicación. Las suscripciones del componente también son rastreadas por el ConversationID. Técnicamente, los flujos de datos se implementan a través de un broker de mensajes. La mensajería tiene que ser instanciada a través de la implementación de los clientes del broker de mensajes. Básicamente, se puede utilizar cualquier corredor de mensajes que proporcione la funcionalidad básica de mensajería. Se han considerado algunos clientes para corredores de mensajes como Apache ActiveMQ, Apache Kafka, cliente MQTT genérico, Vortex OpenSplice y RabbitMQ. Todos ellos podrían ser compatibles con la solución implementada. Finalmente, se ha decidido usar el broker RabbitMQ, que era el más común en proyectos de este tipo.

La integración con la parte semántica, si es necesaria, se logra a través del componente IPSM Request Manager que orquesta la comunicación entre IPSM y los componentes del mecanismo, como los puentes o el gestor de peticiones a las plataformas.

Una base de datos semántica proporciona persistencia y mecanismos avanzados de consulta para el subsistema de Servicios. Todos los requisitos relacionados con el registro que necesitan persistencia o soporte de consulta se implementan indirectamente a través de este registro semántico. Por tanto, utilizan las funcionalidades proporcionadas por este componente el Registro de la Plataforma, el Registro de Recursos y el Registro de Suscripciones. La base de datos seleccionada es Parliament. La implementación de los servicios se ha realizado conectando éstos a la base de datos Parliament basada en triplestores e introduciendo el soporte para el lenguaje de consulta SPARQL RDF en los tipos de mensaje QUERY y DISCOVERY. SPARQL está adaptado a las consultas complejas, y la base de datos del Parlamento permite una ejecución rápida y optimizada de las mismas. Un ejemplo de este tipo de consultas complejas sería devolver una lista de todos los sensores conectados a una plataforma específica y que se encuentran en un lugar, así como los que son propiedad de una persona. Envolver tales consultas en mensajes QUERY y DISCOVERY permite a las aplicaciones del mundo exterior y a los componentes internos

obtener listas de recursos (plataformas conectadas, cosas, etc.) dinámicamente. Los mensajes QUERY y DISCOVERY también abstraen la necesidad de que los usuarios/desarrolladores entiendan los detalles de las diferentes plataformas o aprendan los mecanismos de descubrimiento que pueden (o no) soportar.

La conversión sintáctica y la traducción semántica de los mensajes específicos de la plataforma se han desacoplado. La implementación de un puente para proporcionar interoperabilidad para una nueva plataforma significa, para un desarrollador de puentes, la implementación de un protocolo de comunicación con la plataforma y la traducción sintáctica del mensaje entre el formato específico de la plataforma y JSON-LD común. La definición de reglas para la traducción semántica (alineaciones) sigue pudiéndose utilizar, pero no a nivel de puente. Esta parte del proceso está totalmente implementada en IPSM. Este componente se ha desarrollado en el proyecto INTER-IoT.

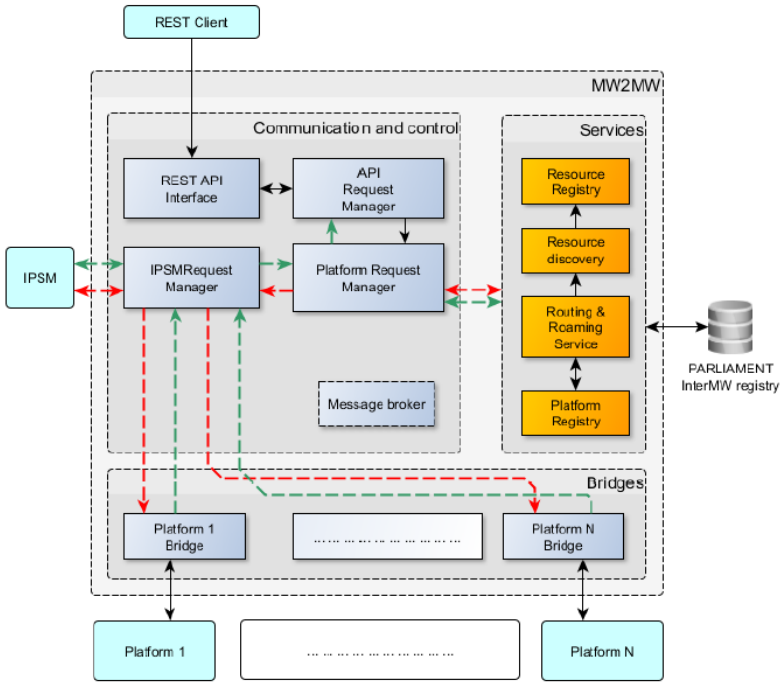


Figura 4.1: Diseño y especificación de la interoperabilidad de plataformas middleware.

Con mayor detalle, los componentes que implementan la arquitectura propuesta en el anterior capítulo, mediante piezas de código software en el lenguaje Java, son los siguientes:

Interfaz REST:

El mecanismo de interoperabilidad de Plataformas Middleware proporciona una interfaz REST API que puede ser utilizada por las aplicaciones cliente. Se ha seleccionado el lenguaje de definición de la API REST Swagger (OpenAPI) para la definición de todas las interfaces de la API REST. Para facilitar el desarrollo y mantener las definiciones actualizadas, se ha utilizado la biblioteca de anotaciones Swagger para Java para documentar la API REST. El componente se despliega como una Webapp con la interfaz REST API expuesta. Aunque en principio puede ejecutarse en cualquier contenedor de servlets web, se utiliza Jetty para fines de desarrollo y demostración.

Gestor de solicitudes de la API:

Gestiona las solicitudes recibidas del proxy de la API. Mantiene el control de las sesiones activas y sus respectivas devoluciones de llamada mediante el uso de identificadores de llamada únicos, reenviando las solicitudes al gestor de solicitudes de la plataforma para su posterior procesamiento, proporcionando información a la persona que llama a la API.

Gestor de flujos de datos:

Orquesta los flujos de datos desde las plataformas (puentes) hasta el llamador original:

- Crea un flujo de datos, asociado a un ID de llamada único.
- Crea flujos de datos permanentes entre los puentes y el Platform Request Manager, que van hacia el sur, al inicio del middleware, enrutando a través de IPSM o no, según el caso.
- Crea flujos de datos, que van hacia el norte, cuando se necesitan.

Gestor de solicitudes de plataforma:

Organiza y gestiona el flujo de peticiones a las plataformas subyacentes:

- Obtene la lista de plataformas disponibles.
- Crea un ID de flujo único, enrutamiento del flujo de peticiones a las plataformas subyacentes.

- Crea flujos de datos permanentes que van hacia el sur, enrutados a través del IPSM de DS2DS o no, según el caso.
- Crea flujos de datos hacia el norte cuando sea necesario.

Descubrimiento de recursos:

El descubrimiento de recursos es un módulo para encontrar recursos basado en una consulta que especifica los resultados deseados. Crea peticiones para obtener la información necesaria de las plataformas IoT o busca la información en el Registro de Recursos. El componente incluye métodos que permiten a los usuarios enviar una consulta para obtener la lista de dispositivos de toda la plataforma integrada a la que tiene acceso, que cumplen con una consulta o filtro de búsqueda, o consultar el Registro de Recursos. Remite la solicitud a todas las Plataformas IoT pertinentes.

Registro de recursos:

Contiene una lista de dispositivos y sus propiedades que puede consultarse rápidamente cuando se necesite. Cualquier dispositivo nuevo puede añadirse a la lista de dispositivos registrados y debe tener una identificación única. Sin embargo, no se garantiza que esté completo, por lo que debe complementarse con consultas desde el descubrimiento de recursos a las plataformas reales.

Servicio de encaminamiento y roaming:

Permite la comunicación con un determinado dispositivo independientemente de la plataforma a la que esté conectado en ese momento. Cuando un dispositivo accede a las instalaciones de una empresa diferente, se conecta de forma transparente a la plataforma. Es un método para registrar automáticamente un dispositivo en diferentes plataformas.

Registro de Plataformas y Capacidades:

Contiene la información de todas las Plataformas conectadas, incluyendo su tipo y capacidades de servicio. Se asigna un ID único a cada plataforma registrada. El componente incluye métodos para:

- Añadir (registrar) una plataforma (junto con sus servicios soportados) al registro.
- Actualizar los servicios soportados de una plataforma determinada.
- Eliminar (anular) una plataforma del registro.

- Obtener la información (servicios soportados) de una plataforma determinada.
- Generar un ID único al registrar por primera vez una nueva plataforma conectada.
- Devolver la lista de todas las plataformas conectadas (registradas)

Puente (interfaz genérica):

El puente gestiona la comunicación con las plataformas subyacentes traduciendo las peticiones y respuestas desde y hacia los mensajes para la cola. La interfaz genérica proporciona una plantilla estructurada para desarrollar fácilmente nuevos puentes. Se encargan también de:

- La traducción de peticiones desde y en mensajes para la cola.
- La estructura de código y capa de abstracción para implementaciones específicas de la plataforma.

De manera más completa los puentes ofrecen estos elementos:

- Arquitectura abierta para el desarrollo de puentes de plataformas IoT.
- Interfaz Java común. Proporciona una interfaz de puente que define las características comunes de los puentes que deben implementarse: suscripciones, actuaciones, gestión de dispositivos virtuales y descubrimiento. Las anotaciones de Java se utilizan en combinación con los mecanismos de reflexión de Java para cargar dinámicamente los puentes en tiempo de ejecución.
- Conversión sintáctica. Un paso importante en el desarrollo de puentes es la implementación de un traductor sintáctico hacia/desde el formato específico de la plataforma y el RDF. Se proporcionan traductores sintácticos genéricos con ejemplos y algunos formatos comunes, pero en principio debería proporcionarse un nuevo traductor para cada tipo de plataforma IoT.
- Traducción semántica. La traducción semántica (si es necesaria) la realiza el IPSM.
- Pruebas unitarias y de integración. Se proporcionan una serie de pruebas unitarias y de integración para facilitar el desarrollo de puentes. Se puede generar una serie de llamadas a la API para probar las implementaciones de los puentes.

- Ejemplos y documentación. Se proporciona documentación y ejemplos para las implementaciones, el uso de la API REST y los desarrollos posteriores.

Puente (interfaz especifica):

Implementación de un puente para una plataforma concreta. Este componente proporciona las funcionalidades para gestionar la comunicación con dicha plataforma. Como ejemplo, se pueden ofrecer los puentes a las plataformas IoT FIWARE, UNIVERSAAL o WSO2.

Abstracción de colas y flujos:

Recibe y proporciona los mensajes a los componentes correspondientes. Las implementaciones varían desde temas temporales ad-hoc para solicitudes únicas hasta canales de plataforma fijos.

Escenarios de interacción de los subcomponentes

En la Tesis Doctoral se han implementado los siguientes escenarios de este mecanismo de interoperabilidad en el que participan los subcomponentes descritos previamente.

Suscribirse a los mensajes de eventos

Los suscriptores podrán suscribirse a los temas, con el fin de ser informados de cualquier nueva información (lectura, actualización del dispositivo, etc.) relacionada con ese tema definido. Un suscriptor podrá crear una suscripción a través del sistema, lo que le permitirá recibir lo antes posible las noticias del publicador sobre cualquier asunto relevante para el tema deseado.

Para cumplir con esto se necesita considerar:

- Solicitud de suscripción al tema
- Nueva información enviada al tema
- Anulación de la suscripción al tema
- Creación de flujo

Por tanto, se maneja el flujo de solicitud de suscripción al tema mediante los componentes, tal y como se muestra en la Figura 4.2.

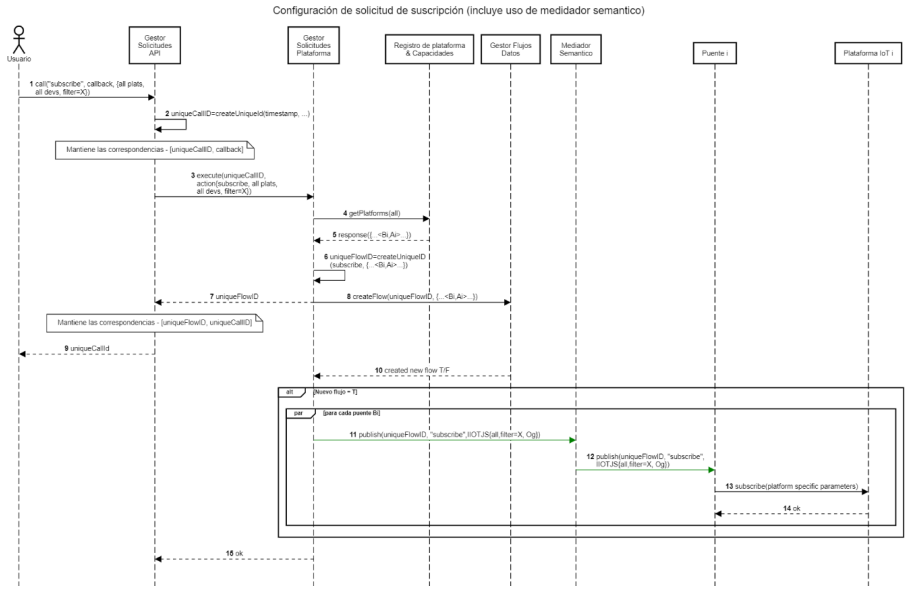


Figura 4.2: Flujo de configuración de solicitud de suscripción a plataforma.

La forma de enviar un flujo de datos de una suscripción con la información de un tema, se puede hacer usando el componente semántico como muestra la Figura 4.3.

Figura 4.3: Flujo de datos de suscripción con traducción semántica.

Por el contrario, también se puede realizar sin utilizar el componente semántico, tal y como se como muestra en la Figura 4.4.

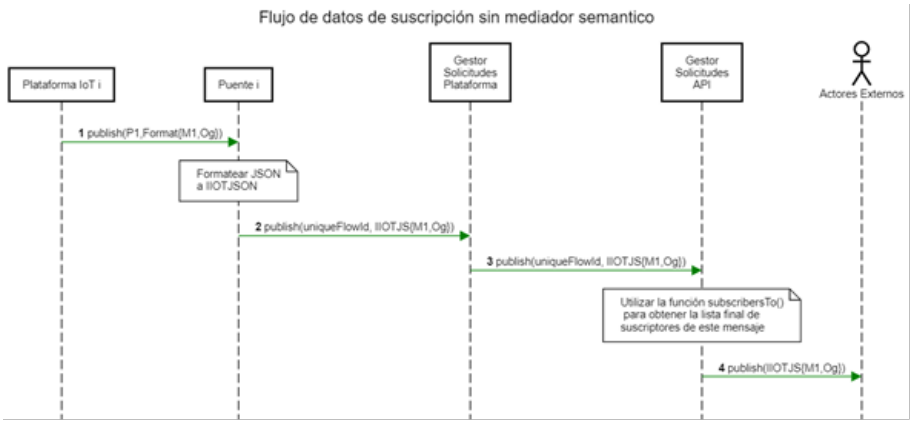


Figura 4.4: Flujo de datos de suscripción sin traducción semántica.

El flujo de la Figura 4.5 muestra cómo dar de baja una suscripción previamente creada.

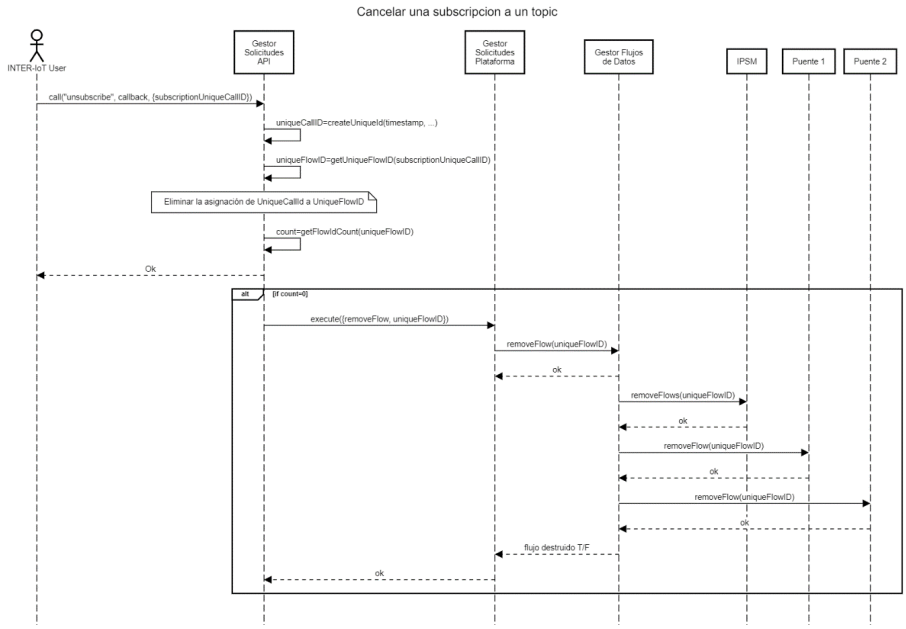


Figura 4.5: Flujo de cancelación de una subscripción.

Finalmente, se muestra como crear flujos de información utilizando el componente externo semántico en la Figura 4.6.

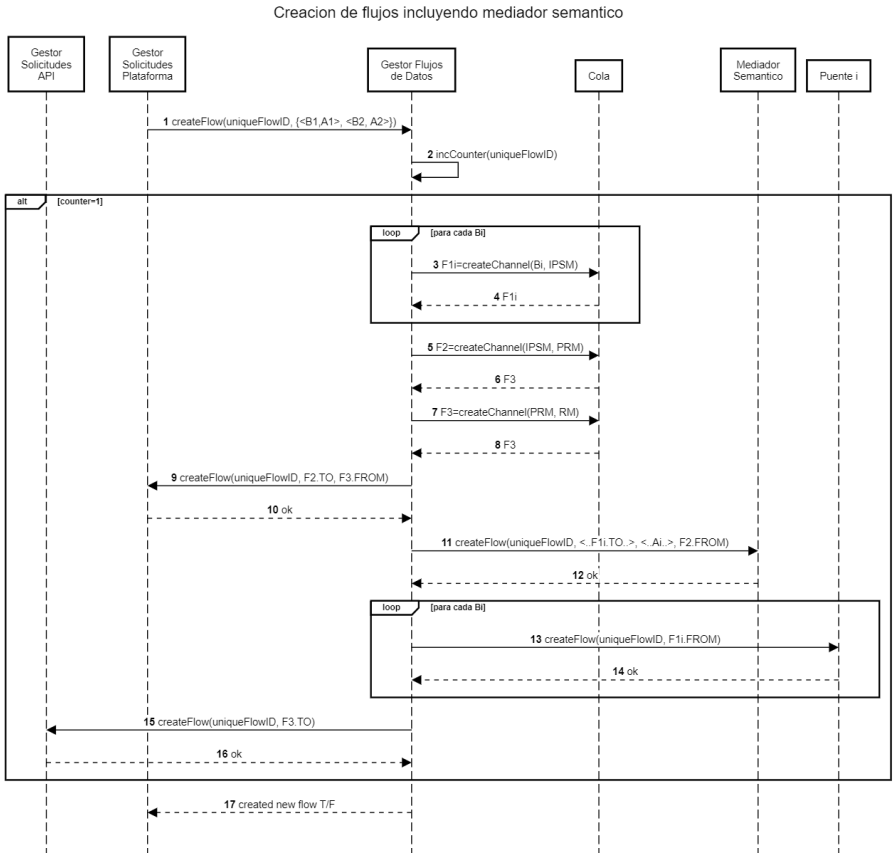


Figura 4.6: Flujo de creación de flujos de infromación mediante componente semántico externo.

Descubrimiento de recursos

Un usuario podrá obtener la lista de dispositivos de todas las plataformas integradas a las que tiene acceso, que cumplan con una consulta o filtro de búsqueda. Permite que la aplicación y los servicios descubran, siempre que sea posible, qué dispositivos, y con qué propiedades, están disponibles para el sistema implementado. La Figura 4.7 muestra cómo se lleva esto a cabo.

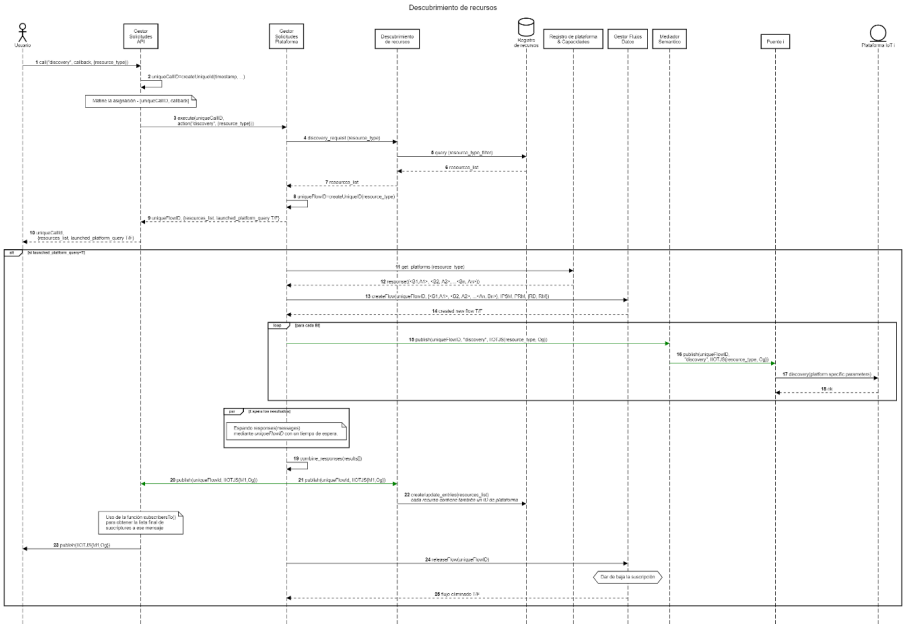


Figura 4.7: Flujo de descubrimiento de recursos.

Solicitud de consulta de información

Un usuario podrá solicitar una lista de valores de un conjunto de dispositivos de las plataformas a las que tiene acceso con las condiciones geográficas, temporales y otras definidas por el conjunto de filtros dado. Permite al solicitante obtener una lista de valores de interés de un subconjunto de dispositivos. La Figura 4.8 presenta este flujo de consulta.

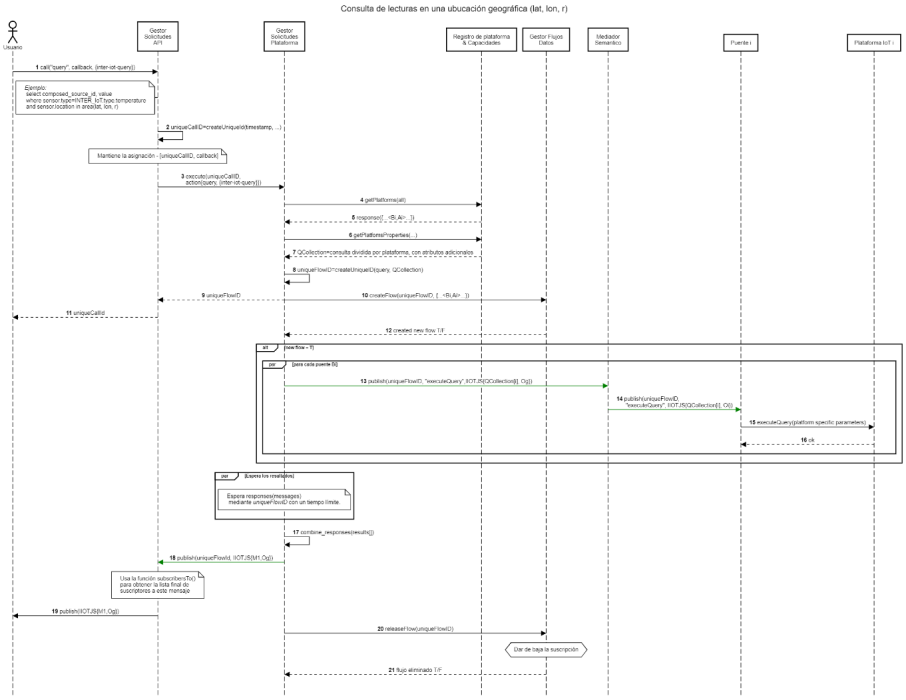


Figura 4.8: Flujo de consulta de información.

Actuación mediante el mecanismo de interoperabilidad

El mecanismo de interoperabilidad envía información a un dispositivo (sensor o actuador). El componente de interoperabilidad middleware puede acceder a un dispositivo (sensor o actuador) y enviarle órdenes o acciones (por ejemplo, cambiar la configuración, activar/desactivar), tal y como se presenta en la Figura 4.9. Para gestionar un dispositivo es necesario enviarle órdenes, además de recibir datos.

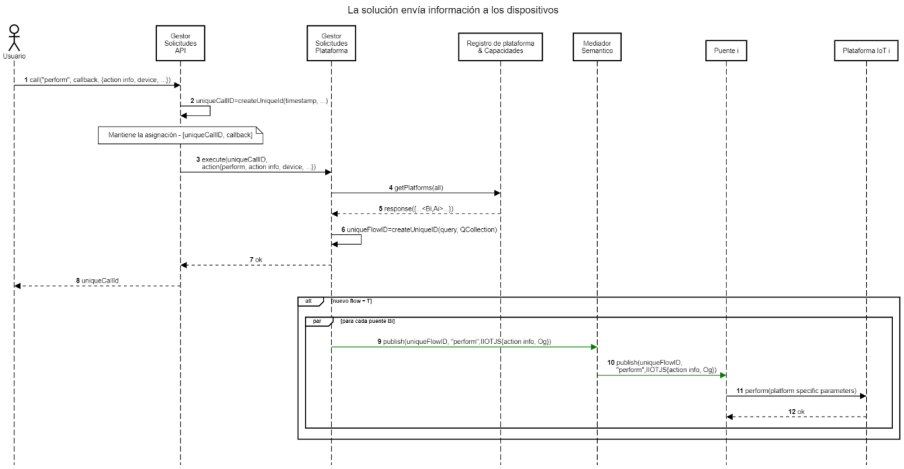


Figura 4.9: Flujo de envío de información a los dispositivos.

Finalmente, es importante indicar que en los Capítulos 6 y 7 relacionados con los casos de uso de INTER-IoT y ACTIVAGE se proporcionan ejemplos de instanciación completa de esta implementación propuesta.

4.2.3. Interoperabilidad de Aplicaciones y Servicios

Los elementos centrales para que este mecanismo de interoperabilidad funcione son el gestor de peticiones y el gestor de servicios. Para que estos componentes funcionen, es necesario implementar un componente encargado de traducir las peticiones realizadas a la API a un formato entendible por las plataformas y servicios conectados al mecanismo de abstracción de la interoperabilidad correspondiente. Este componente permite que la llamada a la API común sea entendible por las APIs o mecanismos de acceso de cada una de las plataformas. Esto implica traducir la llamada a un formato común y echar mano de los conectores de las plataformas para pedir y recibir la información en el formato común esperado.

Esta solución se implementa sobre un mecanismo de interoperabilidad de los previamente diseñados. Es decir, para que el traductor de consultas funcione, tiene que estar directamente conectado a uno de los mecanismos de interoperabilidad presentados en este capítulo. Dicho mecanismo actuará como elemento central que interactuará con la gestión de servicios y peticiones descritos en la arquitectura. La solución, por tanto, es compatible con:

- Interoperabilidad Middleware.

- Interoperabilidad de Plataformas Middleware.
- Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos.

Desde el punto de vista de la implementación, según el elemento central elegido, la implementación tendrá unas características determinadas, aunque los componentes siempre tendrán que integrar el traductor de consultas. El traductor de consultas es el elemento proporciona al desarrollador una funcionalidad central de traducir las consultas formuladas por los usuarios al mecanismo de interoperabilidad. Por tanto, el formato de datos dependerá de si el elemento central es Orion lo que implica NGSI, es MW2MW, que será JSON-LD, o si es Node-RED, que implicará JSON. Por tanto, las consultas específicas a la plataforma IoT, realizadas por el gestor de servicios y los conectores podrán ser mediante agentes, puentes o nodos respectivamente.

Interoperabilidad Middleware

En esta instanciación Orion Context Broker y MongoDB, almacenan la descripción de todos los servicios y peticiones disponibles. Los agentes interactúan con los servicios y realizan la traducción al formato común. El gestor de agentes es el encargado de ejecutar los agentes y que estos devuelvan el resultado en formato común a una dirección especificada en la llamada. La ejecución se realiza mediante llamadas con los parámetros almacenados y accesibles en Orion Context Broker. El componente bajo demanda interactúa con todos estos componentes para orquestar el flujo de acción de las llamadas y capturar posibles errores. Esta instanciación se muestra en la Figura 4.10.

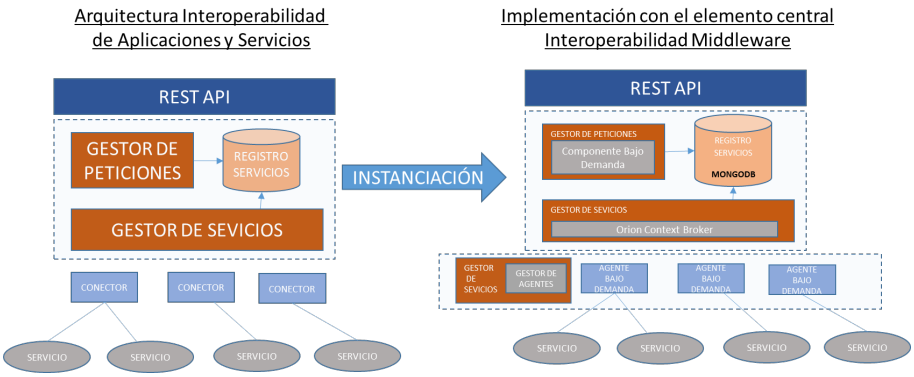


Figura 4.10: Interoperabilidad de aplicación y servicios mediante interoperabilidad MW.

El caso de Uso DataPorts, descrito en el Capítulo 8, ilustra con todo detalle un ejemplo de esta implementación.

Interoperabilidad de Plataformas Middleware

En esta instanciación la base de datos semántica Parliament es quien almacena la descripción de todos los servicios y peticiones disponibles. El componente MW2MW es el encargado de ejecutar los puentes y estos devuelvan el resultado en formato común a una dirección especificada en la llamada. La ejecución se realiza mediante llamadas con los parámetros almacenados y accesibles en Parliament y accesibles vía la API de MW2M. El traductor de peticiones interactúa con todos estos componentes para orquestar el flujo de acción de las llamadas y capturar posibles errores. Esta instanciación se muestra en la Figura 4.11.

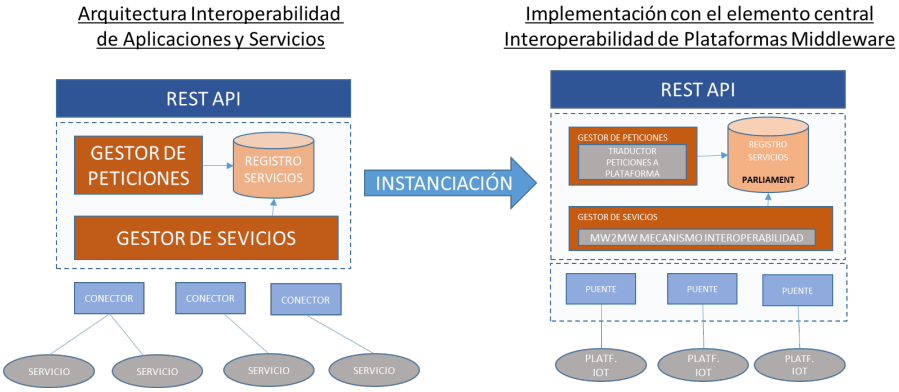


Figura 4.11: Interoperabilidad de aplicación y servicios mediante interoperabilidad MW2MW.

El caso de Uso ACTIVAGE, descrito en el Capítulo 7, ilustra un ejemplo de esta implementación.

Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos

El elemento central de esta instanciación es Node-RED, que permite crear y ejecutar el flujo de interoperabilidad. También ofrece la funcionalidad de nuevas APIs basadas en el flujo de interoperabilidad. En esta instanciación los nodos son los encargados de acceder a los servicios de las plataformas IoT. Se utilizan una serie de nodos función que son los que realizan la traducción de los datos

a los diferentes formatos. Finalmente, los flujos almacenan la llamada común que puede ser realizada por parte del usuario y parametrizada para realizar diferentes tipos de llamadas. Esta instanciación se muestra en la Figura 4.12.

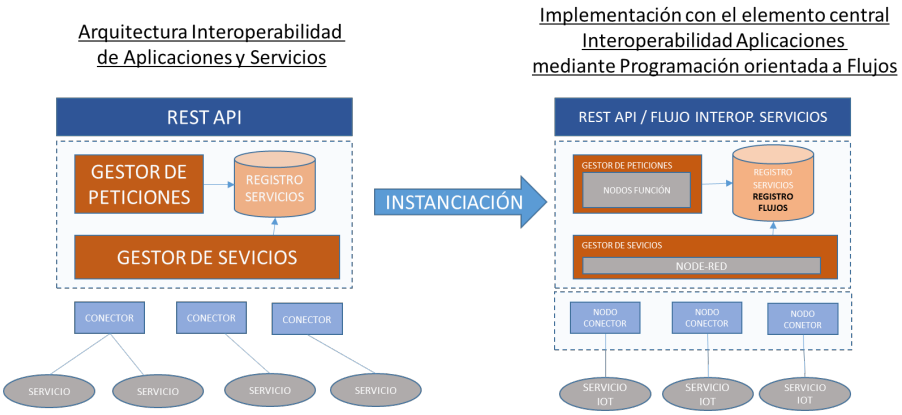


Figura 4.12: Interoperabilidad de aplicación y servicios mediante interoperabilidad AS2AS.

El caso de Uso ACTIVAGE, descrito en el Capítulo 7, ilustra un ejemplo de esta implementación.

Finalmente, destacar que la ventaja del mecanismo descrito en esta Sección respecto a los otros tres mecanismos de interoperabilidad descritos en el Capítulo, es que través de una interfaz fácil de usar, el desarrollador puede escribir una consulta de recuperación de datos, dirigiéndola al modelo de datos determinado, y traducirla a las llamadas API específicas que realiza el mecanismo de interoperabilidad seleccionado hacia las plataformas IoT disponibles. Las llamadas a la API específicas de la plataforma pueden ser útiles para el desarrollador en la programación de aplicaciones específicas de la plataforma, y el desarrollador puede llamar a estas API a través del componente de aplicaciones y servicios.

4.2.4. Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos

Se ha decidido utilizar Node-RED como tecnología central de esta solución, basándose en el análisis desarrollado en el Capítulo 2. A continuación, se listan

brevemente las funcionalidades de la arquitectura que Node-RED cubre de manera nativa:

- Servicios nativos de IoT: Ofrece una herramienta visual para conectar dispositivos de hardware, API y servicios en línea. La solución diseñada se centra en la creación de nuevos nodos para servicios de IoT, así como en el estudio de nodos existentes.
- Catálogo de Servicios: Node-RED viene con un conjunto básico de nodos y mecanismos útiles para implementar nuevos nodos personalizados.
- Descubrimiento de servicios: Los usuarios pueden buscar nodos disponibles en la biblioteca Node-RED o en el repositorio npm. Se pueden instalar nuevos nodos y los nodos existentes se pueden habilitar o deshabilitar. La solución propuesta aprovecha las ventajas que aporta el uso de metadatos para describir los nodos.
- Cliente de registro: Node-RED permite implementar nodos directamente usando el editor. La solución propuesta permite introducir metadatos al registrar un nuevo nodo.
- Modelador: Node-RED proporciona un editor de flujo basado en navegador que facilita la conexión de flujos mediante una amplia gama de nodos en la paleta. La herramienta gráfica Node-RED es una solución que proporciona las funcionalidades deseadas desde el punto de vista del modelador propuesto.
- Repositorio de flujos: Una biblioteca incorporada permite a los usuarios guardar funciones, plantillas o flujos útiles para su reutilización. Node-RED ofrece una solución para acceder y trabajar con los flujos previamente almacenados.
- Orquestador: Los flujos de interoperabilidad entre las diferentes aplicaciones y servicios se pueden desplegar en tiempo de ejecución con un solo clic. Con este elemento, los usuarios pueden realizar llamadas a los servicios en el orden deseado.
- API: Node-RED proporciona una API de administración que se puede utilizar para administrar los flujos de interoperabilidad en tiempo de ejecución y que puede ser utilizada por aplicaciones externas.

Desde el punto de vista de la implementación, se han definido los siguientes pasos:

- Desarrollo de nodos de acceso a servicios de plataformas IoT que aún no se han implementado, colaborando así activamente con la comunidad Node-RED y ofreciendo nuevos nodos que se ajustarán a las necesidades de los usuarios.
- Desarrollo de una solución sobre la plataforma Node-RED para lograr las necesidades de la arquitectura propuesta, como por ejemplo, creando nuevas soluciones en la catalogación y descubrimiento de servicios o sacando rendimiento a todas las posibilidades que ofrece la API.
- Aprovechar los beneficios que brindan los motores de contenedores (Docker), combinándolo con el uso de Node-RED.
- Utilizar las soluciones diseñadas en esta capa en ciertos casos de usos explicados en los siguientes capítulos.

Este proceso se centra en la consecución de los siguientes objetivos de la implementación:

- Introducción de nuevas plataformas y servicios.
- Estandarizar los mensajes JSON devueltos por los nodos.
- Mejorar los componentes.
- Desarrollar funcionalidades por encima de Node-RED.
- Crear nuevas funcionalidades para interactuar con nodos y flujos.
- Mejorar la interacción con la API disponible y crear una nueva API.
- Definir nuevos escenarios y demostraciones.
- Gestionar la seguridad.

Uno de los principales requisitos identificados para lograr una solución de interoperabilidad en esta capa de interoperabilidad es el acceso a servicios y aplicaciones nativos que ofrecen las plataformas IoT. La Figura 4.13 explica cómo se realiza este proceso:

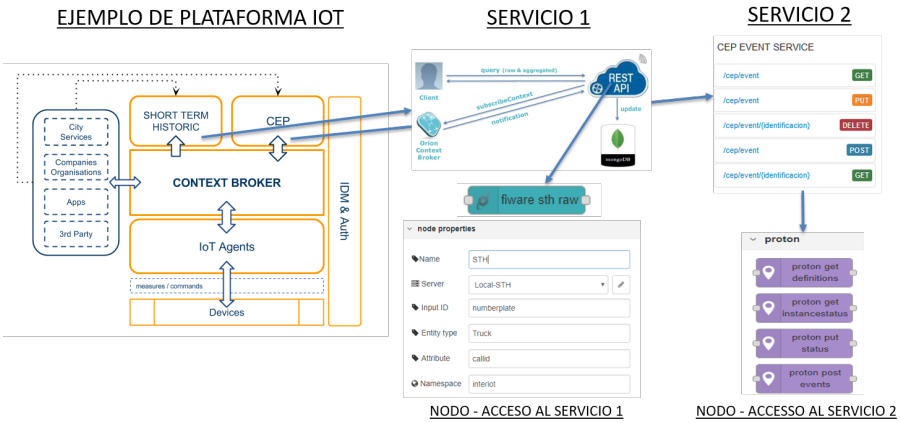


Figura 4.13: Acceso a servicios nativos mediante nodos Node-RED.

En el lado izquierdo de la imagen hay una plataforma de IoT y sus componentes. En la parte superior de la arquitectura aparecen dos servicios (en este ejemplo, un Procesador de Eventos Complejos (CEP) y un Histórico a corto plazo). La solución necesita utilizar las funciones proporcionadas por estos servicios, y, por lo tanto, es necesario estudiar los modos (REST API, SOAP ...) disponibles para acceder al servicio e implementar una solución que proporciona acceso a estas funcionalidades. Los nodos serán los elementos encargados de enviar las solicitudes, ejecutar los servicios, gestionar los resultados y la devolución de mensajes, en un formato compatible con la solución de interoperabilidad.

En el primer ejemplo (Service 1), todas las funcionalidades del servicio están contenidas en un solo nodo, en contraste al segundo ejemplo (Service 2), en el que se han utilizado cuatro nodos para acceder al servicio. Si la función es implementada por un mayor número de nodos, es más fácil comprender el propósito de las funciones, pero las actualizaciones de código son más complejas. Por otro lado, una menor cantidad de nodos para un servicio implica que hay que ingresar más parámetros en el formulario y las funciones son complicadas de entender por el usuario. Cabe destacar que, en el primer ejemplo (Service 1), el usuario tiene la necesidad de introducir información en varios parámetros para poder usar el nodo.

Prestando atención a los aspectos técnicos, un nodo consiste en un archivo JavaScript que se ejecuta en el servicio Node-RED y un archivo HTML que consiste en una descripción del nodo. La descripción aparece en el panel de

nodos con una categoría, color, nombre, un icono, código para configurar el nodo y un texto de ayuda.

Los nodos pueden tener como máximo una entrada y cero o más salidas. Durante el proceso de inicialización, el nodo se carga en el servicio Node RED. Cuando el navegador accede al editor Node RED, el código de los nodos instalados se carga en la página del editor. Node RED carga tanto HTML para el editor como JavaScript para el servidor desde los paquetes de nodos. Hay tres tipos principales de nodos: en primer lugar, los nodos de entrada que generan mensajes para los nodos descendentes; en segundo lugar, los nodos de salida que consumen mensajes, por ejemplo, para enviar datos a un servicio externo, y pueden generar mensajes de respuesta; y finalmente, los nodos de procesamiento formados por funciones que procesan los datos de alguna manera, emitiendo mensajes nuevos o modificados.

Generalmente, los nodos están diseñados para interactuar con un servicio, pero existe un tipo de nodos, denominados nodos de configuración, que son los encargados de compartir información de configuración entre los diferentes nodos que forman parte de un servicio, como por ejemplo, datos de conexión a un mismo servicio. Estos nodos no acceden al servicio, sólo se utilizan para crear y almacenar este tipo de datos de configuración

Los flujos (Figura 4.14) son una colección de nodos conectados entre sí para intercambiar mensajes, los datos contenidos en el flujo se almacenan en un archivo en formato JSON. Consiste en una lista de objetos JavaScript que describen los nodos y sus configuraciones, así como la lista de nodos a los que están conectados y los cables. Los cables definen las conexiones entre los puntos finales de entrada y salida del nodo en un flujo.

Los mensajes que se pasan entre los nodos en Node-RED son, por convención, objetos JavaScript llamados *msg*. Los mensajes son la estructura de datos principal utilizada en Node-RED y son, en la mayoría de los casos, los únicos datos con los que un nodo tiene que trabajar cuando se activa. Esto asegura que un flujo Node-RED sea conceptualmente limpio y sin estado.

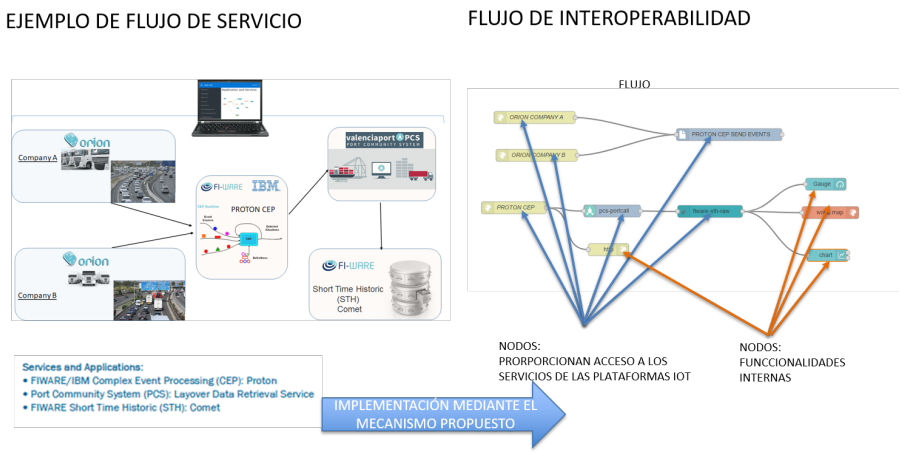


Figura 4.14: Ejemplo creación flujo de interoperabilidad mediante flujos Node-RED.

Como se ha indicado previamente, el primer objetivo de este mecanismo de interoperabilidad es la identificación y el acceso a los servicios y aplicaciones nativas de las diferentes plataformas IoT. De esta manera, se hace posible la reutilización y el intercambio de servicios heterogéneos de las diferentes plataformas IoT y se permite a los desarrolladores de aplicaciones producir nuevos servicios de valor añadido a partir de los servicios IoT existentes. Para lograr esto, en primer lugar, hay que estudiar las especificaciones sobre cómo acceder a ellos. En segundo lugar, ofrecer una descripción o información detallada sobre estos servicios y aplicaciones. Finalmente, obtener una lista de servicios en toda la plataforma integrada a la que se tiene acceso, que cumplan con una consulta o filtro de búsqueda Para lograr esto, es necesario hacer un esfuerzo para familiarizarse con los nodos, los servicios y el entorno de trabajo. Por eso se ha definido una metodología para crear nodos que implica un esfuerzo en la elaboración de documentación y código. Los pasos que siguen esta metodología son los siguientes:

- Análisis del servicio/aplicación:
  - Desplegar/acceder a una instancia funcional de la plataforma IoT.
  - Desplegar/acceder a los servicios y aplicaciones que ofrece la plataforma.
  - Desplegar /acceder a una instancia del servicio.
  - Obtener o crear una batería de datos para probar el servicio.

  - Analizar y listar funcionalidades que ofrece el servicio.
  - Estudiar el API que proporciona el servicio.
  - Documentar las funcionalidades y las API.
  - Analizar los métodos de acceso y ejecución de estas funcionalidades.
  - Analizar los mensajes o acciones que devuelve la ejecución de las funcionalidades del servicio.

- Implementación del nodo:

  - Basado en el análisis previo. Agrupar las funcionalidades del servicio, con el objetivo elegir el número de nodos necesarios y como son dichos nodos necesarios para crear el conector con el servicio/aplicación.
  - Identificar los parámetros necesarios para acceder al servicio.
  - Creación de nodos de configuración. Principalmente son aquellos que almacenan las variables de conexión.
  - Crear la interfaz que recoge los parámetros que consumirán el servicio. Parte del código HTML.
  - Crear el código que ejecutará la funcionalidad. Corresponde con la parte del código en JavaScript y Node.js.
  - Definir los mensajes que envía y recibe el nodo.
  - Si el servicio no se encuentra alojado y es necesario su despliegue. Entonces adjuntar una instancia del servicio (preferiblemente en Docker).

- Test:

  - Acceder al nodo con datos reales.
  - Probar el correcto funcionamiento.
  - Corregir bugs y detectar errores.

- Documentación

  - Documentación interna sobre cómo desplegar el servicio con datos reales, para tener ejemplos de funcionamiento.
  - Documento online sobre las características del nodo para conectar con el servicio.

- Almacenar nodo en el registro

- Almacenar el nodo en el registro correspondiente (registro corporativo o registro oficial, según licencias, para que esté disponible en la paleta)

En cuanto a los flujos, se almacenan en repositorios y se adjunta un archivo con sus características, y se describen los mecanismos (nodos de función) que actúan como pasarelas para conectar dos servicios. Se tendrá en cuenta también la parte de documentación del flujo cómo están funcionando la conexión entre servicios y el modelado del resplandor, con el fin de extraer información para definir buenas prácticas. La dockerización de la solución será explicada en un apartado posterior, pero la implementación de cada instancia de este mecanismo de interoperabilidad queda tal como se indica en la Figura 4.15.

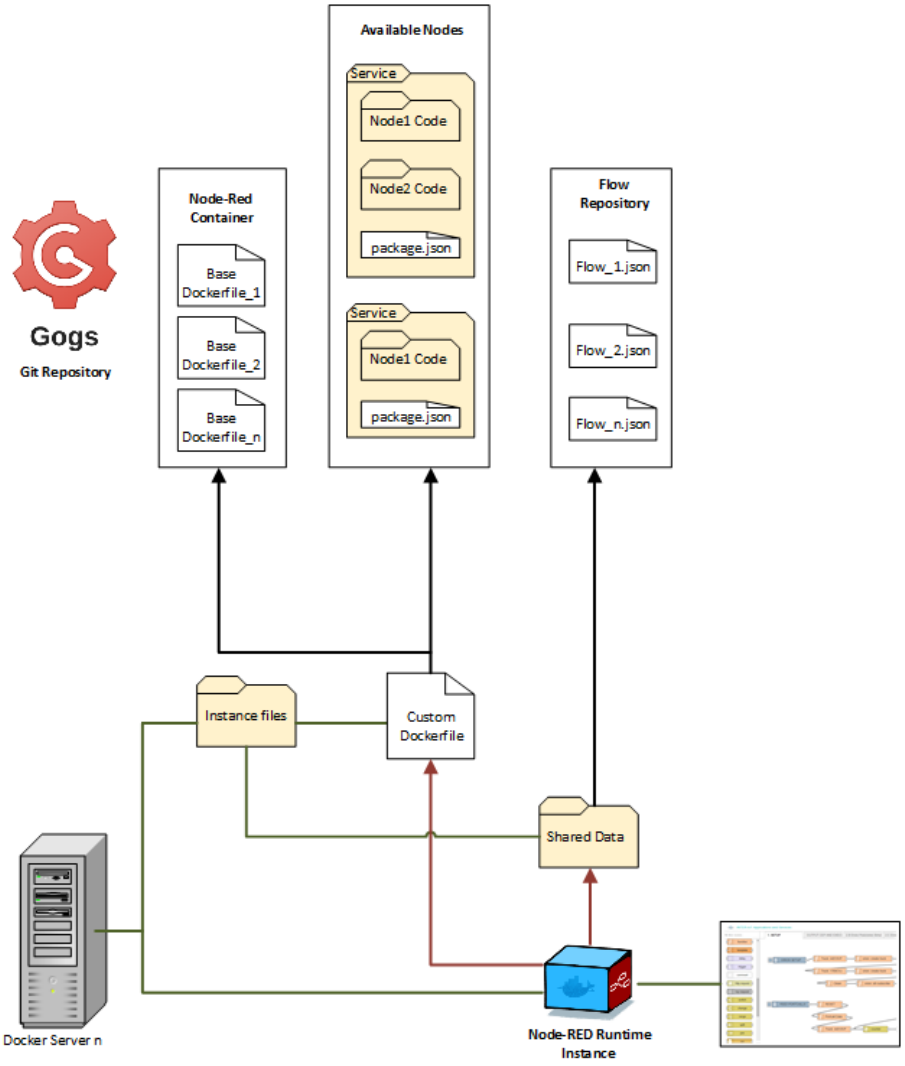


Figura 4.15: Despliegue instancia mecanismos de interoperabilidad AS2AS mediante Node-RED, Docker y Git.

Como se puede observar en dicha figura, cada instancia de la solución ofrece el motor de composición de servicios (correspondiente a la instancia activa de Node-RED), un repositorio de servicios (correspondiente al almacén de nodos

disponibles) y un repositorio de flujos de interoperabilidad (correspondiente al almacén de flujos disponible). El resto de elementos de dicha figura, tienen cubren la funcionalidad de cargar las distintas configuraciones de las instancias creadas, para lo que principalmente se apoya en ficheros de configuración de docker.

Escenarios de interacción de los subcomponentes

Se han implementado los siguientes escenarios del componente:

Catálogo de servicios

Un usuario podrá registrar servicios / aplicaciones con su descripción o información detallada para hacerlos visibles. Se persiguen principalmente los siguientes objetivos:

- Ofrecer una descripción o información detallada sobre los servicios / aplicaciones.
- El servicio registrado pasa a estar disponible en el modelador, por lo que puede ser utilizado por el orquestador.
- Obtener un catálogo de datos uniforme.

La Figura 4.16 ilustra los pasos para implementar esta funcionalidad:

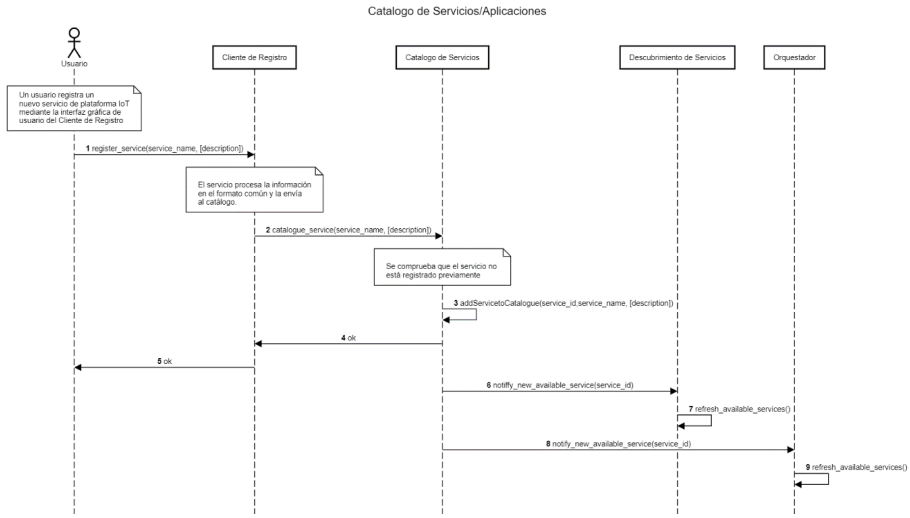


Figura 4.16: Flujo de registro aplicación mediante el catalogo de servicios.

Descubrimiento de servicios

Un usuario podrá obtener la lista de servicios disponibles en el catálogo, que cumplan con una consulta de búsqueda o filtro. El objetivo principal es obtener una lista de servicios y aplicaciones coincidentes de las plataformas de IoT consideradas.

La Figura 4.17 ilustra los pasos para implementar esta funcionalidad:

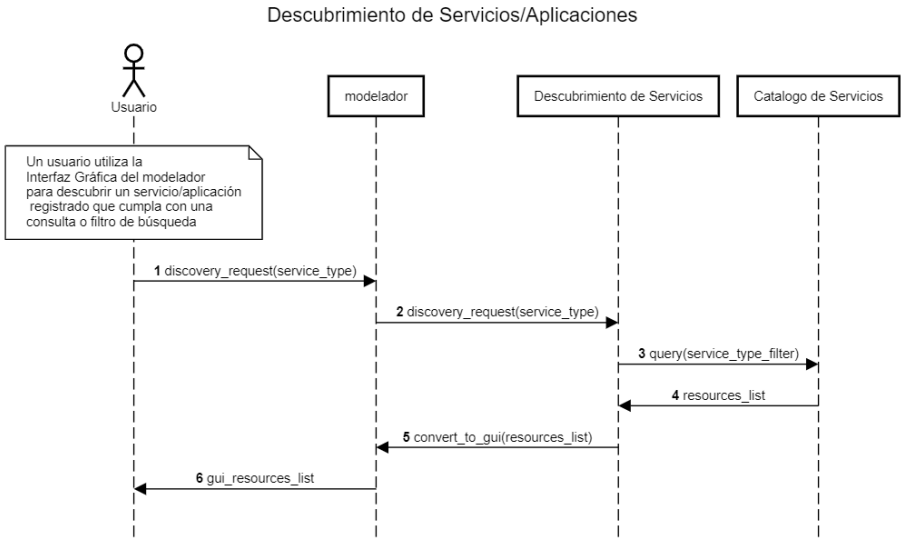


Figura 4.17: Flujo de descubrimiento de servicios.

Composición de servicios

Un usuario podrá crear servicios de valor agregado, llamados servicios compuestos, a partir de servicios existentes. El objetivo principal es conectar las APIs de los servicios y aplicaciones de las plataformas de IoT creando nuevos servicios compuestos.

La Figura 4.18 ilustra los pasos para implementar esta funcionalidad:

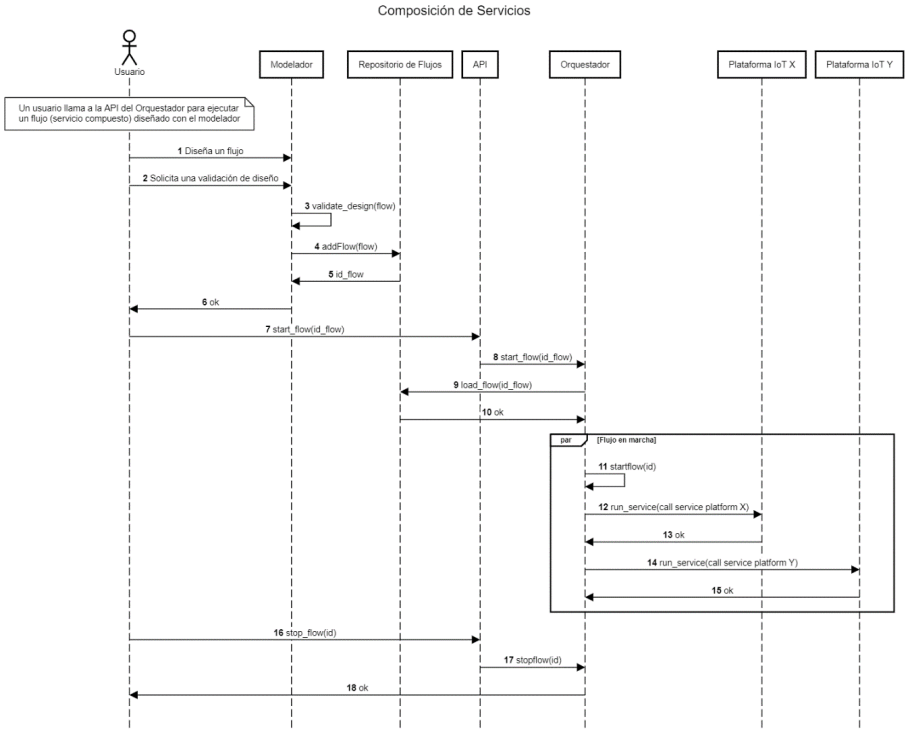


Figura 4.18: Flujo de composición de servicios.

Solicitud de consulta al mecanismo de interoperabilidad

Un usuario (plataforma de IoT, aplicación, persona, etc.) podrá intercambiar información entre los servicios / aplicaciones de la plataforma de IoT a través del sistema habilitador. El objetivo principal es permitir al consultante intercambiar información de interés con servicios y aplicaciones asignados en otra plataforma.

La Figura 4.19 ilustra los pasos para implementar esta funcionalidad:

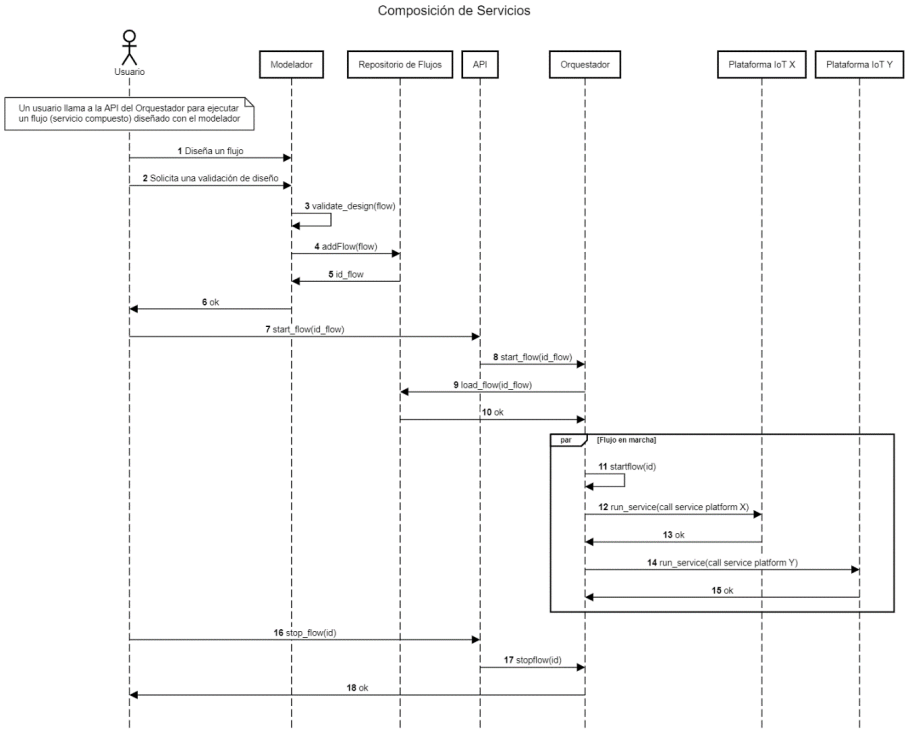


Figura 4.19: Flujo de solicitud de consulta al mecanismo de interoperabilidad.

Finalmente, es importante indicar que los capítulos relacionados con los casos de uso de INTER-IoT y ACTIVAGE, Capítulos 6 y 7, respectivamente, proporcionan ejemplos de instanciación completa de esta implementación propuesta.

4.3. Componentes habilitantes

Los componentes habilitantes se pueden definir como elementos transversales que afectan a más de una capa, y se pueden dividir en tres áreas principales: seguridad y privacidad, interacciones entre capas, y virtualización y clusterización de las soluciones.

Seguridad y Privacidad

La seguridad es un aspecto clave que hay que tener en cuenta cuando más de una solución y más de un cliente acceden al mismo sistema. No todos los sistemas tienen la suficiente capacidad o potencia de procesamiento para implementar mecanismos de seguridad fiables que garanticen la privacidad y autenticidad de los datos proporcionados. Por ello, las capas superiores tienen que suplir estas carencias y asumir el protagonismo de la implementación de la seguridad. Esta seguridad se centra en dos puntos clave: la confianza y la privacidad.

Por un lado, la confianza se refiere a varias partes de un sistema:

- Confianza en los dispositivos: Necesidad de interactuar con dispositivos fiables
- Confianza de procesamiento: Necesidad de interactuar con datos correctos y significativos.
- Confianza en la conexión: Necesidad de intercambiar los datos correctos y sólo con los proveedores de servicios adecuados.
- Confianza en el sistema: Deseo de aprovechar un sistema global fiable. Esto se puede conseguir proporcionando la mayor transparencia posible del sistema.

Por otro lado, la privacidad engloba la información sensible gestionada por un sistema IoT que puede poner en peligro una empresa, el propio sistema o la confidencialidad de los datos de los individuos. Dado que la solución propuesta se compone de varias capas y módulos, la implementación de la seguridad no puede ser monolítica. El marco proporciona un portal de acceso a varios mecanismos para facilitar la interoperabilidad, y cada uno de ellos debe estar aislado y sólo comunicarse a través de los mecanismos y canales oficiales. Su objetivo es proporcionar una interoperabilidad global y abierta a nivel de plataforma entre plataformas IoT heterogéneas acopladas a través de interfaces de interoperabilidad de cada mecanismo específicamente desarrolladas. En este sentido, la concienciación e implementación de la seguridad en este componente debe tener en cuenta estos aspectos particulares y representar una arquitectura que los englobe y cree un marco de seguridad fiable a través de los diferentes mecanismos de la solución global. No hay que olvidar que cada mecanismo es responsable de garantizar la seguridad de la información.

Por tanto, cada mecanismo será responsable de garantizar la integridad y el cifrado (si es necesario) de los datos que gestiona, así como de los datos intercambiados con otros componentes y con el marco de interoperabilidad.

Sin embargo, el mecanismo de autenticación será centralizado, para tener un acceso coherente sobre todos los componentes de un despliegue de la solución. Dicho despliegue interactuará con una o más Plataformas IoT que podrían implementar sus propios mecanismos de seguridad. Por esa razón, cuando una plataforma se registra en el marco de desarrollo, también almacenará (encriptados) todos los detalles de autenticación necesarios para interactuar con esa plataforma IoT.

Existen componentes específicos de seguridad que, unidos a los agentes anteriormente mencionados, conforman todo el diseño del modelo de privacidad y seguridad:

- Plataformas externas de IoT: Son las plataformas externas de IoT que estarán configuradas y disponibles para conectar con los mecanismos de interoperabilidad.
- Servidor Backend del marco de desarrollo: Este es el componente donde se instalará y desplegará el marco de desarrollo. También es el frontend para la gestión y configuración del despliegue de la solución global. En cuanto a los aspectos de seguridad, todas las credenciales de autenticación y los datos específicos de seguridad para interactuar con las plataformas externas de IoT se almacenarán bajo la biblioteca Cryptex.
- Servidor de autenticación de usuarios: Este servidor será responsable de manejar todos los mecanismos relacionados con la autenticación. Interactuará con el servidor backend del marco de desarrollo para proporcionar los puntos de configuración y gestión de la autenticación integrada con el frontend del marco de desarrollo, pero su objetivo principal es actuar como punto centralizado de autenticación para el resto de capas desplegadas en los contenedores. Por esta razón, el Servidor de Identidad WSO2 ha sido elegido como el servidor de autenticación centralizado. Proporciona capacidades de inicio de sesión único y federación de identidades, autenticación multifactorial, gestión de usuarios, grupos y roles, monitorización y auditoría y múltiples conectores y librerías para una fácil integración.
- Contenedor Docker-Swarm: Es el contenedor en el que se desplegarán y gestionarán los componentes de las diferentes capas. En cuanto a la seguridad y la autenticación, cada capa se encargará de implementar sus mecanismos de seguridad en el marco y el lenguaje elegido para cada componente. Para tener un mecanismo de autenticación coherente y centralizado, cada capa tendrá su conector con el Servidor de Autenticación de Usuarios, para aprovechar todas las capacidades de autenticación que éste proporciona.

La Figura 4.20 resume brevemente la interacción entre los componentes presentados.

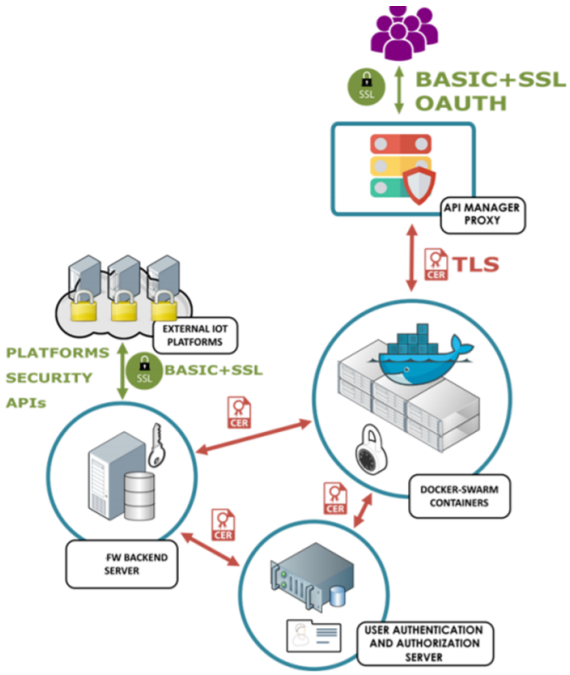


Figura 4.20: Componentes habilitantes de la seguridad y privacidad.

Interacciones entre capas

Los mecanismos habilitadores exponen toda su funcionalidad a través de su interfaz API REST. Cualquier funcionalidad implementada puede exponerse como un nodo en la solución de interoperabilidad de aplicaciones mediante flujos de interoperabilidad, como por ejemplo:

- Consultas de datos contextuales o datos históricos pueden proporcionar información en un formato esperado mediante nodos y ser conectados a otros nodos función o consumidores, principalmente, servicios de plataformas de IoT
- Las suscripciones, que notifican una serie de observaciones de un conjunto de sensores, proporcionando información a los nodos consumidores, como paneles de visualización o servicios CEP de otras Plataformas de IoT.

Virtualización y uso de contenedores para envolver los componentes y servicios

La escalabilidad de los componentes se consigue mediante el uso de contenedores Docker. Docker es una plataforma de contenedores de software que encapsula las aplicaciones para ejecutarlas y gestionarlas de forma paralela en contenedores aislados para obtener un mejor rendimiento y densidad de computación. Estos contenedores pueden comunicarse entre sí a través de una red Docker especificando la dirección y el puerto, y la herramienta Docker puede manejar el ciclo de vida de los contenedores de manera que estos paquetes de software se ejecuten aislados en un sistema operativo compartido siendo iniciados, ejecutados o detenidos cuando sea necesario. A diferencia de otros métodos o máquinas de virtualización, los contenedores no construyen un sistema operativo completo, sino sólo las bibliotecas y las configuraciones necesarias para que el software funcione según las necesidades.

Por tanto, las soluciones implementadas en cada mecanismo de interoperabilidad deben ser envueltas luego en contenedores Docker para que sus ciclos de vida sean gestionados en el marco de desarrollo y gestión. Docker-swarm proporciona una herramienta para observar y gestionar un grupo de nodos Docker como si se tratara de un solo clúster, de manera que se pueden iniciar varios contenedores Docker relacionados al mismo tiempo y tratarlos como un todo. Un enjambre es un cluster de nodos Docker donde se despliegan las soluciones, con una Interfaz de Línea de Comandos (CLI) y una API para establecer comandos para gestionar los nodos del enjambre (inicializarlos, unirse a un contenedor en ejecución, etc.). Esta herramienta se utiliza para agrupar diferentes soluciones del mismo mecanismo (por ejemplo, varias instancias de Node-RED) y gestionarlas como un conjunto. La Dockerización del marco de desarrollo también viene acompañada de un refuerzo de seguridad, ya que sólo los usuarios/roles autorizados pueden acceder a los diferentes contenedores o cluster de contenedores. En resumen, las principales características implementadas son:

- Dockerización de los mecanismos habilitadores.
- Creación de archivos Docker Compose, archivos para definir y ejecutar aplicaciones Docker que involucran múltiples contenedores y que incluyen todos los componentes dockerizados que pertenecen a un mecanismo de interoperabilidad. El propósito es ofrecer una solución completa y un despliegue funcional de todos los elementos de los mecanismos habilitadores de la interoperabilidad.
- Despliegue de registro de Docker privado para almacenar las imágenes de los contenedores.

- Integración de herramientas Docker para la gestión de contenedores: Docker Swarm y Portainer.

4.4. Marco Común

En esta sección, en primer lugar, se describe la aplicación Web desarrollada para implementar el marco común. A continuación, se describe el gestor API implementado para ofrecer el API común del marco de interoperabilidad. Finalmente, se presenta una visión compacta del componente.

Aplicación Web

El objetivo de la aplicación web es unificar la monitorización, configuración y gestión de los componentes IoT (sensores, plataformas, servicios, aplicaciones, puentes, nodos...) de los diferentes mecanismos de interoperabilidad en un único entorno. Proporciona, a través de una Interfaz Gráfica de Usuario (GUI), un único lugar de gestión para todos los mecanismos desarrollados, así como una herramienta de gestión de usuarios para el control de autenticación y autorización. Tras realizar un análisis exhaustivo de los requisitos, se decidió que la aplicación constara de los siguientes componentes:

- Font-end: el conjunto de vistas que se muestran al usuario, como las vistas: principal, plataformas, puentes, agentes, dispositivos, servicios, flujos, configuración y usuarios.
- Controlador: este componente realiza las operaciones clásicas de un controlador: como procesar y reenviar las peticiones de datos al modelo, solicitar campos, validar credenciales y permisos, etc.
- Validador de peticiones de la API: comprueba los campos, reglas y rutas de las peticiones de la API antes de enviarlas.
- Registro de usuarios: es un almacén de los usuarios de la instancia desplegada.
- Registro de elementos: es un punto de almacenamiento de los elementos registrados: plataformas, puente, servicios, etc. Cada entrada contiene la información necesaria para identificar unívocamente el elemento.
- Configuración de elementos: comprueba que la configuración presentada para cada nodo es correcta, antes de llamar a la API asociada.

Se ha diseñado tanto el front-end como el back-end (que incluye algunos de los componentes mencionados) de la aplicación, incluyendo los aspectos de seguridad de la aplicación web y la intermediación de credenciales necesaria para gestionar la seguridad y privacidad de las plataformas conectadas. La aplicación delega gran parte de sus funcionalidades en las interfaces de interoperabilidad. Sin embargo, para aportar características prácticas que puedan permitir al componente entender y gestionar los conceptos de interoperabilidad multicapa de las plataformas IoT, se necesitaban algunas operaciones de back-end. En general, estas operaciones incluyen las características de servir, transformar y persistir los datos de la aplicación, por lo que los componentes de la solución que se vieron afectados por estas operaciones fueron modificados en consecuencia. El back-end se desarrolló con Node.js como framework de la aplicación web, Mongo.db para almacenar los datos no estructurados y Mongoose para el modelado de objetos. Por otro lado, el front-end es la capa de presentación en la que los usuarios pueden ver e interactuar con el contenido en una interfaz fácil de usar. Suele desarrollarse como una mezcla de Lenguaje de Marcas de Hipertexto (HTML), Hojas de Estilo en Cascada (CSS), JavaScript (JS) y bibliotecas auxiliares. En este caso, para el desarrollo de la interfaz se ha utilizado el framework Vue, que simplifica el proceso de construcción de interfaces interactivas con el usuario.

La consola web del marco de interoperabilidad estructura cada uno de sus módulos en función de la arquitectura y los componentes de la solución de cada capa, mostrando la estructura de cada capa e información sobre los mecanismos de interoperabilidad gestionados por ellas. Más concretamente, hay diferentes pantallas para MW, MW2MW, AS2AS y ASFLOW.

Gestión común de APIs

La solución seleccionada para implementar la API común del marco de interoperabilidad es WSO2 API Manager. Esta API Manager se ha desplegado como imagen Docker para facilitar un despliegue en la nube integrado con el resto de componentes del marco de interoperabilidad. La instalación incluye el registro obligatorio del producto, el despliegue de la imagen Docker y la autenticación con la cuenta WSO2. Además, se debe configurar con las URLs adecuadas y con los puntos finales de acceso de la pasarela API. La creación de un acceso unificado a la API a través del gestor de APIs de WSO2 se implementa en varios pasos:

- En primer lugar, crear un documento de diseño de la API para cada componente. Se proporciona a través de definiciones Swagger.

- Después, se analizan las definiciones de Swagger a través de un enfoque de las mejores prácticas de REST y las diferentes convenciones de nomenclatura entre los mecanismos de interoperabilidad.
- A continuación, se define una interfaz API unificada, con el mapeo a los puntos finales correspondientes de los sistemas backend.
- Se crea una definición unificada de OpenAPI, además de un módulo mediador que mapea las APIs. A continuación, el usuario debe suscribirse a las APIs a través de la GUI web de suscripción a la API.

Los principales tipos de usuarios de la API, con su correspondiente conjunto de privilegios de acceso y gestión, son:

- Usuarios del núcleo del marco de interoperabilidad, que son usuarios con acceso completo a todas las funciones.
- Usuarios del frontend del marco de interoperabilidad, que son usuarios con un conjunto restringido de derechos de acceso necesarios para ejecutar las llamadas a la API. Además, se ha desarrollado un conjunto de políticas de acceso, utilizando definiciones SAML.

La API común se documenta considerando un formato Swagger-JSON. Se accede a las APIs de los mecanismos a través de una instancia del gestor de API de WSO2 que actúa como punto de entrada principal para la gestión, el acceso y el uso. Se ha entregado un conjunto de definiciones OpenAPI para cada mecanismo con el fin de exponer sus respectivos componentes. Éstas se han integrado en el gestor de APIs de WSO2 y se han publicado en el almacén de APIs de la solución. Allí, los usuarios pueden registrarse y obtener acceso al despliegue completo de la solución.

Marco común de interoperabilidad

Basado en los anteriores elementos, proporciona un entorno visual completo para permitir el desarrollo, la configuración y la gestión de plataformas IoT interoperables. El resultado se presenta como una aplicación web capaz de controlar múltiples aspectos de la interoperabilidad en diferentes capas. Es un marco visual que permite controlar múltiples plataformas IoT en una única interfaz de usuario. Proporciona gestión de plataformas, dispositivos, y servicios para escenarios con despliegues heterogéneos de IoT, al tiempo que protege la soberanía de los datos con información personal e industrial de forma segura y protegida. Las principales funcionalidades que cubre el marco de interoperabilidad son:

- Controlar todos los mecanismos de interoperabilidad. En una sola pantalla se gestionan visualmente con total configurabilidad.
- Completa API REST disponible que permite controlar todos los aspectos de interoperabilidad, facilitando la construcción de una aplicación sobre plataformas IoT heterogéneas.
- Autorización detallada de las operaciones sobre las capas de interoperabilidad. Control total de las operaciones de la API y del marco de desarrollo por parte del propietario de la plataforma para mantener la soberanía de los datos.

La solución gráfica ofrece pestañas como las siguientes:

- Dispositivos: se representan todos los dispositivos conectados a los mecanismos de interoperabilidad.
- Agentes: Listado de los Agentes disponibles para acceder a datos.
- Plataformas: instanciación, configuración y gestión de plataformas IoT.
- Puentes: listado conectores desplegados o disponibles para interactuar con las plataformas.
- Servicios: gestión de nodos, flujos e instancias de la solución de interoperabilidad de aplicaciones y servicios de las plataformas IoT.
- Políticas: utilidad para definir políticas XML de seguridad para la autorización de grano fino.
- Gestión de la API: enlaces al Gestor de la API común, gestionando el nivel de acceso, el versionado del ciclo de vida y la monitorización, entre otras tareas.
- Gestión de usuarios: herramienta para la gestión de los usuarios.
- Configuración: configuración también de la herramienta

Los usuarios a los que va dirigido este producto son los siguientes:

- Integradores de sistemas.
- Desarrolladores de Aplicaciones.
- Propietarios de múltiples plataformas IoT: normalmente autoridades públicas y grandes empresas.

- Curiosos de la tecnología.

Los principales beneficios obtenidos gracias a la implementación del marco son:

- Utilizar una sola herramienta para controlar múltiples Plataformas IoT.
- Proporcionar funcionalidades de gestión visual para plataformas que carecen de ellas (FIWARE, WSO2, OneM2M).
- Gestionar diferentes mecanismos de interoperabilidad en una aplicación común.
- Desarrollar aplicaciones que aprovechen los datos de múltiples plataformas heterogéneas.
- Construir aplicaciones basadas en una única API evitando el código repetitivo.

Finalmente, es importante indicar que en el Capítulo 6 relacionado con el caso de uso de INTER-IoT se proporcionan ejemplos de instanciación completa de esta implementación propuesta del marco de interoperabilidad con todas las tecnologías descritas.

4.5. Extensibilidad de la solución

Cada mecanismo de interoperabilidad implementado tiene varias posibilidades de ser extendido de forma que permiten añadir nuevas funcionalidades o mejoras a los servicios existentes:

Compatibilidad/inclusión de nuevas plataformas y/o servicios:

La inclusión de nuevas plataformas se garantiza mediante el desarrollo de puentes, nodos y agentes genéricos descritos en cada mecanismo de interoperabilidad, para incluir nuevas plataformas y el mayor número de funcionalidades, servicios y plataformas de las que disponen. Para ello, se ha prestado especial atención a la definición de pruebas unitarias y de integración, probar conjuntos de datos y otras herramientas de validación.

Generación de múltiples instancias de los mecanismos de interoperabilidad

Se han aprovechado las ventajas de la interacción con Docker y su API de gestión de contenedores para generar múltiples instancias de los mecanismos de interoperabilidad. Se puede afrontar desde dos perspectivas:

- Una solución permite acceder a diferentes instancias dockerizadas de la solución de interoperabilidad en el mismo servidor huésped.
- Varios hosts donde en cada uno hay desplegadas múltiples instancias del mecanismo de interoperabilidad. Estas instancias se manejan desde un punto único de gestión a pesar de estar en hosts diferentes. Para facilitar esta implementación se han utilizado herramientas como Docker Swarm y Docker Portainer. En algunas pruebas de concepto también se ha hecho uso de Kubernetes o Minikube.

Esta relación con las API u opciones que ofrece Docker facilita la extensibilidad de la solución en términos de escalabilidad, portabilidad, seguridad y fácil implementación de soluciones. Finalmente, a modo de ejemplo específico, en el mecanismo de interoperabilidad de aplicaciones mediante flujos de interoperabilidad cada instancia del servidor tiene sus propias carpetas con la misma distribución y quedaría como se muestra en la Figura 4.21. En el caso de usar más de un host la solución sería como muestra la Figura 4.22.

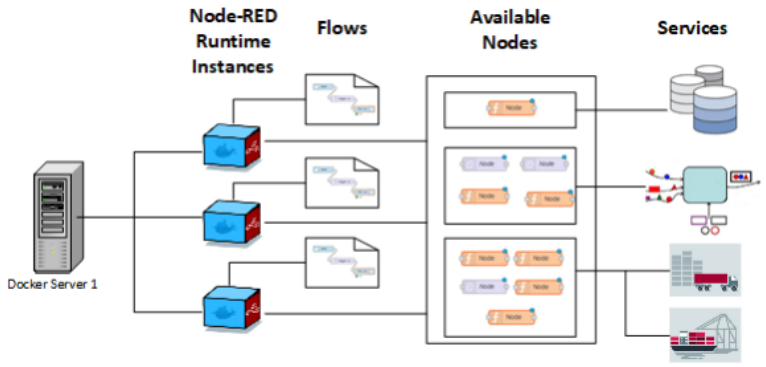


Figura 4.21: Múltiples instancias de un mecanismo de interoperabilidad en único servidor.

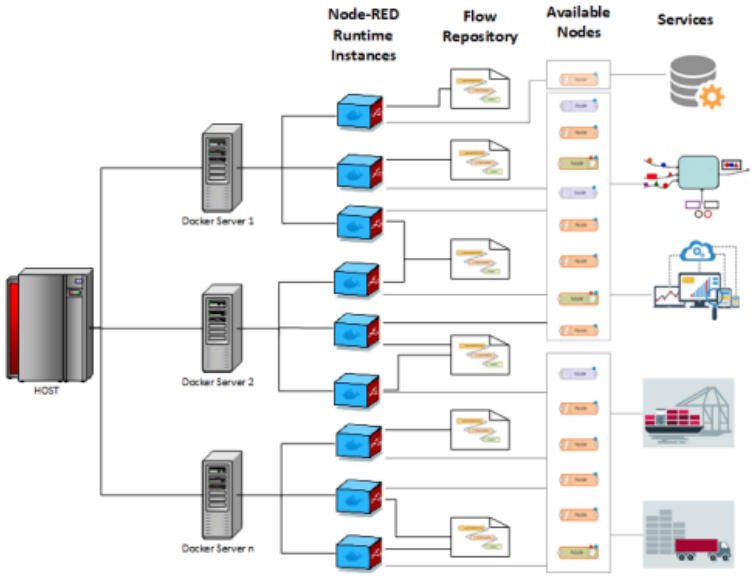


Figura 4.22: Múltiples instancias de un mecanismo de interoperabilidad en varios servidores.

API-REST del Marco común de interoperabilidad

El gestor de API proporciona la gestión del ciclo de vida de la API y el control de versiones, de modo que cada cambio se despliega normalmente como un prototipo para su promoción temprana. Tras un periodo de tiempo durante el cual la nueva versión se utiliza en paralelo con las versiones anteriores, la API prototipo puede publicarse y sus versiones antiguas pueden depreciarse. Con las actualizaciones de las definiciones OpenAPI de la API REST unificada de la solución, cada cambio se documenta y se presenta a los posibles usuarios en un formato estándar adoptado por la mayoría de los marcos de desarrollo de software modernos. En principio, WSO2 API Manager puede integrarse con varios tipos de almacenes de usuarios, como LDAP, Active Directory y entornos personalizados.

Servidor de identidad

La extensibilidad del Servidor de Identidad se logra a través de varios puntos de extensión:

- Creación y aplicación de nuevas políticas de lenguaje extensible de marcas de acceso de control (XACML).
- Agregación de nuevos autenticadores y conectores (nuevos proveedores de identidad) para la federación de identidades. Esto podría permitir que los usuarios accedan a la solución sin tener una cuenta de la solución, aprovechando el Servidor de Identidad para controlar y restringir las partes más sensibles del sistema. Un posible escenario de esta extensión podría ser un despliegue de la solución global por parte de un servicio público que permita a todos los usuarios (por ejemplo, los ciudadanos en una Smart City) acceder a información de sólo lectura sobre las diferentes plataformas IoT de propiedad pública que operan en una zona.
- El uso del Servidor de Identidad tiene un tercer punto de expansión a través de los Mediadores de Derechos. Un Mediador de Derechos intercepta las solicitudes y evalúa las acciones realizadas por los usuarios con respecto a una política XACML. El Servidor de Identidad puede utilizarse como Punto de Decisión de Políticas XACML (PDP), donde se establece la política. Esto también se explica en la documentación oficial, y consiste esencialmente en añadir puntos de autorización al Servidor de Identidad para proteger los puntos finales en la API común y otros recursos.

Contribución Open Source a diferentes iniciativas

Se colabora con diferentes iniciativas mediante:

- Desarrollo de nuevos modelos de datos compatibles con la iniciativa Smart Data Models mantenida por la fundación FIWARE.
- Generación de nodos genéricos para incluir nuevas plataformas IoT en Node-RED.
- Generación de flujos específicos para resolver problemas de interoperabilidad entre aplicaciones.
- Contribuir a proporcionar funcionalidades disponibles en otras plataformas, en aquellas que no las presentan mediante el uso de los conectores.
- Retroalimentación continua con los componentes desarrollados en el marco de los proyectos de investigación, principalmente, proyectos europeos del horizonte H2020.

- Colaboración en mejora de funcionalidades existentes de plataformas IoT abiertas. Por ejemplo, se ha dado el caso de plataformas que no proporcionaban un API REST, mediante los conectores desarrollados se ha proporcionado mecanismos alternativos para trabajar con estas plataformas.
- Colaboración en soluciones software Open Source. Se han creado ramas de proyectos o notificado errores de software abierto desarrollado por terceros.

4.6. Visión técnica de la Solución Software

A modo de resumen, los componentes software dockerizados y necesarios para desplegar los mecanismos habilitadores de interoperabilidad son los siguientes:

- Interoperabilidad Middleware:
  - Componentes Core:
    - Orion Context Broker.
    - MongoDB.
  - Agentes:
    - Piezas código Python.
  - Acceso a Datos:
    - Gestor UI.
    - API.
  - Componente Bajo Demanda.
  - Registro Git del Data Model.
- Interoperabilidad de Plataformas Middleware:
  - Componentes Core:
    - Componente completo de interoperabilidad Middleware desarrollado en Java.
    - RabbitMQ.
    - Parliament Triple Store.
  - Bridges:
    - Codigo Java de cada bridge desarrollado que funciona en el componente core.

- Interoperabilidad de Aplicaciones y Servicios (según implementación seleccionada)
  - Componentes Core:
    - Interoperabilidad Middleware / Interoperabilidad de Plataformas / Interoperabilidad de Plataformas Middleware.
  - Agentes/Nodos/Puentes Adaptados.
  - Componente Bajo demanda:
    - Adaptado a Orion/ Node-RED /Componente core de Interoperabilidad de Plataformas Middleware.
- Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos:
  - Componente Core:
    - Node-RED.
  - Repositorio de Nodos y Flujos.
  - Gestor y configurador de instancias del componente core.
  - Gestor de multiples instancias del componente core:
    - Versión Minikubes o Kubernetes.
    - Versión Docker o Docker Swarm.

Todo esto se despliega dentro del marco de interoperabilidad, compuesto por:

- Marco de interoperabilidad:
  - Componente Core:
    - Frontend (Vue).
    - Backend (Node.js).
    - MongoDB.
  - Gestor de Identidades:
    - WSO2 servidor de Identidades.
    - MySQl.
  - Gestor REST API:
    - WSO2 gestor de APIs .

Capítulo 5

Especificación y desarrollo: Casos de Uso

El presente capitulo está enfocado a ofrecer una visión general de los casos de uso mediante los cuales se ha llevado a cabo la validación y despliegue de la especificación y desarrollo de mecanismos de interoperabilidad presentada en los dos capítulos anteriores (capítulos 3 y 4). Hay que diferenciar dos ámbitos de trabajo para llevar a cabo esta tarea. En primer lugar, la solución de interoperabilidad ha sido desplegada y validada en el laboratorio de investigación, mediante la infraestructura, plataformas, aplicaciones y datos de prueba disponibles. Esta tarea se ha llevado a cabo durante todo el periodo cubierto por la realización de esta Tesis Doctoral y ha cubierto las necesidades experimentales de la solución. En segundo lugar, para obtener un mayor valor aplicado, la solución propuesta se ha validado mediante cuatro casos de uso, que han ofrecido datos y despliegues reales de plataformas y aplicaciones. Dichos casos de uso se corresponden con tareas y paquetes de trabajo concretos de cuatro proyectos de investigación financiados por la Unión Europea dentro del Programa Marco denominado Horizonte 2020. Estos proyectos serán explicados detalladamente en los capítulos 6, 7 y 8. Los resultados obtenidos han sido analizados para extraer conclusiones relevantes sobre la investigación realizada y establecer nuevas líneas de investigación.

5.1. Visión general, motivación y cronología

La Unión Europea ha respaldado el ecosistema y la comunidad IoT mediante diferentes acciones de investigación e innovación durante los últimos años.

Los proyectos de investigación son la pieza fundamental para dar soporte a esta Tesis Doctoral. Particularmente, han sido varios los proyectos en los que el autor del presente trabajo se ha involucrado durante estos últimos años. Los proyectos europeos H2020 INTER-IoT, ACTIVAGE, PIXEL y DataPorts son los utilizados para validar los mecanismos de interoperabilidad desarrollados. Principalmente, ello es debido a que en todos ellos se han desarrollado plataformas en las que los mecanismos de interoperabilidad en las capas Middleware o de Aplicación y Servicios son una pieza fundamental.

Cronológicamente, tal como se muestra en la Figura 5.1, la validación de los mecanismos de interoperabilidad empezó con INTER-IoT, que está totalmente enfocado al desarrollo de nuevos mecanismos y soluciones para garantizar la interoperabilidad de plataformas, aplicaciones y servicios existentes. Posteriormente, continuó con ACTIVAGE, que ofrece las capacidades de ser un piloto a gran escala para validar dichos mecanismos. Finalmente, ha concluido con PIXEL y DataPorts, para los cuales la interoperabilidad es un habilitador dentro del marco de una solución software más completa. Ambos proyectos, al presentar amplias dependencias, se presentarán en el mismo capítulo.

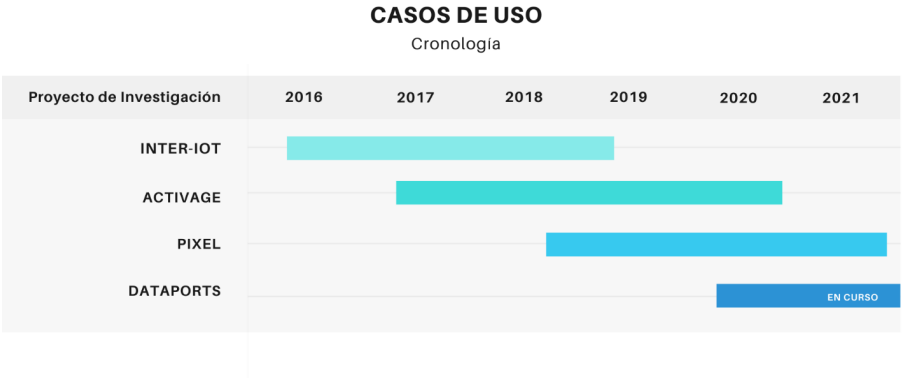


Figura 5.1: Cronología proyectos investigación.

5.2. Integración de la solución en entornos prácticos

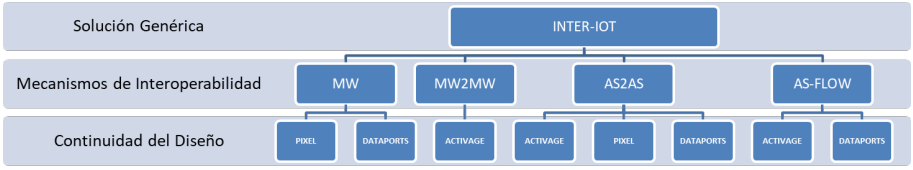
La solución propuesta en esta Tesis Doctoral ha sido concebida para ser genérica y adaptarse a diferentes áreas o dominios de aplicación relacionados con IoT. No obstante, su validación se ha desarrollado e integrado en varios dominios (Figura 5.2) de aplicación mediante el uso de los datos disponibles en los casos de uso.

Caso de Uso	Dominio de Aplicación
Prototipo de Laboratorio	• Solución conceptualmente genérica.
INTER-IoT	• Solución conceptualmente genérica. • Transporte y logística en entornos portuarios. • Salud.
ACTIVAGE	• Salud: ○ Envejecimiento activo y saludable.
PIXEL	• Ecosistemas portuarios: ○ Impacto medioambiental.

DataPorts	• Ecosistemas portuarios:
	  ○ Puertos cognitivos del futuro.
	  ○ Entorno seguro de intercambio de datos.

Figura 5.2: Dominios de aplicación de los casos de uso.

Desde un punto de vista técnico, los casos de uso han seguido una evolución natural basada en el trabajo de investigación. Esto ha sido paralelo a las necesidades técnicas de cada proyecto y los objetivos enunciados por la Unión Europea, que han motivado la financiación de dichos proyectos de investigación. En la Figura 5.3 se puede observar la relación y evolución de los mecanismos de interoperabilidad mediante cada caso de uso.



ABREVIATURA	MECANISMO
MW	Interoperabilidad Middleware
MW2MW	Interoperabilidad de Plataformas Middleware
AS2AS	Interoperabilidad de Aplicaciones y Servicios
AS-FLOW	Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos

Figura 5.3: Desarrollo de los mecanismos de interoperabilidad en los casos de uso.

Finalmente, los casos de uso están relacionados con los mecanismos de interoperabilidad previamente descritos, tal como se resume en las Figuras 5.4 y 5.5.

Caso de Uso	Mecanismos de interoperabilidad desarrollados
Prototipo Laboratorio	• Prueba y validación en laboratorio de todas las posibles herramientas que se pueden utilizar para llevar a cabo la: ○ Interoperabilidad Middleware ○ Interoperabilidad de Plataformas Middleware ○ Interoperabilidad de Aplicaciones y Servicios ○ Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos • Entorno de soporte, living-lab, pruebas y validación antes de poner en marcha las soluciones implementadas en los casos de uso. También entorno de soporte para demostraciones y pruebas de concepto.
INTER-IoT	• Interoperabilidad Middleware ○ Análisis pormenorizado de cada plataforma. ○ Diseño e implementación de mecanismos de acceso y gestión a cada plataforma. • Interoperabilidad de Plataformas Middleware ○ Despliegue y configuración Plataformas IoT Middleware ○ Diseño e implementación de la solución y validación en casos de uso • Interoperabilidad de Aplicaciones y Servicios ○ Análisis pormenorizado de servicios y aplicaciones de cada plataforma. ○ Diseño e implementación de mecanismos de acceso y gestión a cada plataforma. • Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos ○ Despliegue y configuración de servicios y aplicaciones de plataformas IoT ○ Diseño e implementación de la solución y validación en casos de uso

Figura 5.4: Mecanismos de interoperabilidad desarrollados en los casos de uso (I).

Caso de Uso	Mecanismos de interoperabilidad desarrollados

ACTIVAGE	• Interoperabilidad de Plataformas Middleware
	  ○ Despliegue y configuración Plataformas IoT Middleware
	  ○ Validación de la solución en casos de uso
	• Interoperabilidad de Aplicaciones y Servicios
	  ○ Despliegue y configuración de servicios y aplicaciones de plataformas IoT
	  ○ Diseño e implementación de la solución y validación en casos de uso
	• Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos
	  ○ Despliegue y configuración de servicios y aplicaciones de plataformas IoT
	  ○ Validación de la solución en casos de uso

PIXEL	• Interoperabilidad Middleware
	  ○ Despliegue y configuración Plataformas IoT Middleware y fuentes de datos heterogéneas
	  ○ Implementación nuevos mecanismos para extender la solución Middleware existente
	  ○ Diseño e implementación de la solución y validación en casos de uso

DataPorts	• Interoperabilidad Middleware
	  ○ Despliegue y configuración Plataformas IoT Middleware y fuentes de datos heterogéneas
	  ○ Implementación nuevos mecanismos para extender la solución Middleware existente
	  ○ Diseño e implementación de la solución y validación en casos de uso
	• Interoperabilidad de Aplicaciones y Servicios
	  ○ Despliegue y configuración de servicios y aplicaciones de plataformas IoT
	  ○ Diseño e implementación de la solución y validación en casos de uso
	• Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos
	  ○ Despliegue y configuración de servicios y aplicaciones de plataformas IoT
	  ○ Validación de la solución en casos de uso

Figura 5.5: Mecanismos de interoperabilidad desarrollados en los casos de uso (II).

5.3. Entornos de validación

Este apartado ofrece una visión general del entorno de validación disponible en el laboratorio de pruebas y en los diferentes proyectos de investigación. La infraestructura, materiales y equipos disponibles para llevar a cabo la validación en cada entorno se detalla en los siguientes subapartados:

Entorno de Laboratorio

Se dispone de un laboratorio de pruebas virtual compuesto por los servidores necesarios, además de los routers o switches necesarios para las pruebas y otros dispositivos, como sensores, Raspberry PIs u ordenadores portátiles. Principalmente se dispuso de los siguientes elementos:

En primer lugar, se dispone de tres servidores actuando como huésped en los que se instaló el hipervisor ESXi de VMWARE con licencia de estudiante (Figura 5.6), mediante el cual se podían ir creando los servidores virtuales necesarios. Fueron adquiridos aproximadamente en 2016, con las siguientes prestaciones cada uno: 4 CPU de 3GHz, 64 GB de RAM y 3 TB de disco duro combinado SSD y mecánico.

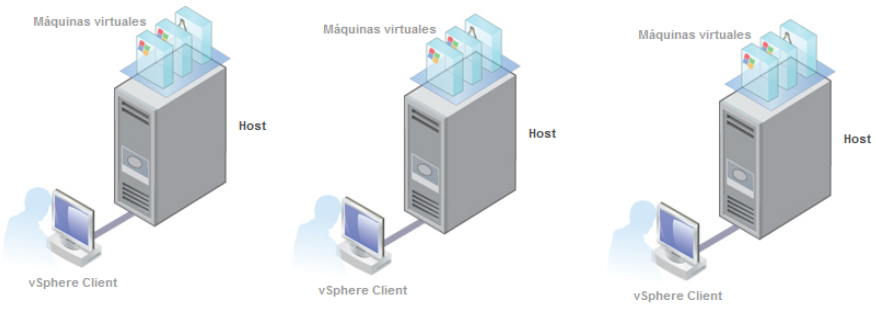


Figura 5.6: Servidores VMWARE disponibles en laboratorio.

En segundo lugar, se dispone de un clúster de computación en la nube privada, adquirido en 2020, que consta de tres servidores (cada uno con 2x Intel Xeon Gold 6320R, 512 GB de RAM y 1 TB SSD), dos conmutadores de alto rendimiento (con 48 x 10 Gbps SFP +) y dos servidores de almacenamiento conectado a la red (NAS), cada uno de ellos con SSD de 16 TB. El hipervisor utilizado en este cluster fue PROXMOX (Figura 5.7).

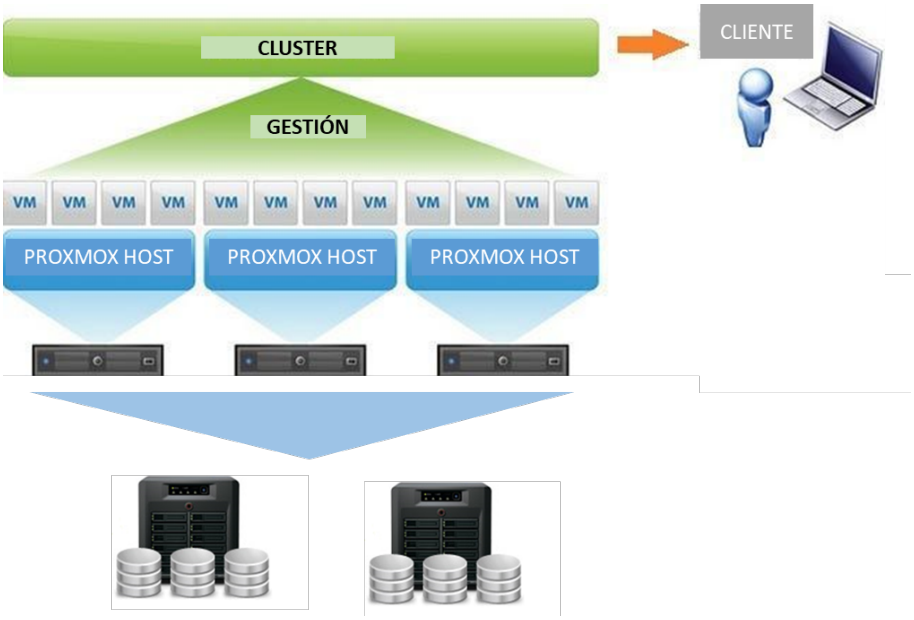


Figura 5.7: Servidores PROXMOX disponibles en laboratorio.

Finalmente. se dispone de una báscula y un tensiómetro digitales que establecían conexión por Bluetooth, tres Raspberry PI 2, tres Raspberry PI 3, y sensores y arduinos, principalmente utilizados en otros proyectos.

INTER-IOT

En este proyecto, el desarrollo de los diferentes mecanismos de interoperabilidad, elementos habilitantes y marco común se ha desplegado en un entorno de desarrollo en la nube. Este entorno está basado en la nube de Microsoft Azure y comprende un conjunto de 7 servidores para desacoplar el desarrollo de los diferentes módulos de software y, al mismo tiempo, hacer que las últimas características estén disponibles para ser probadas. A estos servidores se accede a través de un único servidor de paso. El acceso a este entorno está protegido a través de mecanismos de seguridad estándar de Microsoft Azure. Se parte de una máquina virtual que permite una gestión pública y es accesible a través de una dirección de Internet pública utilizando el protocolo SSH. Esta es la puerta de entrada a una nube privada compuesta por los siete servidores. Todos los servidores están accesibles usando SSH a través del servidor de gestión mediante su nombre de host o dirección IP privada. Las IP públicas y privadas

son estáticas. Los servidores estuvieron activos desde las 07.00 hasta las 19.00 durante aproximadamente dos años, quedando apagados por la tarde y la noche. La Figura 5.8 muestra cómo se configuró el entorno en la nube Microsoft Azure.

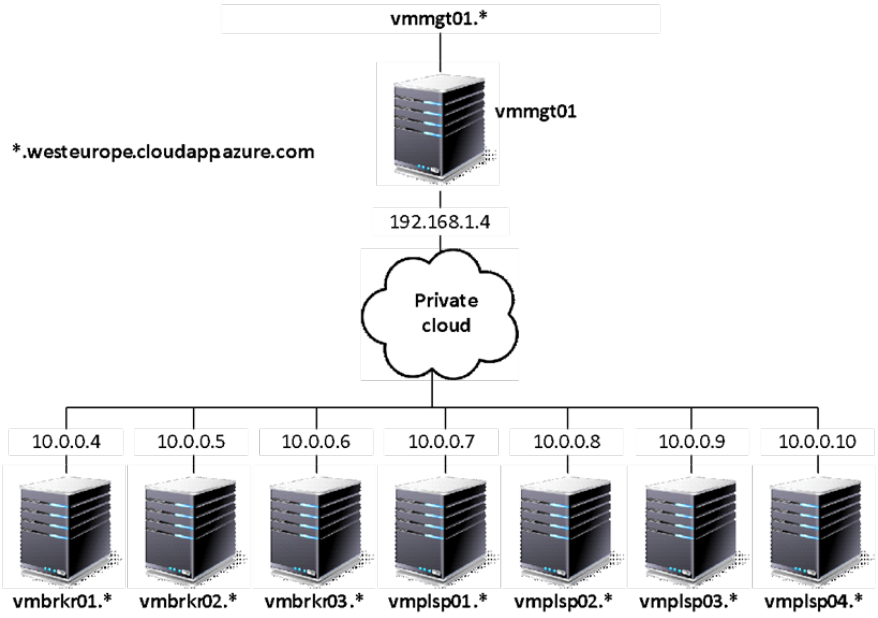


Figura 5.8: Despliegue de máquinas Microsoft Azure en el proyecto INTER-IoT.

ACTIVAGE

En este proyecto, en primer lugar, principalmente se validó la solución utilizando las máquinas virtuales incluidas en los tres servidores virtuales disponibles en el laboratorio, mediante sensores virtuales, el tensiómetro y la báscula digitales. En segundo lugar, se dispuso de dos máquinas virtuales de un proveedor de servicios en la nube proporcionadas por el proyecto. Finalmente, cada entorno de despliegue (ubicado en siete localizaciones diferentes a lo largo de Europa) ofreció, al menos, una máquina de pruebas accesible por SSH que cubría los requisitos mínimos para desplegar la solución.

PIXEL

Los mecanismos de interoperabilidad en este proyecto se alojan en un entorno OpenStack de Fiware Lab con servidores virtualizados disponibles bajo demanda dentro de dicho entorno. Todos los despliegues están basados en Docker y, por lo tanto, se pueden desplegar y migrar fácilmente. La infraestructura para realizar dichos despliegues se centró en dos objetivos: en primer lugar, en una demostración individual de los componentes habilitadores diseñados; y, en segundo lugar, para cubrir las necesidades de la integración en un marco común que ofrezca mecanismos habilitadores como la seguridad, gestión y visualización común. FIWARE Lab es el entorno no comercial de la Comunidad FIWARE. Ofrece la capacidad de innovar y experimentar con las tecnologías FIWARE de forma gratuita. Empresarios y particulares pueden probar las tecnologías FIWARE, así como sus aplicaciones dentro del FIWARE Lab, con la posibilidad de explotar los Datos Abiertos publicados por las ciudades y otras organizaciones. FIWARE Lab se despliega en una red distribuida geográficamente de nodos federados de FIWARE Lab. Cada nodo de FIWARE Lab se asigna a uno, o a una red de centros de datos sobre los que se ha desplegado una instancia de OpenStack, federada y configurada como un nodo de FIWARE Lab operado por una organización específica. El socio del proyecto PiXEL, Orange, junto con otras organizaciones gestiona uno de esos nodos y lo utiliza para proporcionar una plataforma de integración para el proyecto PIXEL. Se ofrecieron recursos para el proyecto, como la posibilidad de desplegar 15 instancias de máquinas virtuales, 30 CPUs virtuales o un máximo de 82 GB de RAM para todas las máquinas.

DataPorts

En momento actual, correspondiente al periodo de finalización del presente documento, la principal infraestructura usada en este proyecto para la validación fueron los recursos ofrecidos por el entorno de laboratorio de la UPV. Además, se dispuso de servidores virtuales de otras organizaciones, principalmente, para integrar los diferentes componentes dentro de un entorno seguro y validar la seguridad en el intercambio de datos entre organizaciones. Por ejemplo, el puerto de Valencia ofreció dos servidores virtuales dentro de un hipervisor VMWARE y otros socios en Grecia también ofrecieron acceso a tres máquinas virtuales. Los servicios externos ofrecían acceso por SSH y por VPN, se ofrecía algún puerto de acceso público en caso de ser necesario y los recursos ofrecidos eran transparentes pero los adecuados para desplegar los componentes de manera eficiente.

5.4. Plataformas y Tecnologías implicadas en la validación

Para la validación de las soluciones se ha contado con diferentes plataformas y aplicaciones. Se presenta esta información mediante dos figuras para que el lector pueda ver a golpe de vista la información relacionada con cada proyecto. La primera, Figura 5.9, ofrece las plataformas y aplicaciones, explicadas en el Capítulo 2 relacionado con el estado del arte, para las cuales se han desarrollado los mecanismos de interoperabilidad. Las segunda, Figura 5.10, muestra las tecnologías habilitantes del mecanismo de interoperabilidad que se han utilizado en cada proyecto.

Caso de Uso	Plataformas / Aplicaciones / Servicios
Prototipo Laboratorio	Plataformas: Azure, UniversAAL, WSO2 port, FIWARE OM2M, OpenIoT, sensiNact, IoTivity, Sofia2, openHAB, OneSite
INTER-IoT	Plataformas: Azure, SEAMS2, UniversAAL, WSO2 port, FIWARE OM2M, e3tcity, Body Cloud, OpenIoT, sensiNact, IoTivity, Sofia2, openHAB.
ACTIVAGE	Plataformas: FIWARE, OpenIoT, SENSINACT, IoTIVITY, UniversAAL, INTER-IoT, IoT Eclipse, OneM2M, SOFIA2
PIXEL	Plataformas: VIGIESip, OpenWeatherMap, DarkSky, HERE, AISHub, TrafficThess, NAMI (Tide level platform), ATMO (Air Pollution), Sencrop (weather), PVGIS (European Commission Phovoltaic GIS Info), SILI (Integrated transport platform FVG Italy region), SDAG's Access Control System Platform, MarineTraffic. Aplicaciones y servicios especificos: AirStation sensor, NGSI agents to specific web endpoints, REST APIs from ports, Websites' scraping, FAL forms storage servers (FTP), MQTT Sensors HOPU
DataPorts	Plataformas: Traxens Platform, Tradelens, Posidonia Operations, Posidonia Management, Valenciaport PCS, vForwarding, Valencia Port Platform, Valencia Port Management System, COREOR Aplicaciones y servicios específicos: Datos de TOS(contenedores, barcos), , Sistema de citas para camiones, Aplicación de zonas libres, Escalas previstas de buques portacontenedores, Sistema de acceso a la puerta, llamadas de cruceros, datos de movilidad…

Figura 5.9: Plataformas y servicios utilizados en los casos de uso.

Caso de Uso	Tecnologías destacadas usadas
Prototipo Laboratorio	Plataformas IoT (FIWARE (Orion Context Broker, IoT agents, STH, Proton, Cepheus, Perseo, CKAN, Idra, Cygnus, Draco, Wirecloud), UAAL, SOFIA 2, SENSINACT...), RABBIT-MQ, MongoDB, ElasticSearch, Kibana, Wilma, Keyrock, Authforce, Keyclock, Traeffik, WSO2, Kong, Node Express Gateway, Node-RED, Codigo software en disitintos lenguajes: Java, Javascript, Python, PHP, Node.js, Vue.js, Angular y Element
INTER-IoT	FIWARE (Orion Context Broker, IoT agents, STH, Proton, Cepheus, Perseo, CKAN, Idra, Cygnus, Draco, Wirecloud), Plataformas y servicios IoT (FIWARE, UAAL, SOFIA 2, SENSINACT...), RABBIT-MQ, MongoDB, WSO2, Kong, Node-RED, Codigo software en disitintos lenguajes: Java, Javascript, Python, PHP, Node.js y Vue.js
ACTIVAGE	FIWARE (Orion Context Broker, IoT agents, STH, Cygnus, Draco), Plataformas y servicios IoT (FIWARE, UAAL, SOFIA 2, SENSINACT...), RABBIT-MQ, Node Express Gateway, Keycloack, Autoproxy, MongoDB, Node-RED, Codigo software en disitintos lenguajes: Java, Javascript, Python, PHP, Node.js y Vue.js
PIXEL	FIWARE (Orion Context Broker, STH, Wilma, Keyrock, Authforce), MongoDB. Docker, ElasticSearch, Código software en Python, PyNGSI, Swagger.
DataPorts	FIWARE (Orion Context Broker, Cygnus, CKAN, Idra, Wirecloud), MongoDB, Docker, Codigo software en Python, Java, Node.js, Angular, Element y Vue.js, PyNGSI, Swagger, Traeffik, Keycloak

Figura 5.10: Tecnologias utilizadas en los casos de uso.

5.5. Lecciones aprendidas durante el desarrollo de los casos de uso

En esta sección se detalla una lista de las consideraciones a tener en cuenta y obtenidas durante el desarrollo de los mecanismos de interoperabilidad y en su validación en los casos de uso. Se parte del objetivo de esta Tesis Doctoral de proporcionar un marco de IoT interoperable y abierto, con herramientas y metodología de ingeniería asociadas, para una integración con las plataformas de IoT heterogéneas, independientemente de los dominios de aplicación a los que pertenezcan. Han sido un gran numero de lecciones aprendidas durante la elaboración de la presente Tesis Doctoral y están relacionadas, principalmente, con la integración de diferentes plataformas, servicios y aplicaciones en la solu-

ción propuesta en la misma. Por ello, con el fin de proporcionar la información de una forma más estructurada, se ha decidido presentar el conocimiento en diferentes áreas de interés, mostradas en la Figura 5.11.

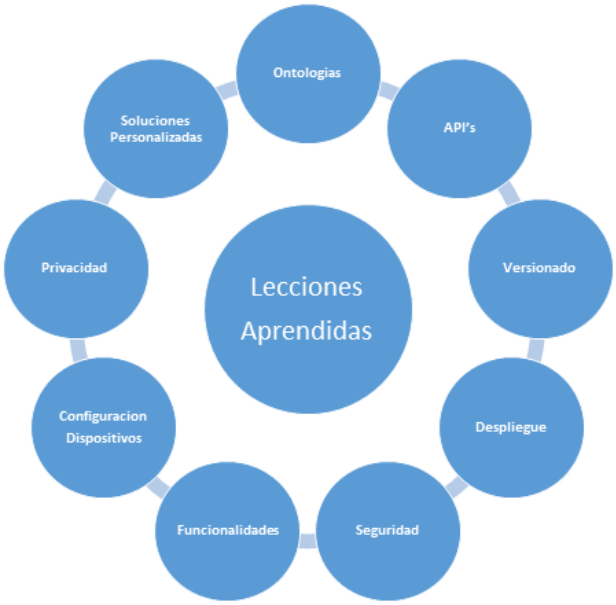


Figura 5.11: Categorización de las lecciones aprendidas en los casos de uso.

En las siguientes subsecciones se identifica la problemática encontrada en cada uno de estos grupos de conocimiento:

Ontologías y modelos de datos de las plataformas

Para interoperar con éxito diferentes plataformas de IoT, es fundamental permitirles comunicarse de manera significativa. Sin embargo, sería totalmente irreal suponer que todas hablan el mismo idioma. Por lo tanto, obviamente, se requiere un mecanismo de traducción eficiente que apoye/permita el intercambio de información. Para lograr este objetivo, las plataformas deben proporcionar descripciones de los modelos de datos que exponen al mundo exterior. Dicha descripción se puede proporcionar a través de una ontología estándar o personalizada, es decir, una representación de un conocimiento de dominio mediante un conjunto de conceptos y las relaciones entre esos conceptos. Por supuesto, cuanto más estandarizada es la ontología de la plataforma, menos

difícil se vuelve el problema de interoperabilidad, ya que la reutilización de las herramientas y soluciones existentes se vuelve factible.

Desafortunadamente, no todas las plataformas de IoT han definido sus modelos de datos con la suficiente rigurosidad, y aún menos lo hicieron utilizando ontologías explícitas. En el peor de los casos, es posible que una plataforma no ofrezca ninguna descripción semántica de los datos. En tal caso, se debe hacer un esfuerzo adicional para extraer la información necesaria y construir / elegir una ontología adecuada para ello. Otro posible problema surge del hecho de que las tecnologías semánticas todavía no son ampliamente aceptadas por la industria, e incluso características simples como, los formatos de datos enriquecidos o anotados semánticamente, no se utilizan con tanta frecuencia. Para hacer la situación aún más difícil, al mismo tiempo, muchas de las herramientas disponibles relacionadas con la semántica no están realmente preparadas para la producción y requieren conocimientos especializados por parte del usuario.

En algunos casos, la plataforma de IoT puede admitir semántica (personalizada o una de las ontologías estándar disponibles abiertamente) pero utiliza un modelo de datos que contiene elementos muy particulares. Es imposible que la solución de interoperabilidad prevea todas estas situaciones. En cambio, un enfoque de interoperabilidad exitoso tiene que ofrecer una arquitectura modular que permita extensiones. Preferiblemente, la arquitectura también debería hacer que el proceso de agregar una nueva plataforma al ecosistema sea lo más simple y eficiente posible.

Funcionalidades de las plataformas

Existe una gran cantidad de plataformas IoT y se han establecido una serie de funcionalidades básicas que deben ofrecer las plataformas IoT, como, por ejemplo, la capacidad de suscripción, actuación, gestión de dispositivos virtuales y descubrimiento. Las plataformas ofrecen acceso a sus funcionalidades, ofreciendo sus propios métodos. Las principales funcionalidades tienden a estar cubiertas por las plataformas IoT, pero, en algunas situaciones, puede ser necesario diseñar e implementar un módulo o componente para obtener la información en la forma o formato deseado o para realizar alguna acción en la plataforma. El caso más difícil es el de una plataforma que no tiene implementada una funcionalidad concreta y, por tanto, el puente no tiene capacidad para realizar esa funcionalidad. También, relacionado con la ausencia de funcionalidades, está el caso de funcionalidades, a veces, de Plataformas IoT que declaran ser abiertas y disponibles, pero en la práctica son inaccesibles, no se mantienen o no están terminadas.

Despliegue de la plataforma

Una dificultad añadida al intentar conectar tantas plataformas heterogéneas son las diferentes configuraciones de despliegue que pueden adoptar éstas. Algunas de las plataformas integradas han sido diseñadas para ser desplegadas únicamente en una arquitectura basada en la nube. Otras son precisamente lo contrario y sólo pueden desplegarse localmente, no en la nube. Los mecanismos de interoperabilidad tienen que ser capaces de conectar cualquier opción y proporcionar el acceso posible a los integradores, independientemente del tipo de plataformas que se integren.

Diferentes versiones o implementaciones de las plataformas

Existen diferentes situaciones que han dado lugar a diferentes versiones de una misma plataforma IoT:

- En primer lugar, estas plataformas aparecieron hace unos años y, durante este tiempo, han ido apareciendo nuevas tecnologías, estándares y tendencias. Por tanto, cada versión de la plataforma puede incluir diferentes funcionalidades y dificultar la retrocompatibilidad.
- En segundo lugar, muchas plataformas se lanzaron en un estado de madurez que no era óptimo, con el objetivo de ser probadas y mejoradas por una comunidad abierta de desarrolladores. Sin embargo, a veces sucede que hay Plataformas de IoT que afirman estar abiertas, pero en la práctica son inaccesibles o no se mantienen.
- En tercer lugar, las empresas que se encargan de desarrollar e implementar este tipo de soluciones, suelen ofrecer distintas versiones comerciales según la inversión económica que se realice en las mismas, ofreciendo, por tanto distintas funcionalidades y componentes según la versión adquirida (por ejemplo, libre o de pago). Además, en otros casos, adaptan la solución a las necesidades específicas de los clientes.

Por tanto, es necesario considerar todos estos factores para lograr el correcto desarrollo de los mecanismos para conectar con la plataforma. El mejor método consiste en almacenar la información de la versión de la plataforma para la que ha sido creado el conector y en que circunstancias ha funcionado.

Seguridad de las plataformas

Cada plataforma interoperable integrada tiene su propio enfoque en materia de seguridad. No sólo depende de cómo permitir el acceso de un determinado cliente a la plataforma, sino también de cómo asegurar su comunicación, sus datos almacenados, etc.

No es posible aplicar un enfoque de seguridad unificado para todas las plataformas integradas, ya que la seguridad es parte integral de cada plataforma independiente y no pueden ser modificadas para ser conectadas a los mecanismos de interoperabilidad. Eso desafiaría el objetivo de interoperabilidad. Sin embargo, hay algunos enfoques que deben seguirse para asegurar un despliegue global con una infraestructura particular y un conjunto determinado de plataformas integradas.

El capítulo anterior proporciona los mecanismos de seguridad para las características propias de la solución propuesta en esta Tesis Doctoral. Los diferentes mecanismos de seguridad, combinados con las características de seguridad de la solución propuesta, deberían proporcionar un nivel de seguridad deseado para toda la solución. Incluso en los casos en que algunas de las plataformas integradas no proporcionen suficiente seguridad en escenarios específicos, la solución global orquestada en esta Tesis Doctoral debería ser suficiente para asegurar todo el despliegue.

API de plataformas de IoT heterogéneas

El objetivo de la solución propuesta es la hacer uso de las APIs de las plataformas IoT. Mediante ellas exponen sus funcionalidades. En un escenario de integración de la vida real, se pueden encontrar diferentes niveles de complejidad con respecto a las APIs de la plataforma. Además, es posible que algunas plataformas de IoT no sigan los estándares establecidos o incluso no proporcionen ninguna API.

Privacidad

En los casos de uso se aborda la privacidad desde el punto de vista del sector industrial o de las soluciones relacionadas con la salud y el bienestar:

- En un sector industrial, como el portuario, en el que existe una enorme competencia entre las distintas empresas y conseguir la interoperabilidad de los datos entre ellas es una ardua tarea. Ahora mismo, sólo comparten la documentación mínima exigida, por ejemplo, por un organismo público como las aduanas. Además, otra razón para no compartir datos con otras entidades es que sus sistemas no están preparados para las operaciones. Esta exigencia supone un esfuerzo económico y en personal considerable.

Es importante demostrar que el beneficio que obtienen es mayor que los problemas que les ocasiona un cambio o adaptación de sus sistemas. En el caso de que exista un acuerdo para intercambiar algunos datos, las empresas quieren estar seguras de que los datos son accesibles para las personas autorizadas y se utilizan para el propósito definido.

- En el sector sanitario, la preservación de la privacidad es una de las principales preocupaciones de cualquier solución informática en este ámbito. Estas soluciones tratan con datos muy personales y sensibles y, en consecuencia, se trata de un entorno fuertemente regulado, desde el nuevo GDPR hasta las normativas locales. Estas imponen muchas restricciones a las soluciones tecnológicas que se utilizan, no solo técnicas sino también de procedimiento y legales. Otro obstáculo es que hasta ahora es difícil, al menos en algunos entornos sanitarios, encontrar soluciones informáticas integradas. Hay muchos subcampos en cualquier institución sanitaria y puede haber soluciones informáticas para cada uno de ellos. Históricamente, tanto por la normativa mencionada como por otros muchos factores, estos sistemas no suelen ser interoperables.

Configuración de dispositivos

Uno de los problemas más comunes a los que se enfrentan los despliegues de IoT es encontrar dispositivos (sensores, actuadores) capaces de realizar las tareas requeridas. En algunos casos, hay una gran variedad de sensores y en otros casos, hay pocos. La selección de los sensores apropiados para un caso de uso suele requerir que se trabaje con diferentes fabricantes y quizás con diferentes interfaces.

En este escenario, los distintos dispositivos pueden tener diferentes mecanismos de configuración, como el uso de un formulario integrado en un servidor web, la modificación de un archivo de configuración almacenado en el sistema operativo, el uso de una API REST o el envío de comandos a través de un canal de comunicación. Esto representa una dificultad añadida ya que este tipo de operaciones son difíciles de estandarizar, impidiendo que se incluyan estas características en los mecanismos de interoperabilidad.

Plataformas, aplicaciones o servicios a medida

En los casos industriales, existen numerosas plataformas que se construyen inicialmente para resolver usos específicos, como por ejemplo, las necesidades de una empresa concreta. Algunas de estas soluciones software evolucionan y se convierten en una plataforma IoT. Por ello, en la mayoría de estos casos, no

siguen estándares, buenas prácticas o metodologías, lo que dificulta la adaptación de mecanismos de interoperabilidad como los desarrollados en esta Tesis Doctoral. En ella se tiende a usar idea de que, para realizar tareas relacionadas con el IoT, los dispositivos y las plataformas deben realizar actividades comunes que puedan ser generalizadas. Sin embargo, en este escenario, de soluciones a medida, puede ser difícil encontrar estas generalizaciones, ya que algunas de las funcionalidades comunes pueden no ser necesarias en el desarrollo de estas plataformas.

Finalmente, a modo resumen, se han identificado y enumerado en la Figura 5.12 los principales temas de cada área de interés. Dicha figura proporciona una muestra representativa de los principales enfoques o problemas que pueden interesar a un nuevo usuario o futuro integrador.

Figura 5.12: Resumen completo de las lecciones aprendidas en los casos de uso.

Capítulo 6

INTER-IoT: Interoperabilidad de plataformas IoT heterogéneas

El proyecto europeo INTER-IoT [80] se inició en 2016 bajo el programa Horizonte 2020 (H2020) de la Unión Europea. El proyecto (Figura 6.1) estaba enfocado en diseñar, implementar y experimentar una solución basada en un marco de desarrollo abierto dividido en múltiples capas, una metodología asociada y herramientas para permitir la interoperabilidad entre plataformas heterogéneas de Internet de las Cosas (IoT). El objetivo del proyecto es facilitar un desarrollo eficaz y eficiente de aplicaciones y servicios de IoT inteligentes y adaptables sobre diferentes plataformas heterogéneas de IoT que abarcan uno o varios ámbitos de aplicación.

Figura 6.1: Logo proyecto INTER-IoT.

Los dominios de aplicación considerados en el proyecto son el transporte portuario y logística, con el objetivo de mejorar la eficiencia en el tiempo de transporte, reducción de la emisión de CO2 y mejora del control de acceso y la seguridad; y m-Health con el objetivo de mejorar la monitorización remota de pacientes, incrementando el número de personas que las unidades médicas pueden atender con los mismos recursos. No obstante, INTER-IoT ofrece una solución conceptualmente genérica y tiene la posibilidad de extenderse a otros dominios de aplicación mediante la metodología y las herramientas que se han desarrollado en el proyecto.

6.1. Visión general del proyecto

En la literatura, existen diferentes clasificaciones para acercarse al problema de la interoperabilidad. A estas clasificaciones se les conoce como niveles de interoperabilidad. Una de las clasificaciones más importantes de los niveles de interoperabilidad para los sistemas técnicos se llama Modelo de Niveles de Interoperabilidad Conceptual (LCIM). Define seis niveles de interoperabilidad: interoperabilidad técnica, sintáctica, semántica, pragmática, dinámica y conceptual. INTER-IoT sigue una estructura de capas similar, aunque el enfoque ha sido diferente en cuanto a la identificación de las capas, tal y como se explicó en el Capítulo 3.

En el proyecto se han diseñado, implementado y experimentado con un marco abierto de capas cruzadas, una metodología asociada y herramientas para permitir la interoperabilidad entre plataformas heterogéneas de IoT (Figura

6.2). La propuesta ha permitido el desarrollo eficaz y eficiente de aplicaciones y servicios IoT adaptativos e inteligentes, sobre diferentes plataformas IoT heterogéneas, abarcando dominios de aplicación únicos y/o múltiples. El proyecto se ha probado en dos dominios de aplicación: transporte y logística en un entorno portuario, y salud móvil. Además, se validó en un caso de uso multidominio apoyado por la integración en el proyecto de doce organizaciones o empresas colaboradoras externas. El enfoque de INTER-IoT es de propósito general y puede aplicarse a cualquier dominio de aplicación y entre dominios, en los que exista la necesidad de interconectar sistemas IoT ya desplegados o añadir otros nuevos. Además, INTER-IoT es una de las siete Research and Innovation Action (RIA) y dos Coordination and Support Actions (CSA) que componen IoT-EPI, apoyando la creación de un espacio común europeo para la interoperabilidad de IoT

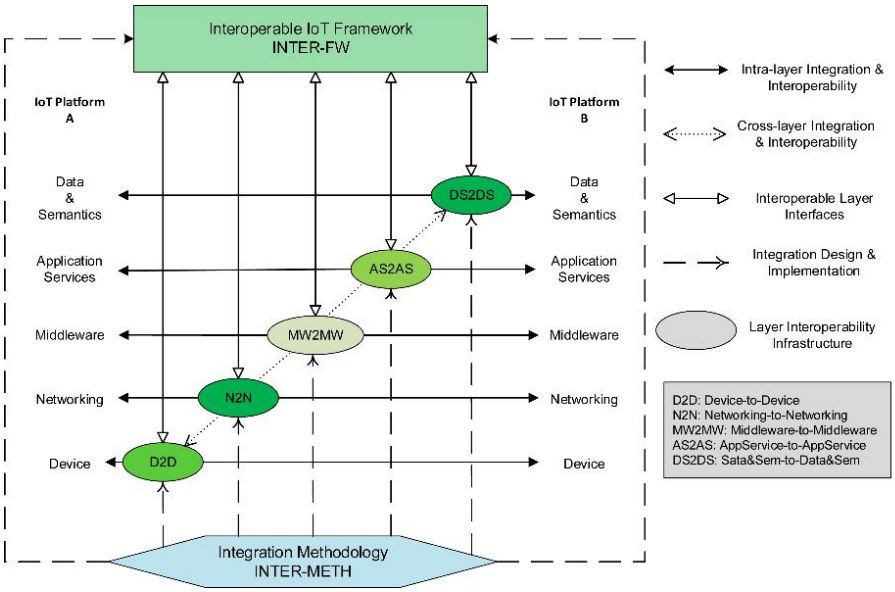


Figura 6.2: Visión general de los componentes de INTER-IOT.

INTER-IoT se basa en tres bloques principales (Figuras 6.2 y 6.3), que, a su vez, están compuestos por diferentes subcomponentes:

- **INTER-LAYER:** métodos y herramientas para proporcionar interoperabilidad entre y a través de cada capa (pasarelas/dispositivos virtuales,

red, middleware, servicios de aplicación, datos y semántica) de las plataformas IoT [178]. En concreto, se explotan:

  - D2D: Pasarelas reales/virtuales, para la comunicación dispositivo a dispositivo.
  - N2N: Conmutadores virtuales basados en SDN para la interconexión red a red.
  - MW2MW: Un meta middleware para la integración middleware a middleware.
  - AS2AS: Un broker de servicios para la orquestación de la capa de servicios.
  - DS2DS: Un mediador de semántico para la interoperabilidad de datos y semántica.
  - Capa transversal: encargada de aspectos como la seguridad, privacidad, confianza, calidad de servicio y relación entre las distintas soluciones que componen INTER-LAYER.

- **INTER-FW:** un marco global, basado en una meta-arquitectura interoperable y un modelo de metadatos, para programar y gestionar plataformas IoT interoperables, incluyendo una API para acceder a los componentes de INTER-LAYER y permitir la creación de un ecosistema de aplicaciones y servicios IoT. INTER-FW proporciona funciones de gestión específicamente dedicadas a la interconexión entre capas. La API proporcionada incluye funciones de seguridad y privacidad y apoya la creación de una comunidad de usuarios y desarrolladores [168].
- **INTER-METH:** una metodología de ingeniería basada en la herramienta CASE (Computer Aided Software Engineering) para impulsar sistemáticamente la integración e interconexión de plataformas heterogéneas de IoT no interoperables [179].

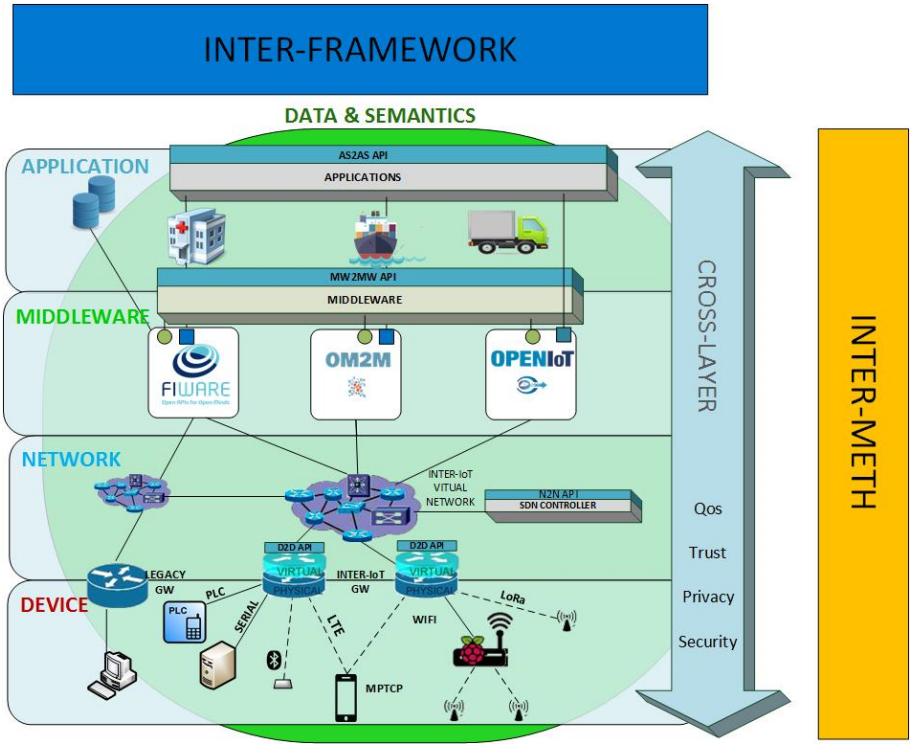


Figura 6.3: Visión general arquitectura INTER-IOT.

El enfoque de INTER-IoT se basa en casos de uso, implementados y probados en tres pilotos realistas a gran escala. En primer lugar, se probó en el puerto de Valencia, que involucra plataformas heterogéneas con alrededor de 400 objetos inteligentes. En segundo lugar, se probó en el Centro Nacional de Salud italiano para salud móvil que involucra a alrededor de 200 pacientes, equipados con redes de sensores corporales con sensores y dispositivos móviles inteligentes. En tercer y último lugar, se probó en pilotos de dominio cruzado que involucran plataformas IoT de diferentes dominios de aplicación y ampliados por la colaboración de las soluciones asociadas a las diferentes capas y subcapas de los terceros que han sido seleccionados de la convocatoria abierta de desarrollo de nuevas soluciones sobre la plataforma. Con mayor detalle, los casos de uso son:

- **INTER-LogP**: El uso de plataformas IoT en los puertos del futuro permite localizar, monitorizar y manejar diferentes equipos de transporte y

carga y zonas de almacenamiento. Este caso de uso facilita la necesidad de manejar sin problemas las plataformas IoT dentro de las instalaciones portuarias: terminal de contenedores, empresas de transporte, almacenes, transportistas por carretera, autoridades portuarias, aduanas, y fuera del puerto [180].

- **INTER-Health**: El caso de uso de la monitorización descentralizada y móvil del estilo de vida de las personas asistidas, tiene como objetivo desarrollar un sistema integrado de IoT para la monitorización del estilo de vida de las personas de forma móvil descentralizada para prevenir enfermedades crónicas. Dicho proceso de monitorización se produce de un modo descentralizado desde el centro de salud hasta los hogares de los sujetos monitorizados, y apoyado en la movilidad mediante el uso de monitores de actividad física en el cuerpo [181].
- **INTER-Domain:** compuesto por las plataformas IoT y soluciones específicas que fueron seleccionadas en una convocatoria abierta de colaboración con el proyecto. Concretamente se seleccionaron las propuestas que integraban las plataformas SENSINACT y OM2M en el marco de INTER-IoT y diez propuestas relacionadas con soluciones especificas que interactuaban con las diferentes soluciones software proporcionadas por INTER-IoT. [182].

En INTER-IoT se han analizado los requisitos proporcionados por las partes interesadas del proyecto y la usabilidad de las soluciones proporcionadas desde la perspectiva de los creadores de plataformas IoT, los propietarios de plataformas IoT, los programadores de aplicaciones IoT y los usuarios que investigan las perspectivas de negocio y crean nuevos modelos de negocio. Estos resultados han permitido poner en marcha el ecosistema INTER-IoT. Este ecosistema se compone de nuevas características y componentes: metodologías, herramientas, protocolos y APIs. La variedad y la disponibilidad cruzada de los resultados puede utilizarse para construir e integrar servicios y plataformas en diferentes capas según las necesidades de las partes interesadas y los desarrolladores. La disponibilidad de nuevos trabajos en esta línea de investigación, como la presente Tesis Doctoral, estimulará la creación de nuevas oportunidades y productos.

6.2. Componentes de la arquitectura e implementación involucradas

Los componentes de la arquitectura e implementación de la presente Tesis Doctoral involucradas en la arquitectura de INTER-IOT se corresponden con los bloques señalados en rojo en la Figura 6.4. Principalmente, estos son los subcomponentes pertenecientes al componente INTER-LAYER del proyecto, llamados:

- Produdcto software de interoperabilidad miwddleware (MW2MW)
- Producto software de interoperabilidad de aplicaciones y servicios (AS2AS)

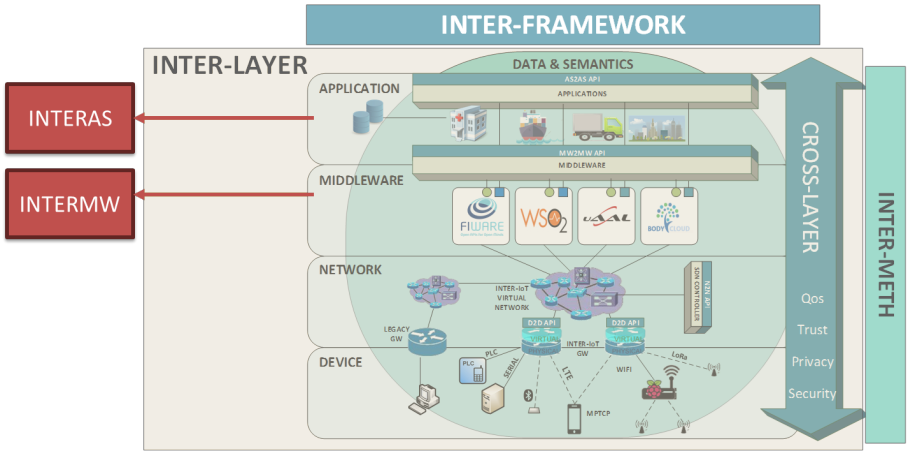


Figura 6.4: Localización solución propuesta dentro de la arquitectura INTER-IOT.

La solución propuesta está relacionada con la Capa Transversal (Cross Layer) y el marco de desarrollo (INTER-FW), puesto que son componentes necesarios para ofrecer un producto final seguro, accesible y gestionable. Desde el punto de vista de diseño e implementación, la solución de la presente Tesis Doctoral proporciona una solución enfocada y extendida en satisfacer concretamente las necesidades de los mecanismos de interoperabilidad presentados en los Capítulos 4 y 5. La solución de INTER-IoT proporciona una solución enfocada en todas las capas involucradas en el proyecto, desde dispositivo a semántica.

Se ha colaborado con la capa INTER-IoT Engineering Methodology (INTER-METH) en la documentación de metodologias de los mecanismos de la presente Tesis Doctoral que pueden servir tanto en el contexto de la misma como en el marco del proyecto INTER-IoT. Proporcionan los pasos a seguir por usuarios y desarrolladores para extender los componentes, crear nuevos conectores o utilizar los componentes.

Finalmente, el mediador semántico DS2DS es un componente externo a la presente Tesis Doctoral, que se ha realizado en el marco del proyecto INTER-IoT y del que se ha hecho uso en aquellas situaciones en las que se necesitaba para la conversión de los datos de las diferentes plataformas.

6.3. Descripción de la solución implementada

La cronología de Casos de uso proporcionada en la Sección 5.1 señala a INTER-IoT como el primer proyecto utilizado para diseñar y especificar los mecanismos de interoperabilidad presentados en la presente Tesis Doctoral. En la sección 5.2 se puede observar que la solución implementada corresponde con las bases de la solución genérica ofrecida en los Capítulos 3 y 4. Concretamente, la Figura 5.3 muestra INTER-IoT como la raíz del desarrollo de los siguientes casos de uso. La solución genérica ofrecida por los Capítulos 3 y 4 fue desarrollada en paralelo con este proyecto y luego fue extendida con nuevos conceptos derivados del resto de casos de uso.

Por tanto, la descripción de la solución implementada en INTER-IoT forma parte de las siguientes secciones de la presente Tesis Doctoral:

- INTER-LAYER:
  - Produdcto software de interoperabilidad miwddleware (MW2MW): Este producto corresponde con la arquitectura presentada en la sección 3.4.2 y la implementación ofrecida en la sección 4.2.2.
  - Producto software de interoperabilidad de aplicaciones y servicios (AS2AS): Este producto corresponde con la arquitectura presentada en la sección 3.4.4 y la implementación ofrecida en la sección 4.2.4.
- La capa tranversal de INTER-IoT corresponde con la arquitectura e implementación presentada en la secciones 3.4.5 y 4.3. Allí se describen los mecanismos habilitadores de la interoperabilidad y las relaciones entre mecanismos que cubren las capas cubiertas por la presente Tesis Doctoral.

- El marco de interoperabilidad corresponde con la arquitectura e implementación presentadas en la secciones 3.5 y 4.4. Dichas secciones ponen el foco en las capas objeto de la presente Tesis Doctoral.

Finalmente, los otros mecanismos de interoperabilidad presentados en el Capítulo 3 correspondientes con las secciones 3.4.1 y 3.4.3 no forman parte de la arquitectura de INTER-IoT. Su desarrollo e implementación tuvo su incio en herramientas y soluciones desarrolladas para extender las soluciones proporcionadas por INTER-IoT. Algunas de ellas se desarrollaron exclusivamente en el marco del proyecto o en colaboración con los otros casos de uso presentados en esta Tesis Doctoral.

6.4. Validación y resultados

Partiendo de la descripción de la solución implementada, en esta sección se presentan los principales resultados y pruebas de validación obtenidos respecto a cada mecanismo de interoperabilidad.

Relacionados con la interoperabilidad en la capa Middleware

Respecto a este mecanismos de interoperabilidad se describen los puentes para conectar con las plataformas y la solución de interoperabilidad.

Puentes

Durante el proyecto se realizaron puentes para garantizar la compatibilidad con once plataformas IoT: Azure, SEAMS2, universAAL, FIWARE, OM2M, e3city, BodyCloud, OpenIoT, IoTivity, SOFIA2 y openHAB.

Las principales funcionalidades que se deben cubrir, tal y como se ha indicado en los dos capítulos anteriores, son suscripciones, actuaciones, gestión de dispositivos virtuales y descubrimiento. Se ofrece una arquitectura abierta para el desarrollo de puentes de plataformas IoT, compuesta por:

- Interfaz Java común. MW2MW proporciona una interfaz de puente que define las características comunes de los puentes que deben implementarse: suscripciones, actuaciones, gestión de dispositivos virtuales y descubrimiento. Las anotaciones de Java se utilizan en combinación con los mecanismos de reflexión de Java para cargar dinámicamente los puentes en tiempo de ejecución.

- Conversión sintáctica. Un paso importante en el desarrollo de puentes es la implementación de un traductor sintáctico hacia/desde el formato específico de la plataforma. Se proporcionan traductores sintácticos

genéricos con ejemplos y algunos formatos comunes, pero, en principio, debería proporcionarse un nuevo traductor para cada tipo de plataforma IoT.

- Traducción semántica. La traducción semántica la realiza el IPSM en caso de ser necesaria.
- Pruebas unitarias y de integración. Se ha proporcionado una serie de pruebas unitarias y de integración para facilitar el desarrollo de puentes. Se puede generar una serie de llamadas a la API para probar las implementaciones de los puentes.
- Ejemplos y documentación. Se proporciona documentación y ejemplos para los desarrollos.

Un puente debe implementar todos los métodos abstractos definidos en la clase abstracta común de todos los puentes:

- Registrar plataforma: Registra la plataforma especificada en el INTER-MW y crea una instancia de puente para la plataforma.
- Anular el registro de la plataforma: Anula el registro de la plataforma especificada.
- Suscribir: Suscribe al cliente a las observaciones proporcionadas por la plataforma para el dispositivo especificado.
- Anular la suscripción: Cancela la suscripción especificada creada por la suscripción.
- Consulta: Realiza una consulta sobre el estado y la última observación realizada por un dispositivo especificado.
- Listar dispositivos: Realiza una consulta a una plataforma para obtener todos los dispositivos gestionados por ella, que considere detectables
- Crear dispositivo en plataforma: Instrucción para que la plataforma de interoperabilidad comience a gestionar un nuevo dispositivo.
- Actualizar dispositivo en plataforma: Instrucción para que una plataforma de interoperabilidad actualice la información sobre un dispositivo que está gestionando.
- Eliminar dispositivo en plataforma: Instrucción para que la plataforma de interoperabilidad deje de gestionar un dispositivo.

- Observar: Envía un mensaje de observación desde la solución de interoperabilidad a la plataforma IoT. El puente actúa como editor para la plataforma.
- Actuar: Igual que Observar, salvo que la carga útil del mensaje contiene instrucciones de actuación.
- Error: Envía información sobre cualquier error ocurrido dentro del puente o dentro de INTER-MW.
- No reconocido: Tipo de mensaje personalizado que no ha sido reconocido por la solución de interoperabilidad.
- Descubrimiento de dispositivos: Con el descubrimiento de dispositivos el puente de la plataforma puede comunicar todos sus dispositivos con la solución de interoperabilidad.

El principal desafío de la integración de la plataforma IoT es el desarrollo de puentes específicos de plataforma. Este proceso se ha perfeccionado a lo largo del ciclo de vida del proyecto proporcionando documentación adicional, ejemplos y pruebas de integración para facilitar el desarrollo en la mayor medida posible. La complejidad del desarrollo de puentes para una plataforma específica depende del nivel de estandarización proporcionado por la propia plataforma. Las plataformas que proporcionan APIs estándar de la industria bien documentadas son mucho más fáciles de integrar. Por ejemplo, para una plataforma que proporciona una API REST con formato de datos JSON o XML bien documentado, se tendrían un conjunto de ayudas y pautas, mientras que para una plataforma que proporciona alguna interfaz propietaria, o incluso sólo acceso a la base de datos, se requeriría mucho más esfuerzo. Los principales desafíos son, por supuesto, las plataformas cerradas que no proporcionan ninguna API documentada. Es muy difícil garantizar un sistema estable y fiable construido sobre una fuente de datos no documentada.

Cuando se implementa un puente para conectar una plataforma IoT es necesario conocer la plataforma y las funcionalidades que ofrece. Por tanto, es una tarea muy importante analizar qué métodos se pueden utilizar para acceder a estas funciones. En aquellos casos en los que se detecta que alguna funcionalidad no está disponible, es necesario realizar una función o componente que se encarga de realizar las acciones necesarias para obtener la información o realizar las acciones deseadas. No ofrecer dicha funcionalidad es la última opción. Puede implicar crear un módulo para implementar una forma de acceder a esa función en la plataforma o agregar un componente en los puentes.

Una plataforma que no ofrece una funcionalidad esperada no es una situación muy común, porque todas las plataformas suelen cubrir las necesidades

básicas que requieren los puentes. Sin embargo en algunos casos se han requerido algunos pequeños ajustes. Un ejemplo concreto de una plataforma que no implementa mecanismos directos para soportar algunas funcionalidades se puede explicar con el ejemplo de FIWARE. Esta plataforma almacena el estado actual y la información de contexto de los dispositivos, pero en ese caso esta plataforma no implementa un método directo para realizar la actuación en sus sensores usando Orion Context Broker. Para solucionarlo, se han realizado algunos ajustes. A nivel de plataforma existe la posibilidad de utilizar agentes o crear suscripciones a la información de los sensores para desarrollar scripts que realicen la actuación cuando la suscripción notifique un cambio de estado del sensor. A nivel del puente la forma de realizar esa acción es la misma que enviar una observación al puente. En el caso de plataformas con funcionalidades pendientes o no mantenidas, se han utilizado sus foros para buscar una solución, contactar a los principales desarrolladores de la plataforma o abrir incidencias en los repositorios. Estas plataformas suelen tener poca documentación y son difíciles de seguir.

Los principales aspectos clave a considerar son:

- Las funciones de interoperabilidad definidas en los puentes siempre deben ser obligatoriamente implementados. Sin embargo, estos métodos pueden devolver que la función no es compatible. Desde el punto de vista de la programación, la interfaz de interoperabilidad se propaga al puente. Pero permitir devolver que no es soportado brinda flexibilidad para enfrentar escenarios reales.

- Algunas funciones, no presentes en las plataformas, se pueden emular realizando una programación adicional en el lado del puente.

- Si la plataforma está en desarrollo o se mantiene activamente, se envía una solicitud de cambios al equipo de desarrollo.

- Algunas características son obligatorias debido a la arquitectura del mecanismo de interoperabilidad entre plataformas middleware. Si estos no se pueden emular, entonces el puente no es posible y la plataforma no se considera compatible con la solución.

- Es importante proporcionar mecanismos simples pero potentes de interoperabilidad para tantas plataformas como sea posible. Las estrategias seguidas aseguran un elevado número de plataformas compatibles y también identifican aquellas plataformas que por su construcción se alejan de los estándares y deben ser consideradas un caso especial. Algunos ejemplos de casos reales son:

- FIWARE Orion no admite el método de actuación: aunque Orion no es una plataforma industrial, carece de mecanismos de actuación. Este método se implementó en algunos casos como que es una funcionalidad no admitida y en otros añadiendo un script intermedio.
- El descubrimiento de sensores no es posible en la plataforma SEAMS2. Esto se emula en INTER-IoT enumerando los sensores y enviándolos a INTER-MW.
- El mecanismo de suscripción no es compatible con Bodycloud. Se emula realizando un sondeo de datos periódico desde el lado del puente hasta la plataforma.

A modo de experimentación final se recogen algunas experiencias con las plataformas:

- El puente para FIWARE fue el primer puente que se implementó, y se ha utilizado como prueba de concepto para la definición de una arquitectura INTER-MW adecuada. El resultado ha sido una clara división de la parte sintáctica y semántica para la transformación de datos.
- Al desarrollar el puente BodyCloud, el principal problema fue el de lidiar con una plataforma distribuida (instalada en teléfonos inteligentes) con múltiples puntos finales y direcciones IP. Para encajar en una amplia arquitectura INTER-MW, se desarrolló un proxy BodyCloud separado para proporcionar un único punto final HTTP para el puente INTER-MW. Este enfoque también ayudó a simplificar las reglas de acceso a la red del servidor, fortaleciendo así la seguridad general del sistema.
- En el puente de plataforma en la nube Sofia2, se encontró el problema de tres versiones diferentes de las instancias de la misma plataforma. Se utilizaron principalmente las APIs estándar que brinda la plataforma, pero para las instancias más personalizadas de esta plataforma. Por ejemplo, una instancia de una institución pública, utiliza APIs personalizadas por lo que es necesario readaptar el código del puente.
- La plataforma de IoTivity admite el intercambio y el control de la información en función de la mensajería a través del protocolo CoAP. Se tuvieron que desarrollar dos módulos para el puente de IoTivity. El primero es el traductor sintáctico que utiliza el mismo procedimiento seguido para otros puentes. El segundo es el cliente COAP responsable de la comunicación con el servidor de IoTivity. Cuando el puente recibe una solicitud, el traductor la traduce al formato de IoTivity y luego la solicitud se envía al servidor de IoTivity a través del cliente CoAP.

Se han proporcionado algunos ejemplos más representativos de los desafíos de integración que enfrentamos, pero, de hecho, con cada nuevo puente que se desarrolla hay un detalle nuevo e imprevisto que debe resolverse. El enfoque abierto de los puentes ha permitido encontrar siempre una solución eficiente para la integración. Finalmente, es importante resaltar la aproximación a seguir cuando se quiere realizar un puente para una plataforma. Los pasos son los siguientes:

- Si no se ha desarrollado previamente un puente para esta plataforma: En este caso es muy interesante realizar un estudio preliminar de la plataforma para encontrar las posibles modificaciones que puedan ser necesarias en el futuro. Una buena planificación en ese momento implicará menos cambios en el futuro. Se recomienda crear una clase relacionada con las utilidades de la plataforma, para acceder a las funcionalidades concretas que ofrece la misma, indicando a qué versión de dicha plataforma corresponde este puente.
    - El puente desarrollado para FIWARE es el ejemplo de una plataforma con un puente desarrollado considerando las diferentes versiones que pueden aparecer en el futuro. En este caso se tuvo en cuenta que la versión actual de la plataforma es Orion NGSI API v2. Luego se creó una clase llamada utils en la que se incluyeron todas las características de esta versión. Por lo tanto, si apareciera una versión posterior, se podría crear una nueva clase de utilidades.
- Si existe un puente desarrollado previamente para esta plataforma, pero corresponde a una versión diferente: Si durante la realización del primer puente se realizaron los pasos indicados en el apartado anterior, entonces, es posible crear una clase de utilidades correspondiente a esta versión y adaptar aquellas funcionalidades que se hayan visto afectadas por el cambio de versión. Eso implica menos esfuerzo en el desarrollo del puente. De lo contrario, si el desarrollador del primer puente no ha tenido en cuenta estos cambios, el desarrollo de un nuevo puente puede implicar tener que crear una nueva versión del puente adaptada a esta versión de la plataforma.
    - Durante el desarrollo del puente para Sofia2 Cloud Platform. Hubo acceso a tres versiones de la plataforma en la nube. El primero fue el laboratorio en la nube, una instancia de prueba abierta a la comunidad pública de desarrolladores. El segundo fue una versión de la plataforma ubicada en una empresa de telecomunicaciones que colabora con una entidad pública. El tercero fue un despliegue ubicado en una entidad pública, con grandes medidas de seguridad y al que

no se les permitió el acceso a terceros. Debido a la disponibilidad de instancias, el puente se desarrolló en primer lugar para la versión del laboratorio en la nube. Se tuvo en cuenta la información indicada en el primer apartado del punto anterior y se creó una clase útil para la plataforma. Esto facilitó adaptar la versión de la empresa de telecomunicaciones, ya que requería cambios mínimos. Por lo tanto, podrían usar el mismo puente. En cambio, la versión desarrollada para conectar con la instancia pública requirió cambios profundos y un esfuerzo extra considerable en el desarrollo.

- Si un desarrollador quiere desarrollar un puente para la plataforma, pero la instancia de la plataforma no tiene la misma implementación que la instancia de la plataforma que tiene un puente previamente desarrollado: en estos casos suele ocurrir que es la misma plataforma e internamente tiene el mismo funcionamiento, pero las interfaces para acceder a las funcionalidades son completamente diferentes. En estos casos puede implicar el desarrollo de un nuevo puente para adaptarse a las nuevas interfaces.

  - En relación con la plataforma Iotivity, los desarrolladores deben crear funcionalidades adicionales en la instancia de la plataforma para proporcionar una API REST para conectarse con la plataforma. Este desarrollo se hizo enfocado en una instancia concreta de la plataforma. En parte, fue influenciado porque en algunos aspectos la plataforma tiene similitudes con un SDK. Por esa razón, el puente desarrollado depende más de esta instancia concreta de la plataforma y es muy difícil reutilizar el puente para otra instancia de la misma.

INTER-MW

La solución de interoperabilidad entre plataformas Middleware, en líneas generales, proporciona la arquitectura e implementación descrita en el presente capítulo y en los capítulos 3 y 4. Los principales resultados son:

- Se ha proporcionado un API REST abierta para acceder a las funcionalidades. Las principales funcionalidades son el registro de dispositivos, el mecanismo de descubrimiento, las observaciones, actuaciones y la gestión de dispositivos virtuales.
- Las suscripciones se pueden recibir mediante métodos push o pull.
- Los resultados se pueden obtener en un formato JSON simple, que cumple la mayoría de los requisitos básicos del usuario, o en el formato JSON-LD

más complejo que también ofrece un conjunto más rico de funciones y una interoperabilidad semántica completa.

- Un kit de desarrollo de puentes con ejemplos y test de integración.
- Aproximadamente, una latencia en el acceso a las plataformas menor de 20 ms.
- Un rendimiento capaz de soportar 120 mensajes por segundo.
- Integración con el gestor de APIs, el gestor de identidades y el marco de interoperabilidad.

Un ejemplo de valor añadido de la solución proporcionada en este caso de uso fue presentado en la IoT Week de 2017 en el que se implementó una aplicación de demostración para probar y demostrar la aplicabilidad en la práctica (Figura 6.5). La configuración de demostración que se muestra conecta dos plataformas de middleware (UniversAAL y FIWARE) y permite la interoperabilidad entre ellas. Además de los componentes de interoperabilidad se contó con:

- Una báscula digital habilitada para Bluetooth que está emparejada con un teléfono inteligente y envía valores de pesaje.
- Un teléfono Android con una aplicación que obtiene valores de pesaje de la báscula y los reenvía a través de su aplicación universAAL a otras instancias universAAL en la red local.
- Wirecloud. Una plataforma mashup de aplicaciones web proporcionada por los habilitadores de FIWARE.

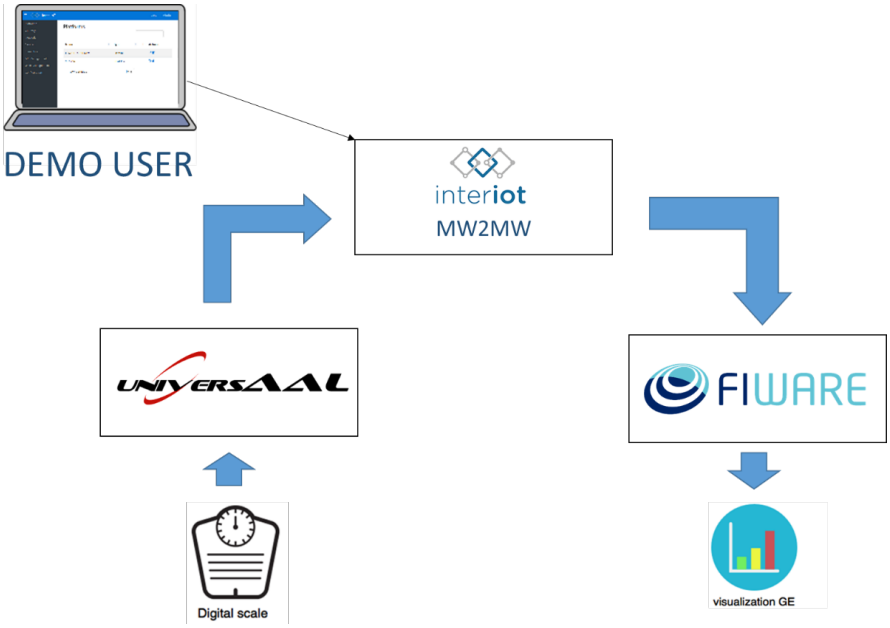


Figura 6.5: Visión general de la demostración con las plataformas univerSAAL y Fiware.

La demostración consiste en que los pacientes tienen en su casa una báscula conectada a la aplicación móvil que está conectada a UniversAAL. Los datos de la báscula conectada a la plataforma universAAL son conectados a la solución de interoperabilidad middleware. La plataforma FIWARE, ubicada en el hospital, se suscribe al sensor mediante la solución de interoperabilidad middleware y muestra los resultados de cada lectura en su aplicación, que forma parte de la plataforma FIWARE, capaz de ofrecer cuadros de mando y mashup de aplicaciones. Por tanto, esta solución suple la falta de una interfaz gráfica por parte de UniversAAL. La Figura 6.6 enfatiza los dominios involucrados en esta demostración.

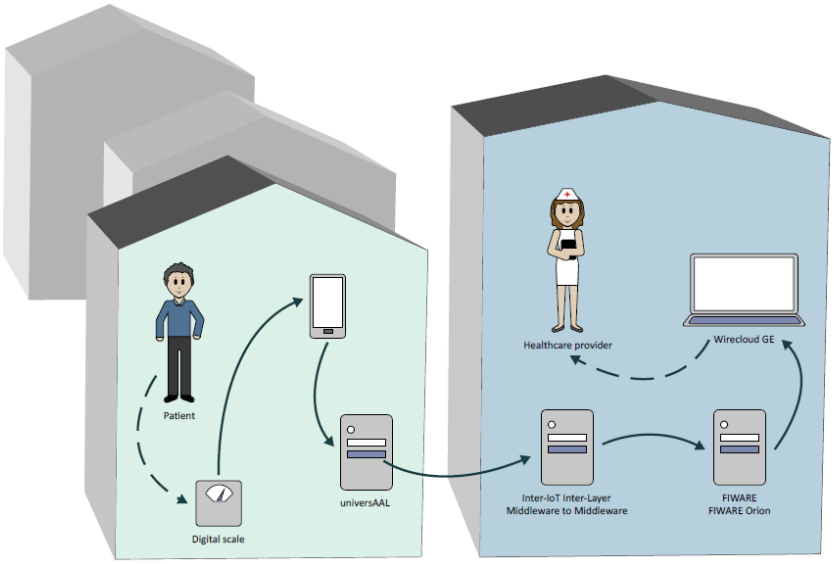


Figura 6.6: Dominios de la demostración con las plataformas universAAL y FIWARE.

La demostración presenta los pasos básicos, como la implementación de diferentes puentes, gestión de suscripciones, enrutamiento de mensajes, uso de servicios de registro, descubrimiento y gestión de API.

El siguiente capítulo, dedicado al caso de uso ACTIVAGE, enfatizará en otros escenarios para demostrar las ventajas que puede ofrecer el uso de esta solución.

Relacionados con la interoperabilidad en la capa de Aplicación y Servicios

Se crearon nodos con servicios, tales como, almacenamiento a corto plazo, consulta de datos históricos, procesador de eventos complejos, herramientas de análisis, herramientas de notificación, herramientas de envío de correos, herramientas de visualización herramientas de transformación de datos, herramientas de actuación, mapas y cuadros de mando.

Se crearon nodos para acceder a los servicios disponibles en las plataformas exploradas en el proyecto como FIWARE, SOFIA2, OM2M, Azure, universAAL, OpenHAB o Azure.

A su vez se dejaron flujos predefinidos para crear ejemplos de flujos, tales como, notificación de resultados en cuadro de mando, visualización en mapa, creación de API, consulta combinada a datos históricos o suscripciones a diferentes plataformas.

En la Figura 6.7 se muestra un ejemplo de las principales tecnologías utilizadas, una parte de las plataformas involucradas y sus servicios y nodos.

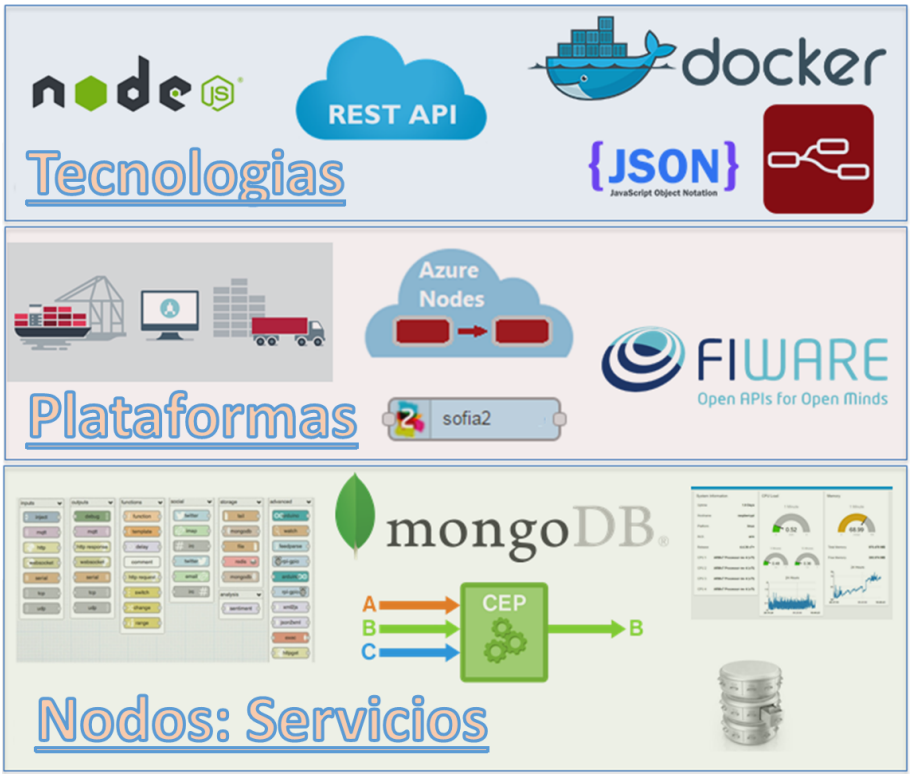


Figura 6.7: Tecnologías, plataformas y nodos utilizados en AS2AS.

Un ejemplo de valor añadido de la solución uso también fue presentado en la IoTweek de 2017 en el que se implementó una aplicación de demostración para probar y demostrar la aplicabilidad en la práctica. La demostración pressenta el uso de la herramienta para la creación de flujos. Son una colección de nodos conectados entre sí para intercambiar mensajes.

La demostración comienza con la posición y los datos de un camión, tal y como se muestra en la Figura 6.8.

Figura 6.8: Datos de los camiones en la demostración AS2AS.

Esta información se obtiene de las plataformas (FIWARE y SOFIA2) que monitorizan la posición de los dispositivos que están presentes en los vehículos, tal y como se muestra en la Figura 6.9.

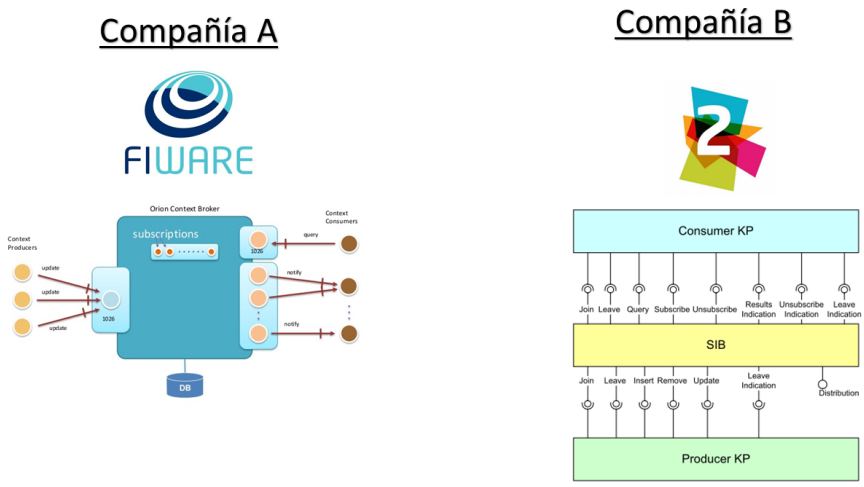


Figura 6.9: Plataformas involucradas en la demostración AS2AS.

Automáticamente cuando se cumpla una regla, en la terminal del puerto, recibirán información del camión, la escala hacia donde se dirige e información de hechos históricos de dicho camión. La información se presenta a través de mapas, tableros y paneles ofrecidos por un servicio de duadro de mando, tal y como se refleja en la Figura 6.10.

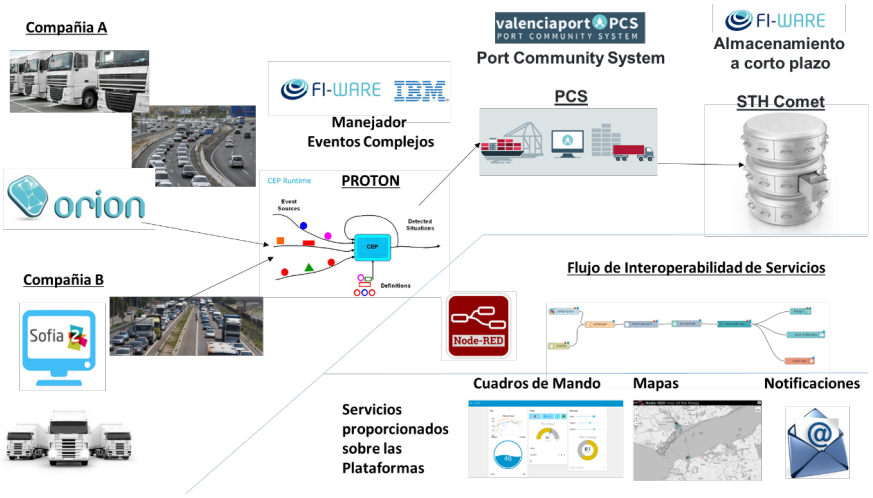


Figura 6.10: Flujo de interoperabilidad de la demostración AS2AS.

Los nodos brindan acceso a servicios nativos y facilitan un acceso simple y visual a las funciones que ofrecen dichos servicios. Los nodos utilizados son los siguientes:

Manejador eventos complejos (CEP) Proton

Proporciona las siguientes acciones:

- Analiza datos de eventos en tiempo real.
- Genera conocimiento inmediato.
- Permite una respuesta instantánea a las condiciones cambiantes

En esta demostración, el CEP activa una acción cuando se detecta una regla. Es responsable de activar la ejecución del flujo. El nodo Proton proporciona las siguientes formas de interactuar con la aplicación Proton CEP:

- Envío de eventos.
- Gestión del repositorio de definiciones.
- Administrar instancias.

Servicio de llamada al puerto

Los usuarios de PCS tienen acceso a las consultas del Servicio de llamada al puerto, que proporcionan información en tiempo real sobre las llamadas al puerto de embarcaciones planificadas y actuales. En esta demostración, con los datos del CEP, un nodo Nodo-RED realiza la llamada al PCS, obteniendo los datos de la llamada al puerto desde el PCS mediante el servicio de recuperación de datos de la llamada al puerto.

El nodo PCS proporciona dos métodos para acceder a la información de llamada al puerto:

- Obtener los datos de llamada al puerto, ofrecidas por el PCS, a partir de su número de identificación.
- Obtener una lista de llamadas al puerto, ofrecidas por el PCS, en un rango de fechas.

Histórico a corto plazo

Gestiona la información de contexto histórica cruda y agregada de las series temporales sobre la evolución en el tiempo de los datos de contexto. En esta demostración, el flujo hace una llamada al Nodo Histórico para obtener información sobre la evolución en el tiempo de los datos de contexto. El nodo de acceso a la aplicación Historico a Corto Plazo (STH) obtiene los valores históricos de un nodo. En esta demostración, recoge los ID de la última llamada de un camión cuando está dentro de la zona portuaria.

Nodos del panel de control

Estos nodos proporcionan, entre otras, las siguientes funciones en la demostración:

- La creación rápida de un tablero de datos en vivo.
- Enviar y recibir correos electrónicos, con credenciales válidas para un servidor de correo electrónico.
- Una página web con un mapa para trazar cosas.

Partiendo de los nodos descritos, los pasos necesarios para la composición del servicio son los siguientes:

- CEP FIWARE (Proton) recibe la información de las empresas de camiones.
- CEP calcula la distancia de los camiones a un punto definido.

- Activa una alerta cuando el camión está más cerca de 10 kilómetros. Proporciona el ID de la llamada (número de llamada), el ID del camión y la posición (longitud, latitud).
- Obtiene los datos de escala de PCS
- Obtiene los datos históricos del STH
- Muestra los datos a la Terminal Portuaria a través de mapas, paneles o correos electrónicos.

La Figura 6.11 ilustra las diferentes compañías involucradas: las compañías de transporte, la autoridad portuaria y la terminal de contenedores. Orquestando todo está el usuario que gestiona el flujo de interoperabilidad.

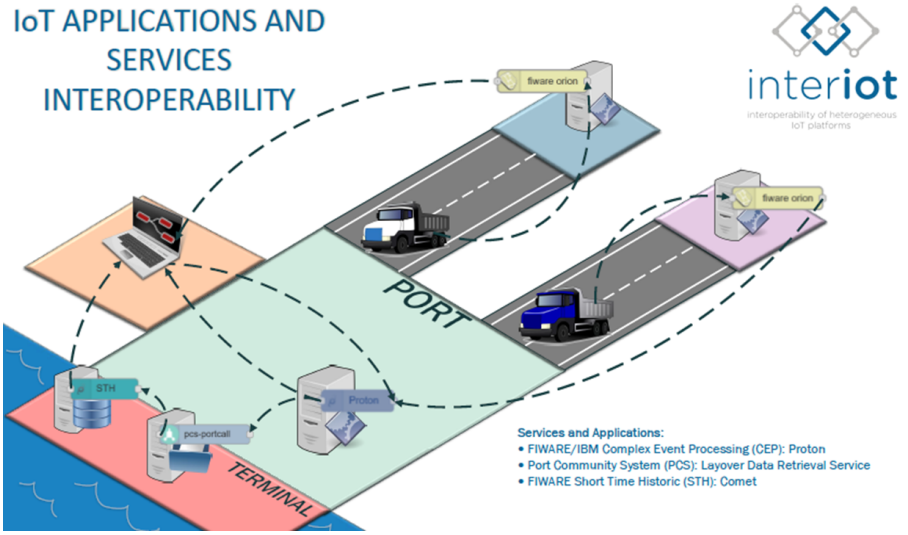


Figura 6.11: Compañías involucradas en la demostración AS2AS.

La Figura 6.12 muestra la representación de los datos que se muestran en tiempo real mediante el servicio de visualización diseñado.

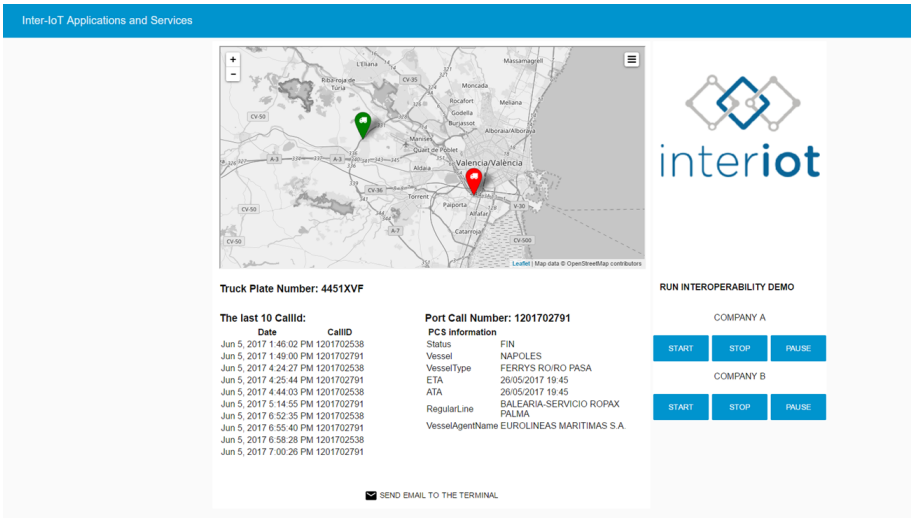


Figura 6.12: Interfaz gráfica de la demostración AS2AS.

Esta captura de pantalla del servicio de visualización muestra un camión dentro del área indicada (menos de 10 km). Dicho camión tiene un color rojo y la información obtenida de los servicios se muestra en el panel (la información de la llamada portuaria y los últimos números de llamada portuaria visitados) y se envía un correo electrónico a la terminal informando de que está llegando. El otro camión tiene un color verde porque está fuera de la zona indicada y por lo tanto, la única información que tenemos de él es su posición.

Relacionados con los componentes habilitantes y el marco de interoperabilidad

El marco de interoperabilidad y su integración con los componentes habilitantes ofrece:

- Un diseño global, usabilidad y unificación de los mecanismos de interoperabilidad.
- Proporciona acceso a las APIs, configuración, monitorización y gestión.
- Facilita las tareas de extensión y escalabilidad. Un marco para el desarrollo.
- La integración con la seguridad y la privacidad.

- Control total de las operaciones de la API y del marco por parte del propietario de la plataforma para mantener la soberanía de los datos.
- Administración de plataformas con y sin interfaz de usuario para que el usuario no tenga que hacerlo.
- Proporciona funciones de gestión visual para plataformas que carecen de ellas (FIWARE, WSO2, OneM2M).
- El desarrollo de aplicaciones que aprovechen los datos de múltiples plataformas heterogéneas.
- La construcción de aplicaciones basadas en una única API, evitando la repetición de tareas.

La Figura 6.13 ofrece una captura de la herramienta con algunas plataformas conectadas.

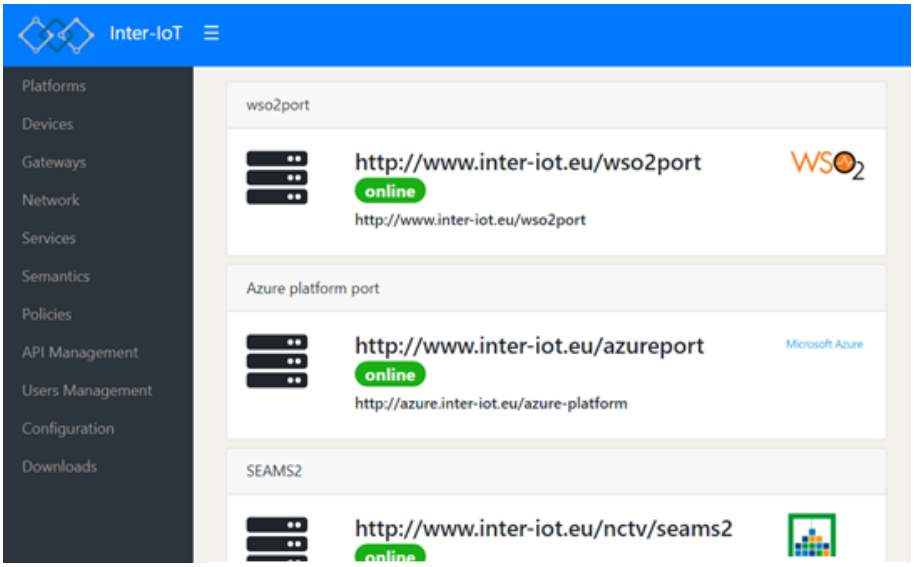


Figura 6.13: Captura ventana plataformas IoT del marco de interoperabilidad.

La Figura 6.14 ofrece una visión de las facilidades para acceder a la API como el del componente de interoperabilidad de plataformas middleware.

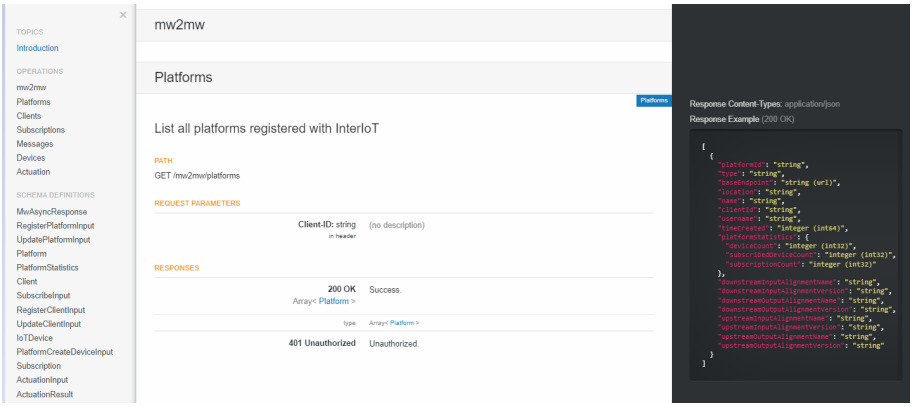


Figura 6.14: Captura ventana APIs de los mecanismos de interoperabilidad.

Para ilustrar los mecanismos de gestión de múltiples instancias desde el marco de interoperabilidad se describe el menú de Aplicaciones y Servicios que está compuesto por tres componentes: Gestor de Instancias, Repositorio de Flujos y Repositorio de Nodos (Figura 6.15).

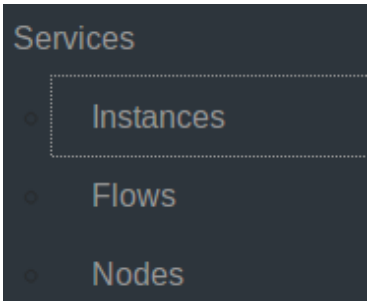


Figura 6.15: Captura ventana del gestor de instancias.

El gestor de instancias (Figura 6.16) permite:

- Mostrar una lista de las instancias disponibles.
- Mostrar el estado de las instancias (iniciadas, paradas...).
- Una descripción de cada instancia en ejecución.

Si se pulsa en una instancia se accede a otro menú, en el que se puede ver:

- Información completa de esta instancia.
- Ir al navegador en ejecución de Node-RED, para trabajar con esta instancia de NODE-RED.
- Proporcionar más información, como la dirección de la UI de la imagen de Node-RED, si existe.
- Puede gestionar la instancia. Por ejemplo, se puede iniciar o detener una instancia, además de ejecutar, cargar flujos de trabajo, solicitar una instancia y gestionar los usuarios.
- Eliminar la instancia.

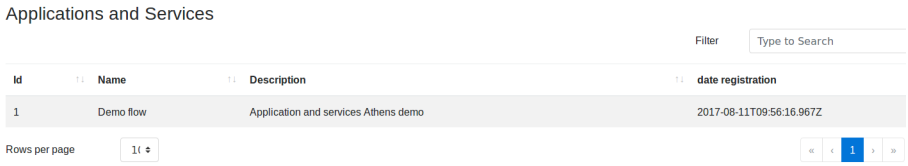


Figura 6.16: Captura ventana del listado de instancias.

El repositorio de flujos y nodos (Figura 6.17):

- Muestra una lista de los flujos y nodos disponibles.
- Puede seleccionar un flujo o un nodo y obtener más información.
- Puede crear un nuevo flujo o un nodo, describirlo y cargar el archivo JSON del flujo o la carpeta con los archivos para desplegar el nodo.

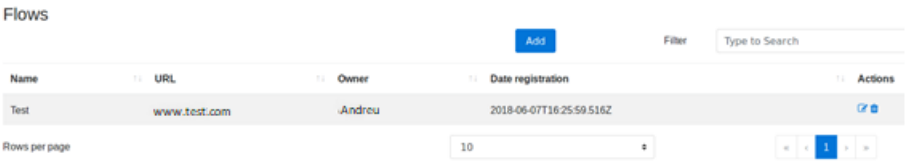


Figura 6.17: Captura ventana del listado de flujos.

Si se pulsa en un flujo se accede a otro menú (Figura 6.18), en el que se puede ver:

- Información y descripción del flujo o nodo.
- Actualizar esta información.
- Puede pulsar un botón y desplegar el flujo en una nueva instancia de Node-RED. Entonces, esta instancia va a ser accesible en el Gestor de Instancias.

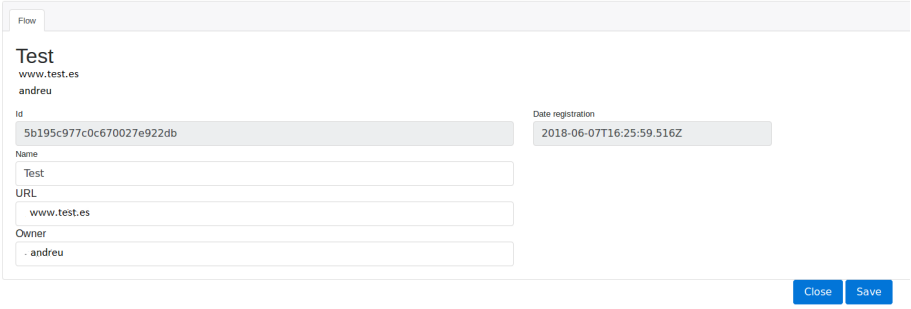


Figura 6.18: Captura ventana información de flujo de interoperabilidad.

Como información adicional, estos mecanismos de AS2AS funcionan sobre un servicio Web, que conecta con Kubernetes para la gestión de instancias, que permite:

- Ver si una instancia esta disponible.
- Crear una nueva instancia.
- Crear una instancia con un flujo de interoperabilidad predeterminado.
- Listar las instancias disponibles.
- Listar la información de las instancias disponibles o de una instancia en concreto
- Eliminar una instancia.
- Instalar un nodo en la instancia
- Borrar un nodo de una instancia.
- Cargar un flujo en una instancia activa.

Además del uso mediante el marco de interoperabilidad descrito anteriormente, el estado de las instancias se puede observar en Kubernetes (la tecnología que permite las múltiples instancias), tal y como se muestra en la Figura 6.19.

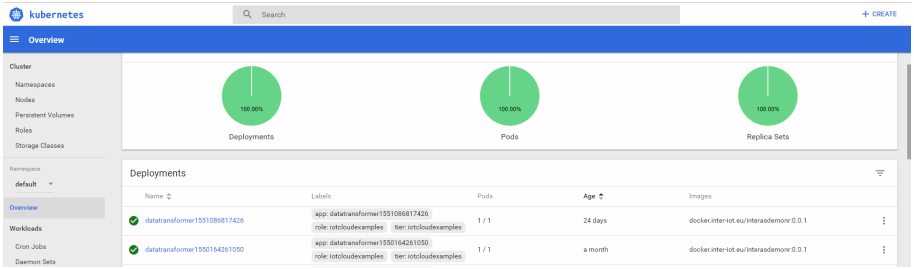


Figura 6.19: Captura ventana información de instancias en Kubernetes.

Estos mecanismos, además de mediante el marco de interoperabilidad, se pueden utilizar mediante llamadas a la API, como se muestra en la Figura 6.20, donde se ofrece el resultado devuelto de listar las instancias del mecanismo de interoperabilidad disponibles:

GET [localhost]:3004/datatransformer/list

Pretty Raw Preview JSON


```
[
    {
        "_id": "5c6a9def3eae711592dca64f",
        "tenantId": "Review_1",
        "description": "Instance for review purposes",
        "name": "Instance Review",
        "location": "http://192.168.1.183:30070",
        "createdAt": 1550491118946,
        "datatransformerId": "datatransformer1550491118946",
        "port": 1880,
        "url": "http://192.168.1.183:30070",
        "__v": 0
    },
    {
        "_id": "5c6aa2993eae711592dca650",
        "tenantId": "Review_2",
        "description": "Test services",
        "name": "Instance Services",
        "location": "http://192.168.1.183:30397",
        "createdAt": 1550492313456,
        "datatransformerId": "datatransformer1550492313456",
        "port": 1880,
        "url": "http://192.168.1.183:30397",
        "__v": 0
    },
    {
        "_id": "5c6aa3cb3eae711592dca651",
        "tenantId": "Review_3",
        "description": "Data from Platforms",
        "name": "Instance Data",
        "location": "http://192.168.1.183:30823",
        "createdAt": 1550492619177,
        "datatransformerId": "datatransformer1550492619177",
        "port": 1880,
        "url": "http://192.168.1.183:30823",
        "__v": 0
    },
    {
        "_id": "5c6d7798e4b0463b443ee224",
        "tenantId": "FlowForReview",
        "description": "This",
        "name": "ForReview",
        "location": "http://192.168.1.183:30905",
        "createdAt": 1550677912243,
        "datatransformerId": "datatransformer1550677912243",
        "port": 1880,
        "url": "http://192.168.1.183:30905",
        "__v": 0
    }
]
```


Figura 6.20: Captura ventana información de la API del gestor de instancias.

6.5. Alcance de la solución dentro del proyecto

La solución y arquitecturas propuesta en esta Tesis Doctoral tienen el siguiente alcance y relación con INTER-IoT:

- Interoperabilidad Middleware:
  - Análisis pormenorizado de cada plataforma.
  - Diseño e implementación de mecanismos de acceso y gestión de cada plataforma. Por ejemplo, permitió un conocimiento pormenorizado de FIWARE, que se utiliza posteriormente en los casos de uso de los proyectos ACTIVAGE y DataPorts.
  - Interoperabilidad de Plataformas Middleware:
    - Despliegue y configuración de Plataformas IoT Middleware.
    - Diseño e implementación de la solución y validación en casos de uso.
- Interoperabilidad de Aplicaciones y Servicios:
  - Análisis pormenorizado de servicios y aplicaciones de cada plataforma.
  - Diseño e implementación de mecanismos de acceso y gestión de cada plataforma.
- Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos:
  - Despliegue y configuración de servicios y aplicaciones de plataformas IoT.
  - Diseño e implementación de la solución y validación en casos de uso.

Finalmente, es interesante indicar que el trabajo realizado se plasmó en presentaciones en eventos, congresos y demostraciones del proyecto. Por ejemplo, se realizaron diferentes presentaciones en la IoT Week de 2017 [183].

Además, ha dado lugar a diferentes publicaciones por parte del presente autor. Las principales son las siguientes:

- *Solución de interoperabilidad mediante programación basada en flujos para aplicaciones de plataformas IoT* [127]

  Ofrece una visión completa de la especificación y diseño de la solución de interoperabilidad de aplicaciones y servicios mediante una solución de programación orientada a flujos. Existen varias soluciones que utilizan este paradigma dentro de las plataformas de IoT, pero ninguna está enfocada principalmente en el objetivo de conectar servicios desde diferentes plataformas de IoT. Para abordar esto, se describe una metodología para desarrollar funcionalidades de acceso a los servicios, y se proporciona una arquitectura con diferentes componentes para ofrecer una solución a este problema de interoperabilidad. Esta solución ofrece ventajas en el registro, catalogación y descubrimiento de servicios, y en la creación y gestión de Servicios IoT compuestos. Finalmente, este artículo muestra un caso de uso relacionado con el transporte y la logística para validar este enfoque.

- *Explotación de datos de IoT y servicios de ciudades inteligentes para la monitorización de factores de riesgo de enfermedades pulmonares obstructivas crónicas* [184]

  Ofrece una visión especifica en el uso de los servicios IoT y el dominio de las ciudades inteligentes para destacar los mecanismos de interoperabilidad de aplicaciones y servicios. Existen pocos sistemas de IoT en la literatura actual centrados en la monitorización y gestión de pacientes con Enfermedad Pulmonar Obstructiva Crónica, pero no se centran en la gran cantidad de datos que genera el IoT en un despliegue a gran escala, ni en la integración de servicios de Smart City. Por estas razones, este trabajo presenta un sistema innovador basado en tecnologías de Cloud Computing y Big Data para integrar servicios de Smart City en un escenario a gran escala.

- *Arquitectura de funcionamiento basada en OneM2M para la interoperabilidad de dispositivos heterogéneos en IoT* [185]

  Ofrece una solución utilizando OneM2M como mecanismo de interoperabilidad de Plataformas Middleware. Por lo tanto, en esta investigación, para abordar este problema, se analizan las especificaciones oneM2M y se propone una arquitectura de funcionamiento basada en dichas especificaciones para respaldar tanto la interoperabilidad perfecta de dispositivos IoT heterogéneos como su integración con el ecosistema oneM2M. Se evalúa la viabilidad de esta arquitectura mediante un caso de uso aplicado a un escenario real, que deriva de INTER-IoT.

- *INTER-LAYER: un enfoque en capas para la interoperabilidad de la plataforma de IoT* [178]

  Ofrece una visión completa y detallada del diseño de todos los mecanismos de interoperabilidad descritos en el presente capítulo.

- *INTER-Framework: un marco de interoperabilidad para respaldar la interoperabilidad de la plataforma IoT* [168]

  Ofrece una visión completa y detallada del diseño de los componentes habilitantes y del marco de interoperabilidad de plataformas IoT descritos en el presente capítulo.

- *Interoperabilidad en plataformas IoT en la nube* (Capítulo de libro aceptado y pendiente de ser publicado)

  Cloud Computing representa un paradigma clave para apoyar el desarrollo de Plataformas IoT avanzadas, razón por la cual el número de Plataformas IoT orientadas a la nube aumenta continuamente, exponiendo arquitecturas, funcionalidades y objetivos muy diferentes. Este trabajo ofrece una perspectiva del mecanismo de interoperabilidad de Plataformas Middleware desde el punto de vista de un despliegue en la nube. Además, ofrece una categorización de la interoperabilidad Middleware desde el punto de vista de su tipo de despliegue.

Capítulo 7

ACTIVAGE: Entornos innovadores IoT para envejecer mejor

El proyecto europeo ACTIVAGE [81] se inició en 2017, bajo el programa H2020 de la Unión Europea. Es un proyecto piloto europeo a gran escala. El objetivo principal es construir el primer ecosistema europeo de IoT a través de 9 Entornos de Despliegue (DS) en siete países europeos. Esto se pone en práctica reutilizando y ampliando las plataformas, tecnologías y estándares de IoT abiertos y propietarios subyacentes, e integrando nuevas interfaces necesarias para proporcionar interoperabilidad a través de dichas plataformas heterogéneas, que permiten el despliegue y la operación a gran escala de soluciones y servicios de Envejecimiento Activo y Saludable [186] basados en IoT.

Figura 7.1: Logo proyecto ACTIVAGE.

Los dominios de aplicación considerados en el proyecto son sistemas que se utilizan para apoyar y ampliar la vida independiente de las personas mayores en sus entornos cotidianos, y respondiendo a las necesidades reales de los cuidadores, proveedores de servicios y autoridades públicas. El consorcio del proyecto está formado por industrias, centros de investigación, PYMES, proveedores de servicios y autoridades públicas que abarcan toda la cadena de valor en cada sitio de despliegue.

7.1. Visión general del proyecto

El proyecto ACTIVAGE utiliza las soluciones IoT para crear entornos que contribuyan a los sistemas de salud ampliando el tiempo de vida independiente de las personas mayores, actuando así en respuesta al cambio demográfico [187]. Desde un punto de vista técnico, el proyecto maneja las limitaciones y barreras que impiden lograr una aplicación sostenible y a gran escala de las tecnologías para un envejecimiento activo y saludable. El proyecto destaca por:

- Integrar una gran variedad de dispositivos.
- Recoger y analizar la información sobre el estilo de vida e identificar las necesidades.
- Proporcionar soluciones personalizadas, al tiempo que garantiza la privacidad y la seguridad de los datos.
- Contar con miles de participantes en nueve ciudades de siete países de la UE.
- Aprovechar los beneficios de las TIC para mejorar la calidad de vida.
- Garantizar la sostenibilidad de los sistemas sociales y sanitarios en Europa.

La arquitectura funcional genérica propuesta por ACTIVAGE sigue el modelo funcional de la Arquitectura de Referencia de Nivel Alto (HLA) de IoT, descrito por el AIOTI-WG3, que cumple con el Modelo de Referencia IoT de ITU-T Y.2060, la arquitectura de referencia OneM2M y la Arquitectura de Referencia Industrial-Internet (IIRA) del IIC, y considera las especificidades del dominio de Envejecimiento Activo y Saludable (AHA).

El núcleo del marco de interoperabilidad de ACTIVAGE es llamado ACTIVAGE IoT Ecosystem Suite (AIoTES) [188]. El proyecto proporciona un desacoplamiento total de las aplicaciones con las complejidades de la plataforma, la variabilidad en el diseño y los estándares utilizados. ACTIVAGE define

una arquitectura de referencia para la interoperabilidad de las plataformas IoT. Esta arquitectura tiene como objetivo construir enfoques generales para afrontar la interoperabilidad de forma universal, con el objetivo de servir de marco común para construir aplicaciones interoperables de envejecimiento que puedan ser desplegadas en diferentes entornos de despliegue dispersados por Europa.

Las plataformas IoT permiten la conexión de sensores y dispositivos, garantizando una integración perfecta con las redes de comunicación y proporcionando recursos y mecanismos para la gestión de estas cosas y datos conectados. La capa de plataforma en ACTIVAGE contiene las plataformas que forman parte del proyecto ACTIVAGE, a saber, FIWARE, SOFIA2, universAAL, OpenIoT e IoTivity. La motivación que da lugar a la agrupación de las plataformas en una capa es el objetivo de interconectar varias plataformas para conseguir un ecosistema interoperable de IoT a través de 9 lugares de despliegue. Esta capa sirve de puente entre las plataformas, funcionando como una capa de abstracción y permitiendo la conexión entre los dispositivos y la aplicación [189]. De este modo, un servicio podrá ser replicado en cualquier lugar de despliegue en toda Europa.

La solución propuesta por ACTIVAGE consiste en un conjunto de técnicas, herramientas y metodologías de software para la interoperabilidad, la protección de la privacidad y los datos y la seguridad entre plataformas heterogéneas de IoT y aplicaciones, servicios y soluciones de envejecimiento activo y saludable. Los principales componentes software de AIOTES proporcionada por ACTIVAGE, mostrados en la Figura 7.2, son los siguientes:

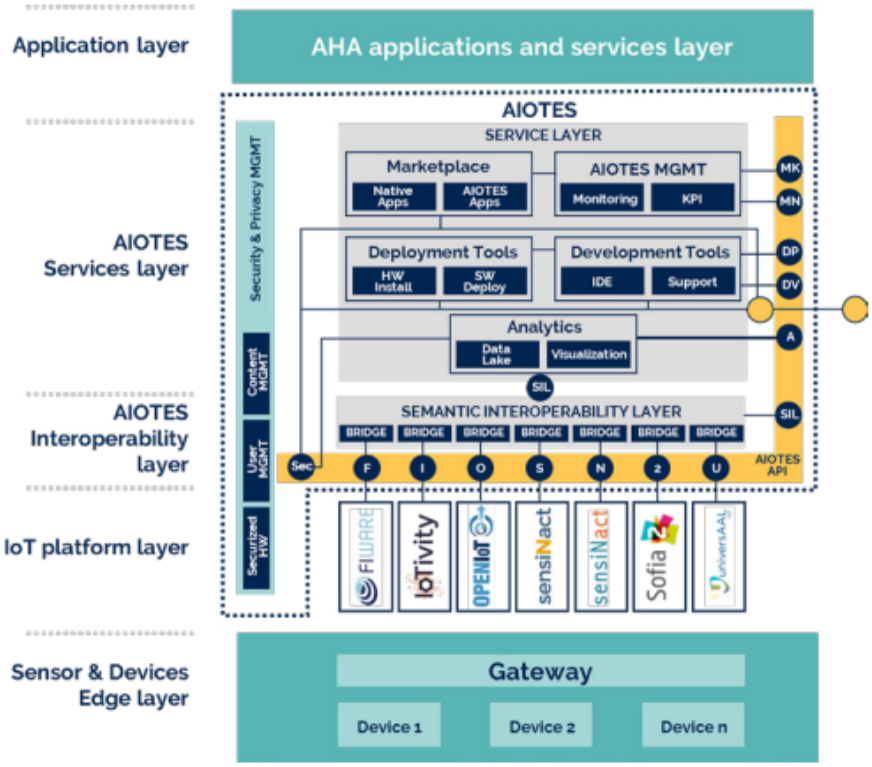


Figura 7.2: Visión general de la arquitectura de ACTIVAGE.

- **La capa de interoperabilidad semántica (SIL** es el elemento que proporciona interoperabilidad entre plataformas y permite que otros elementos de ACTIVAGE se comuniquen con cualquier plataforma a través de una API común. La capa de interoperabilidad se ha propuesto como solución para superar la falta de interoperabilidad entre las plataformas de IoT existentes.
- **La capa de servicios** ofrece un conjunto de herramientas que facilitan el uso y explotación de las funciones desarrolladas en el marco del proyecto. Está compuesta por:
  - Herramientas de desarrollo: El objetivo es ofrecer la infraestructura de desarrollo adecuada que facilite la creación de nuevas aplicacio-

nes IoT por parte de los desarrolladores técnicos, a través de la (re)utilización de las aplicaciones IoT existentes ya registradas en el ecosistema de aplicaciones de ACTIVAGE.

- Herramientas de despliegue: Permiten registrar nuevos elementos en el entorno, como dispositivos, plataformas, infraestructuras o aplicaciones
- Herramientas de análisis de datos: Proporcionan soluciones para Data Lake (repositorio de almacenamiento que contiene una gran cantidad de datos en bruto), análisis de datos o visualización.

- **La tienda digital (Marketplace)** que ofrece capacidades para desarrollar, proporcionar y obtener aplicaciones desarrolladas con AIOTES.
- **Las funcionalidades de seguridad y privacidad** de la información [190]. Los bloques funcionales que la componen son bloques encargados de:

  - Crear, actualizar, eliminar y gestionar las políticas de seguridad.
  - Tomar decisiones de autorización basadas en las políticas de seguridad.
  - Proporcionar puntos de aplicación de las políticas.
  - Ofrecer un Gestor de Identidades, que desempeña el papel de proveedor de identidad para usuarios y servicios.

7.2. Componentes de la arquitectura e implementación involucradas

Los componentes de la arquitectura e implementación involucradas se corresponden con los elementos resaltados en color rojo en la Figura 7.3, que se corresponden con los siguientes:

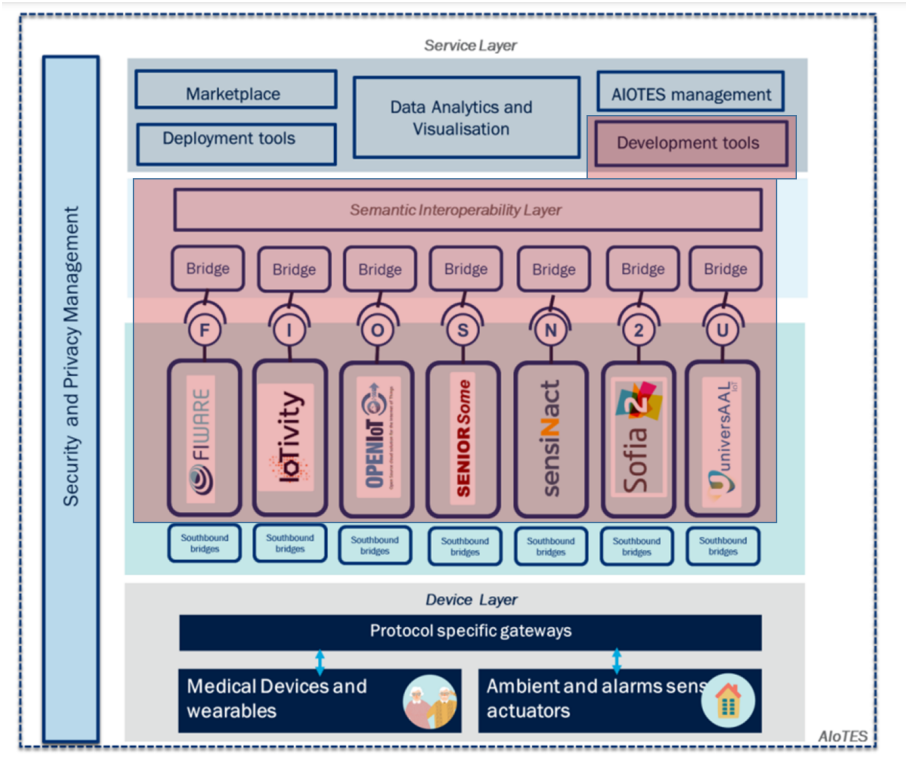


Figura 7.3: Localización de la solución propuesta dentro de la arquitectura ACTIVAGE.

- Los puentes. Se han especificado puentes para acceder a las funcionalidades de las diferentes plataformas IoT del proyecto.
- La solución de interoperabilidad en la capa SIL. Este componente corresponde con la solución de interoperabilidad middleware entre plataformas de la presente Tesis Doctotral.
- Las herramientas de desarrollo. Las mecanismos de interoperabilidad de aplicación y servicios se encuentran localizados dentro de este bloque funcional en ACTIVAGE.

Finalmente, elementos como las herramientas de despliegue y la gestión de AIOTES también corresponden con detalles de diseño y especificación del trabajo realizado en la presente Tesis Doctoral. No obstante, presentan alter-

nativas considerables y diferentes al diseño y la implementación expuesta en los Capítulos 3 y 4, que se explicarán detalladamente en la próxima sección.

7.3. Descripción de la solución implementada

La cronología de Casos de uso proporcionada en la Sección 5.1 señala a ACTIVAGE como el segundo proyecto utilizado para diseñar y especificar los mecanismos de interoperabilidad presentados en la presente Tesis Doctoral. En la sección 5.2 se puede observar que la solución implementada extiende, integra y valida los componentes desarrollados en la solución genérica ofrecida por la presente Tesis Doctoral e instanciados en el proyecto INTER-IoT dentro del proyecto ACTIVAGE. Concretamente, la Figura 5.3 muestra dicha jerarquia.

Por tanto teniendo en cuenta los componentes de ACTIVAGE relacionados con la presente Tesis Doctoral, descritos este capítulo, la solución implementada en el proyecto forma parte de las siguientes secciones de la presente Tesis Doctoral:

- AIOTES:
  - Interoperabilidad Semantica: Este componente corresponde con la arquitectura presentada en la sección 3.4.2 y la implementación ofrecida en la sección 4.2.2. Sigue el mismo diseño e implementación que el producto software INTERMW de INTER-IoT. Permite la interoperabilidad a nivel de middleware de todas las plataformas IoT involucradas en el proyecto ACTIVAGE.
  - Capa de Servicios:
    - Herramientas de desarrollo:
      - Composición de servicios: Este producto corresponde con la arquitectura presentada en la sección 3.4.4 y la implementación ofrecida en la sección 4.2.4. Sigue el mismo diseño e implementación que el producto software de interoperabilidad de aplicaciones y servicios (AS2AS) de INTER-IoT. Permite la interoperabilidad a nivel de middleware de todas los servicios y aplicaciones IoT involucrados en el proyecto ACTIVAGE. En este caso, desde el punto de vista de la arquitectura, este mecanismo de interoperabilidad ha pasado de ser un componente central del proyecto, como ocurría en INTER-IoT, a ser una utilidad dentro la capa de aplicaciones de AIOTES.

- Interoperabilidad de Servicios: Este componente corresponde con la arquitectura presentada en la sección 3.4.3 y la implementación ofrecida en la sección 4.2.3. Corresponde con la instanciación de dicho componente basado en Interoperabilidad de Plataformas Middleware (Figura 4.11) y la instanciación de Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos (Figura 4.12). Se presenta como una funcionalidad añadida de AIOTES para acceder de una manera común a los datos de las diferentes plataformas y servicios.

Finalmente, como se indicaba en la sección anterior, se utilizaron otras soluciones software para implementar los componentes transversales y el marco de interoperabilidad. Las herramienta elegidas en el proyecto ACTIVAGE fueron las siguientes:

- Herramienta de gestión y soporte: Se seleccionó la solución Portainer para ser la encargada de gestionar los componentes desarrollados. Es una solución de código abierto que se encargaba de desplegar y gestionar los contenedores Docker disponibles con las imagenes de los componentes.
- La herramienta de marco de interoperabilidad fue desarrollada por un grupo del proyecto y no entra dentro del alcance de la presente Tesis Doctoral. No obstante, ofreciendo unos breves detalles, fue diseñada mediante una solución software desarrollada en Angular y Node.js. En este caso, estaba centrada exclusivamente en la gestión de los dispositivos conectados a las plataformas.
- Desde el punto de vista de la privacidad y la seguridad:
  - El gestor de identidades seleccionado fue Keycloack.
  - Se utilizó un servidor web, desplegado dentro de un contenedor, para actuar de auto-proxy y con compatibilidad integrada con Let's Encrypt para generar automáticamente certificados SSL.
  - Se desplegó un API Gateway llamado Node Express Gateway para unificar las APIs e interfaces de los componentes desarrollados.

7.4. Validación y resultados

Como se ha indicado durante este capítulo, los mecanismos de interoperabilidad propuestos en esta sección derivan de los mecanismos de interoperabilidad presentados en el Capítulo 6, INTER-IoT. Por tanto, durante ACTIVAGE

se ha puesto especial énfasis en su integración en un piloto a gran escala en diferentes ubicaciones de despliegue en Europa. Se presentan las principales conclusiones respecto a los mecanismos de interoperabilidad especificados en el presente trabajo.

Relacionados con la interoperabilidad en la capa Middleware

La integración de INTER-MW en el ecosistema ACTIVAGE, concretamente dentro de su arquitectura, se ha realizado conectándolo con el resto de componentes de la arquitectura, tal y como se ha detallado a lo largo de este capítulo.

El enlace inferior, el que conecta INTER-MW con las Plataformas IoT ACTIVAGE, se implementa a través de puentes de Plataforma IoT que son los encargados de establecer la comunicación entre INTER-MW y las Plataformas IoT. Durante el Proyecto INTER-IoT se desarrollaron varios puentes para la Plataforma IoT, siendo posible reutilizarlos en el ecosistema ACTIVAGE. No obstante, el proyecto incluye la utilización de plataformas que no han sido utilizadas en INTER-IoT. En ese caso ha sido necesario implementar nuevos puentes de plataforma en el contexto del Proyecto ACTIVAGE, logrando así la total interoperabilidad entre todas las plataformas del ecosistema.

Se han desarrollado puentes e integrado las siguientes plataformas con la solución AIOTES: FIWARE, SOFIA2, universAAL, sensiNact, OpenIoT e IoTivity. En la literatura desarrollada respecto a esta Tesis Doctoral y en los correspondientes entregables del proyecto se puede encontrar información detallada de las plataformas utilizadas en cada lugar de despliegue y las pruebas que se realizaron. Finalmente, se conectaron las plataformas mediante puentes en 12 entornos de despliegue. Cada entorno de despliegue estaba compuesto por una o más plataformas conectadas a AIOTES. Más de 2000 usuarios se conectaron a AIOTES gracias a los puentes.

Otros elementos de AIoTES, a saber, la Gestión y la Capa de Servicio, se comunican con INTER-MW a través de su API REST. INTER-MW actuao como el componente central de la arquitectura proporcionando una funcionalidad básica de interoperabilidad que es explotada en gran medida por el resto de los componentes. Se presentan, a continuación, una serie de casos de uso de interoperabilidad de ACTIVAGE que muestran la interoperabilidad lograda entre plataformas y aplicaciones.

Aplicación multiplataforma

Se busca solucionar el problema de aquellas aplicaciones de un sitio de despliegue que son específicas de una plataforma y no es posible utilizarlas en sitios de despliegue donde dicha plataforma IoT no está desplegada. La meta es

permitir la creación de nuevas aplicaciones que se puedan utilizar en cualquier sitio de despliegue.

Consiste en aplicaciones que se construyen sobre INTER-MW para funcionar en múltiples ubicaciones de despliegue y en sus respectivas plataformas. En este escenario, INTER-MW funciona como una capa de abstracción situada entre las plataformas. Las aplicaciones creadas pueden ser utilizadas independientemente de la plataforma ya desplegada en el entorno.

Hay dos aproximaciones a este tipo de soluciones. La primera aproximación, mostrada en la Figura 7.4, demuestra que las nuevas aplicaciones multiplataforma pueden utilizarse de forma autónoma con cualquier plataforma IoT integrada en el ecosistema. Las nuevas aplicaciones multiplataforma construidas sobre INTER-MW pueden gestionar diferentes despliegues de plataformas al mismo tiempo. La misma aplicación funciona sobre las diferentes plataformas conectadas a AIOTES.

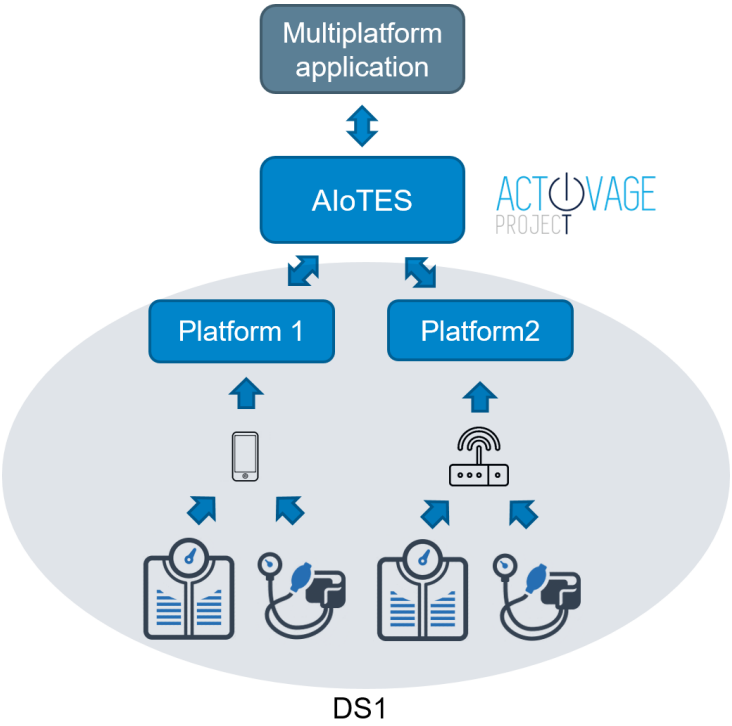


Figura 7.4: Escenario de aplicación multiplataforma en un entorno de despliegue.

En la Figura 7.5 se presenta el escenario de una aplicación que es compatible con todas las plataformas de ACTIVAGE.

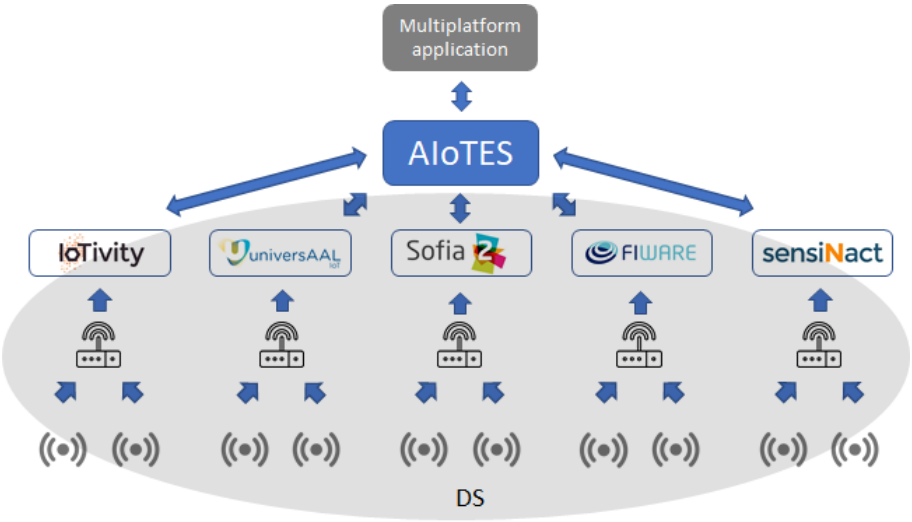


Figura 7.5: Escenario de aplicación multiplataforma conectada a distintas plataformas IoT.

En la segunda aproximación, mostrada por la Figura 7.6, las aplicaciones multiplataforma construidas sobre AIOTES pueden funcionar en distintos entornos de despliegue al mismo tiempo y con cualquier plataforma IoT. La misma aplicación funciona sobre las plataformas del entorno de despliegue 1 y el entorno 2.

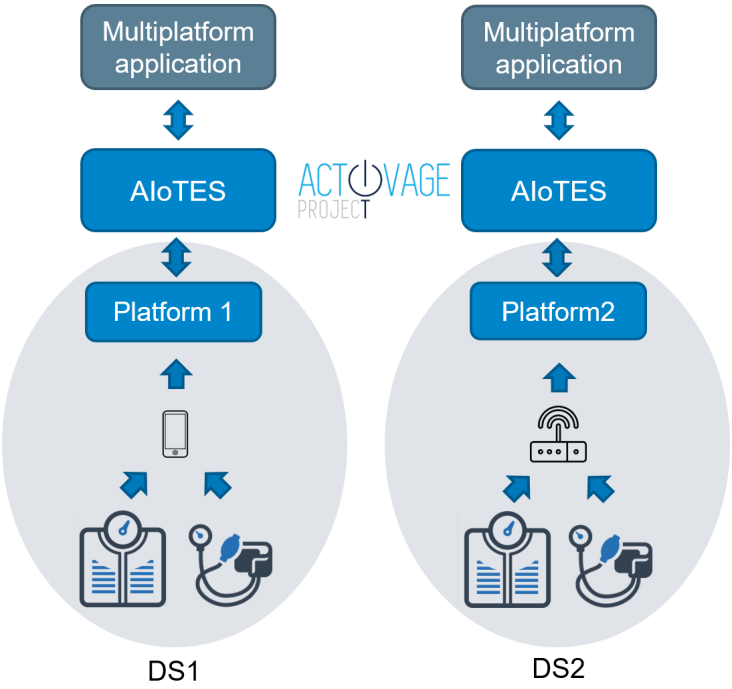


Figura 7.6: Escenario de aplicación multiplataforma en distintos entornos de despliegue.

En la Figura 7.7 se muestra la captura de pantalla de una aplicación que ofrece un mapa de la ubicación de varios puntos geográficos en los que se realizó una demostración práctica.

Figura 7.7: Mapa de ubicación de entornos de desarrollo en demostración practica.

La aplicación multiplataforma puede funcionar en diferentes ubicaciones, la primera haciendo uso de las plataformas SOFIA 2 y FIWARE (como se muestra en la Figura 7.8) y la segunda haciendo uso de FIWARE y universAAL (como se muestra en la Figura 7.8).

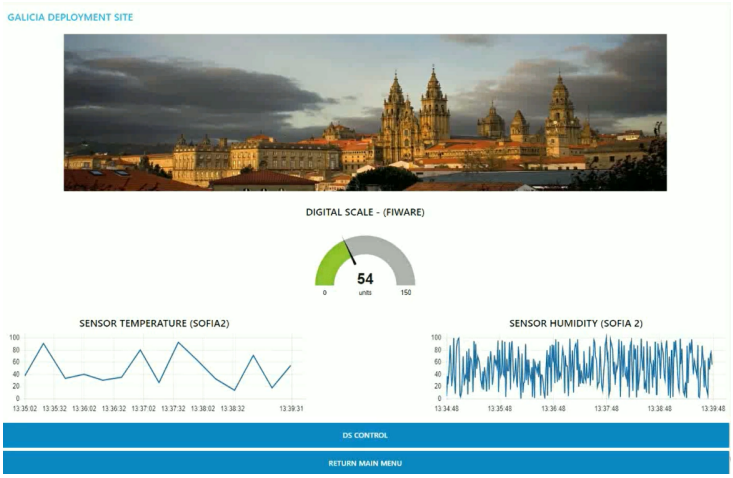


Figura 7.8: Aplicación multiplataforma en entorno de despliegue (I).

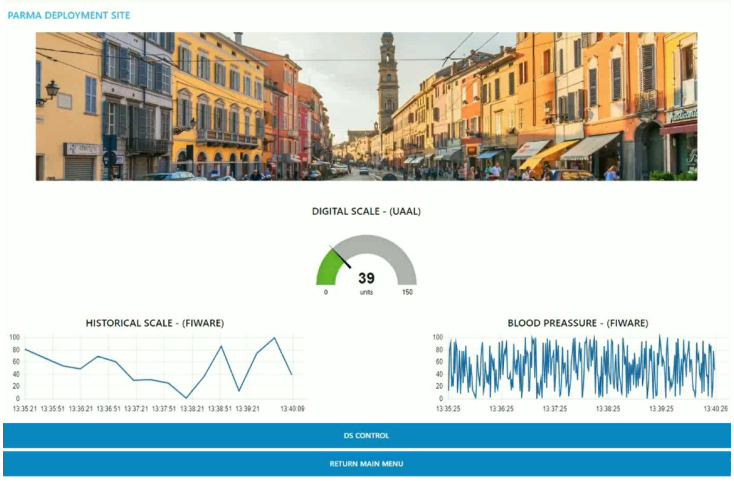


Figura 7.9: Aplicación multiplataforma en entorno de despliegue (II).

Aplicación específica de una plataforma

El siguiente ejemplo de validación permite que las aplicaciones específicas de una plataforma puedan ser utilizadas junto con una plataforma externa en un sitio de despliegue.

Se presenta un caso de uso en el que un determinado entorno de despliegue sólo implementa soluciones correspondientes a una determinada plataforma IoT. El objetivo es que dicha aplicación específica de la plataforma pueda ser utilizada junto con otra plataforma IoT. Dado que la aplicación ha sido diseñada específicamente para una plataforma concreta, en principio, no es posible utilizar aplicaciones externas de las plataformas IoT en los entornos de despliegue en los que no están desplegadas las mismas plataformas.

Esta situación se soluciona con el uso de INTER-MW. Este se encarga de la comunicación entre la aplicación y la plataforma externa. Por lo tanto, dentro de un único entorno de despliegue gestionado por una plataforma IoT específica, es posible utilizar aplicaciones de plataformas externas sin ningún cambio en la configuración del entorno de despliegue (Figura 7.10). Además, esta arquitectura permite que los entornos de despliegue utilicen dispositivos de terceros gestionados por una plataforma diferente. Además, se mantienen sin alterar las aplicaciones nativas que se ejecutan sobre la plataforma inicial.

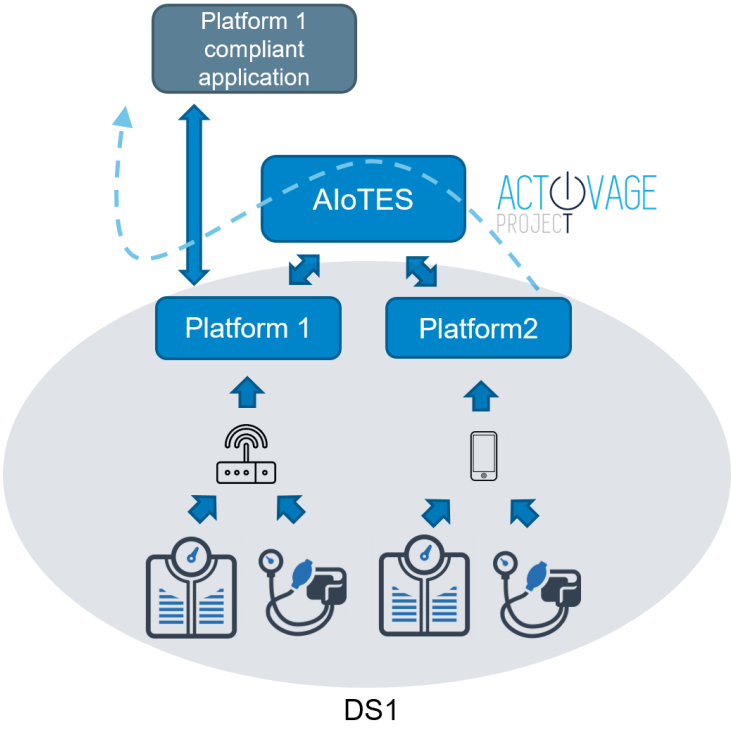


Figura 7.10: Uso de una aplicación específica de una plataforma mediante otra plataforma IoT.

Base de interoperabilidad de nuevos servicios y aplicaciones

Dado que INTER-MW proporciona acceso a los datos de las plataformas de forma unificada, es posible aprovechar todo su potencial utilizándolo como componente clave para alimentar nuevos servicios más complejos (Figura 7.11).

La capa de servicios está compuesta por elementos como un Data Lake, que proporciona un punto de entrada para acceder a los datos disponibles en el ecosistema. Más concretamente, el Data Lake de ACTIVAGE recupera primero los datos de los entornos de despliegue en el formato adecuado, luego transforma los datos al formato de ACTIVAGE y, finalmente, los almacena.

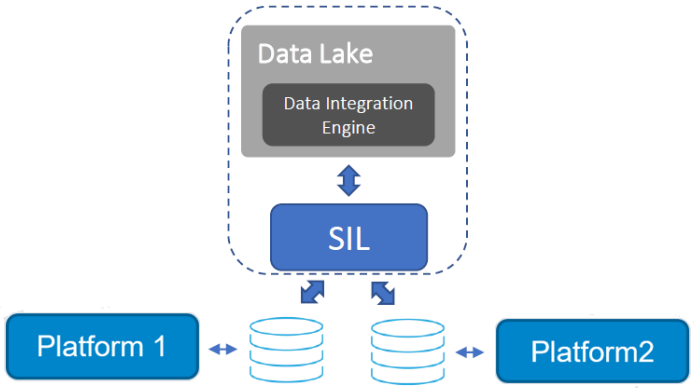


Figura 7.11: Mecanismo de interoperabilidad como base de servicio Data Lake.

Es importante señalar que el objetivo del Data Lake es alimentar a otro elemento software, el analizador de datos. Éste proporciona una interfaz a través de la cual es posible experimentar con los métodos analíticos disponibles, utilizando los datos unificados procedentes de las diferentes plataformas. De este modo, los entornos de despliegue no se ven obligados a utilizar las herramientas de análisis proporcionadas por sus plataformas, que no contemplan la posibilidad de analizar datos externos. Mediante la capa de interoperabilidad, que alimenta el Data Lake y el Analizador de Datos, los entornos de despliegue pueden beneficiarse de un análisis completo para sus sistemas. De este modo, se facilita a los entornos de despliegue la comprobación y el análisis de todos los datos en la misma aplicación.

Beneficios clave de los casos de uso

La gran ventaja de utilizar INTER-MW radica en que, al abstraer los detalles técnicos de implementación, los proveedores de servicios y aplicaciones que quieran utilizar un software específico no necesitan adaptar dicho software a las APIs específicas de la plataforma y a sus modelos de información. De este modo, se pueden desarrollar aplicaciones para múltiples entornos de despliegue. Dado que INTER-MW resuelve la comunicación entre las plataformas de forma transparente para sus aplicaciones, las aplicaciones compatibles con INTER-MW son agnósticas a la plataforma, desarrollándose genéricamente sin tener en cuenta su plataforma de destino. El desarrollo de estas aplicaciones es mucho más eficiente que el convencional, ya que pueden ser desarrolladas y desplegadas en cualquier lugar.

En el caso de la integración de una plataforma IoT compatible con INTER-MW en un nuevo entorno de despliegue, el hecho de haber adaptado la plataforma sólo a INTER-MW en lugar de a plataformas IoT individuales, no sólo facilita la adopción de soluciones específicas de la plataforma, sino que también mejora las oportunidades de explotación a las diferentes entidades desarrolladoras. De este modo, es posible la creación de un ecosistema en el que las aplicaciones que se han desarrollado para plataformas específicas puedan importarse a otras, fomentando el intercambio de conocimientos y evitando la duplicación.

Además, múltiples plataformas IoT pueden coexistir en el mismo entorno de despliegue y soportar la misma aplicación con sus propias características. Esto abre la posibilidad de que la parte interesada incluya nuevas plataformas, facilitando su funcionamiento en su plataforma nativa sin ninguna migración.

Relacionados con la interoperabilidad en la capa de Aplicación y Servicios

La solución de interoperabilidad de aplicación y servicios basada en flujos está ubicada dentro de las herramientas de desarrollo, tal y como se ha indicado en las secciones anteriores de este capítulo. Se ha utilizado esta herramienta para crear una solución de interoperabilidad de aplicación y servicios siguiendo los principios establecidos en los Capítulos 3 y 4.

Uno de los demostradores ha consistido en la creación de un flujo (Figura 7.12) para realizar el proceso de recuperación de la información histórica de una plataforma IoT. Esta flujo de ejecución devuelve la información en el modelo de datos de la plataforma IoT. Finalmente, un servicio se encarga de hacer la conversión al formato ACTIVAGE. El propósito es enviar esta información en formato ACTIVAGE a un nuevo servicio conectado al flujo de interoperabilidad.

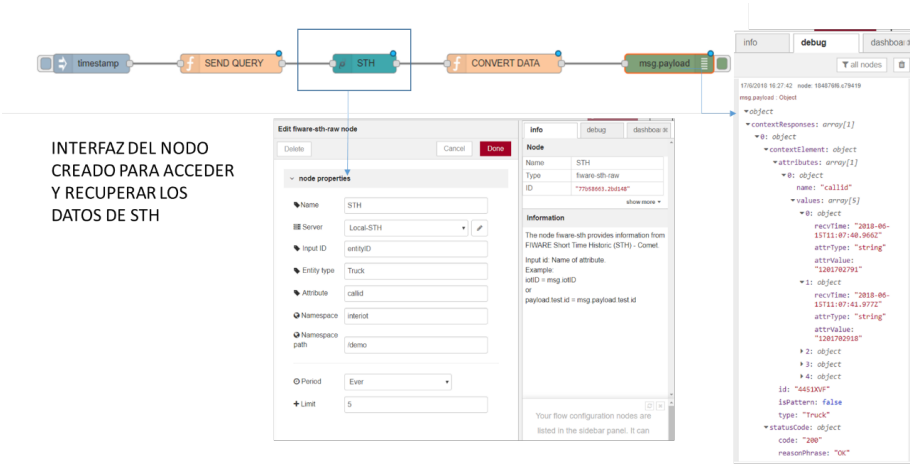


Figura 7.12: Flujo de transformación de servicios históricos de plataformas a un formato común.

En este caso, los nodos de consulta a datos históricos son proporcionados por las plataformas del sitio de despliegue, facilitan el acceso para la información histórica de la plataforma y los datos en el formato de la plataforma. Por ejemplo, en la plataforma FIWARE esto se realiza mediante el servicio STH y un nodo que se encarga de integrar su funcionamiento para que pueda ser llamado por el mecanismo de interoperabilidad.

El nodo de traducción es un nodo especifico desarrollado para traducir los datos de los modelos de datos de una plataforma IoT al modelo de datos AIOTES. Para su funcionamiento, se necesita un puente a la plataforma IoT y una instancia en funcionamiento de INTER-MW. El nodo se conecta a la API de esa instancia en funcionamiento de la INTER-MW. A continuación, el nodo convierte los datos al formato común de ACTIVAGE. La Figura 7.13 explica el flujo de todo este proceso de interoperabilidad.

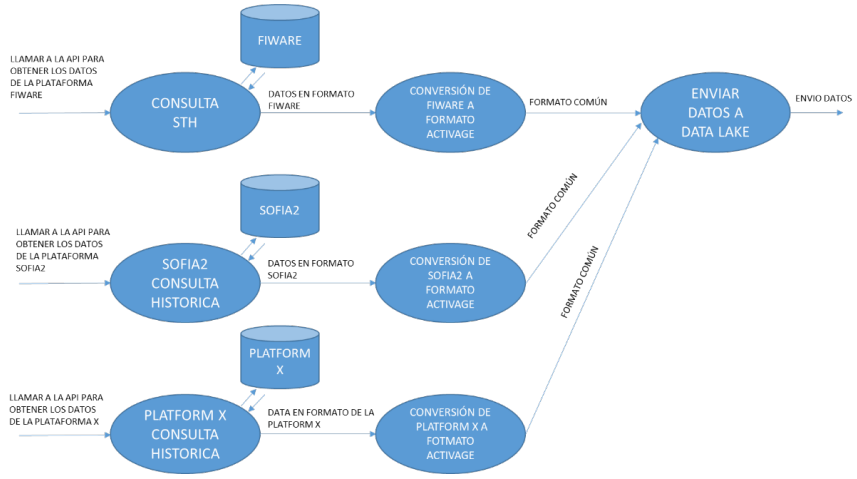


Figura 7.13: Diseño del flujo de transformación de servicios históricos de plataformas a formato común.

La Figura 7.14 presenta la perspectiva de Node-RED, que muestra como parte de un nodo que obtiene los datos de un servicio histórico ubicado en las plataformas IoT y enlaza con el nodo traductor.

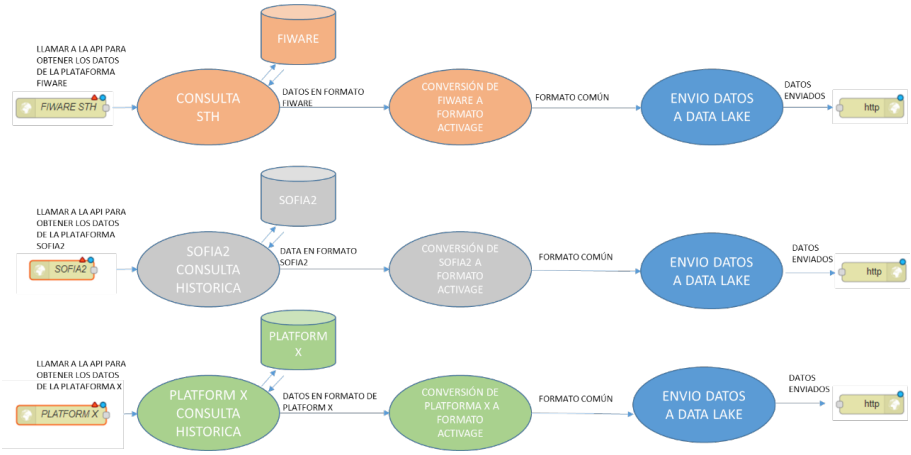


Figura 7.14: Vista de los nodos del flujo de transformación servicios históricos de plataformas a formato común.

Es importante destacar que existe la posibilidad de utilizar una API creada con Node-RED para recibir la información, una vez configurados todos los nodos y flujos (Figura 7.15).

Figura 7.15: Vista API del flujo de transformación servicios históricos de plataformas a formato común.

La Figura 7.16 muestra el resultado de la llamada, que puede ser utilizada tanto por Node-RED como por cualquier servicio externo al mecanismo de interoperabilidad.

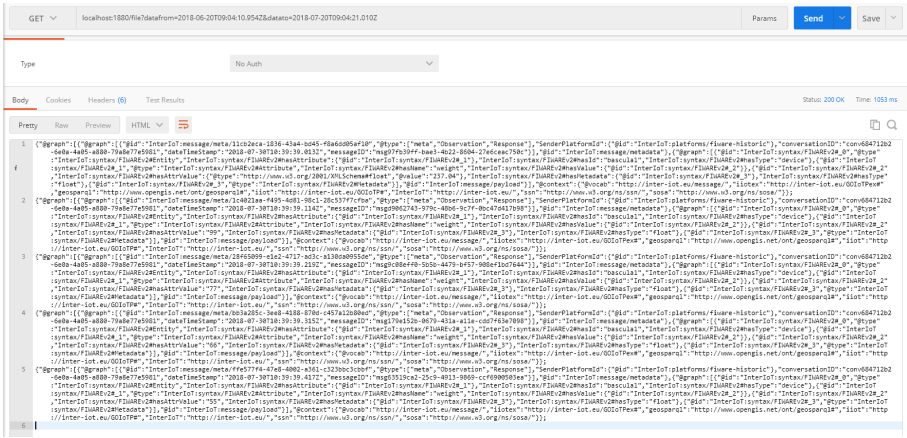


Figura 7.16: Resultado de la llamada a la API del flujo de transformación de servicios históricos de plataformas a formato común.

7.5. Alcance de la solución dentro del proyecto

La solución y arquitecturas propuesta en esta Tesis Doctoral tienen el siguiente alcance y relación con ACTIVAGE:

- Interoperabilidad de Plataformas Middleware:
  - Despliegue y configuración Plataformas IoT Middleware.
  - Validación de la solución en casos de uso.
- Interoperabilidad de Aplicaciones y Servicios:
  - Despliegue y configuración de servicios y aplicaciones de plataformas IoT.
  - Diseño e implementación de la solución y validación en casos de uso.
- Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos:
  - Despliegue y configuración de servicios y aplicaciones de plataformas IoT.
  - Validación de la solución en casos de uso.

El proyecto ha dado lugar a diferentes publicaciones por parte del presente autor, que se resumen a continuación:

- Las publicaciones: *Solución de interoperabilidad mediante programación basada en flujos para aplicaciones de plataformas IoT* [127] y *Interoperabilidad en plataformas IoT en la nube* (Capítulo de libro aceptado y pendiente de ser publicado). Ambas descritas en el capítulo anterior relativo al caso de uso INTER-IoT.

  Estas publicaciones describen el diseño e implementación de los mecanismos de interoperabilidad middleware y de aplicaciones y servicios utilizados en ACTIVAGE. Estas publicaciones van acompañadas de escenarios y demostraciones utilizadas en ACTIVAGE.

- *AIoTES: Establecimiento de los principios para una arquitectura de referencia moderna e interoperable semántica habilitada para IoT para ecosistemas de envejecimiento activo y saludable* [188]

  Este artículo presenta ACTIVAGE IoT Ecosystem Suite (AIoTES), desde el punto de vista de diseñar una arquitectura de referencia y su proceso de implementación. Describe como esta herramienta aborda estos problemas y el proceso de diseño dentro del piloto europeo a gran escala llamado ACTIVAGE. Además, el articulo demuestra como AIoTES ha sido validado con éxito probando todos sus componentes individualmente e integrados en entornos del mundo real con 4.345 usuarios directos. Cada validación está contextualizada en 11 sitios de despliegue (DS) con 13 escenarios de validación que cubren la heterogeneidad de las necesidades de AHA-IoT. Estos resultados también muestran un camino claro para futuras mejoras. Finalmente, destaca la importancia de los esfuerzos de estandarización en el dominio de AHA-IoT, un ecosistema en constante evolución.

Capítulo 8

DataPorts: Plataforma de datos para los puertos cognitivos del futuro

El proyecto europeo DataPorts [82] (Figura 8.1) se inició en 2020, bajo el programa Horizonte 2020 (H2020) de la Unión Europea, y estará en desarrollo hasta final del año 2022. El objetivo del proyecto es proporcionar una plataforma de datos en la que las empresas de transporte y logística de un puerto marítimo puedan gestionar los datos, como cualquier otro activo de la empresa, con el fin de crear la base para ofrecer servicios cognitivos.

Figura 8.1: Logo proyecto DataPorts.

El dominio de aplicación considerado en el proyecto es el entorno portuario. Los actores que operan en las diversas cadenas de suministro están involucrados en el dominio de los puertos marítimos. Por tanto, esto permite la conexión con otras partes interesadas en la cadena de suministro logístico. La adopción y uso de esta plataforma de datos por los puertos digitales existentes implica

su transición a puertos cognitivos reales, aprovechando la enorme cantidad de datos producidos por las partes interesadas y abriendo el camino a nuevas capacidades [191][192]. El proyecto está dedicado a la creación de una plataforma de datos segura que permita compartir la información no solo entre agentes portuarios sino también entre otros puertos. Por lo tanto, este es un entorno seguro de intercambio de datos de manera confiable y segura, con permisos de acceso y contratos para permitir el intercambio de datos y la exploración de nuevos servicios cognitivos e inteligencia artificial.

Antes de pasar a describir con detalle el proyecto DataPorts, se introduce la información relevante de un proyecto cronológicamente anterior, llamado PIXEL. Ambos proyectos comparten un gran numero de puntos en común, tanto en el dominio de aplicación como en las desiciones de desarrollo e implementación. Por tanto, es necesario observar las decisiones derivadas de PIXEL, para continuar con la descripción de la solución diseñada e implementada en DataPorts.

8.1. Proyecto precedente: PIXEL

Durante el proyecto PIXEL se sentaron las bases de diseño e implementación que fueron seguidas por los componentes centrados en la interoperabilidad del proyecto DataPorts. La solución obtenida en DataPorts es una evolución del trabajo que se realizó en el proyecto PIXEL. Es por ello que se ha decidido presentar el proyecto PIXEL, su diseño, implementación y los resultados obtenidos dentro de este capítulo sobre el proyecto DataPorts.

El proyecto europeo PIXEL [193] se inició en mayo de 2018, bajo el programa Horizonte 2020 (H2020) de la Unión Europea. Ofrece una solución inteligente, flexible y escalable para reducir los impactos ambientales al tiempo que permite la optimización de las operaciones en los ecosistemas portuarios a través de IoT.

Figura 8.2: Logo proyecto PIXEL.

Los dominios de aplicación considerados en el proyecto son puertos medianos o pequeños, donde el presupuesto es limitado y los servicios de las TIC están subcontratados. El proyecto permite una colaboración bidireccional de puertos, agentes de transporte multimodal y ciudades para un uso óptimo de los recursos internos y externos, el crecimiento económico sostenible y la mitigación del impacto ambiental, hacia los Puertos del Futuro. Construido sobre las tecnologías de interoperabilidad, PIXEL centraliza los datos de los diferentes silos de información donde las partes interesadas internas y externas almacenan su información operativa. PIXEL aprovecha una infraestructura de comunicación basada en IoT para intercambiar datos de forma voluntaria entre puertos y partes interesadas para lograr un uso eficiente de los recursos en los puertos.

8.1.1. Descripción de la plataforma PIXEL

La plataforma PIXEL tiene como objetivo cubrir varios desafíos y facilitar las operaciones en los puertos. Es necesario recopilar e intercambiar datos heterogéneos, incluidos los sensores de IoT, entre las distintas partes interesadas que operan en los puertos. Normalmente, estos datos son difíciles de manejar debido a la falta de homogeneización y de herramientas para operarlos. Mediante el uso de modelos de datos FIWARE que han sido extendidos, PIXEL aporta claridad, homogeneidad y semántica común a estos datos. Además, estos datos también se tratan y explotan mediante modelos específicos y algoritmos predictivos (por ejemplo, energía, contaminación, tráfico, etc.) que ayudan a definir cada Indicador clave de rendimiento (KPI) medioambiental y operativo en los puertos. Por último, el acceso y tratamiento de los mismos se presenta a través de un panel de control de fácil manejo para los operadores portuarios (y, eventualmente, para otras partes interesadas del puerto) [194]. La arquitectura PIXEL se divide en varios bloques de construcción para cubrir una amplia gama de necesidades de los puertos pequeños, medianos y grandes. El enfoque está centrado en los datos, de modo que múltiples partes interesadas y aplicaciones pueden acceder de forma homogénea a los datos y proporcionar servicios de valor añadido sobre ellos.

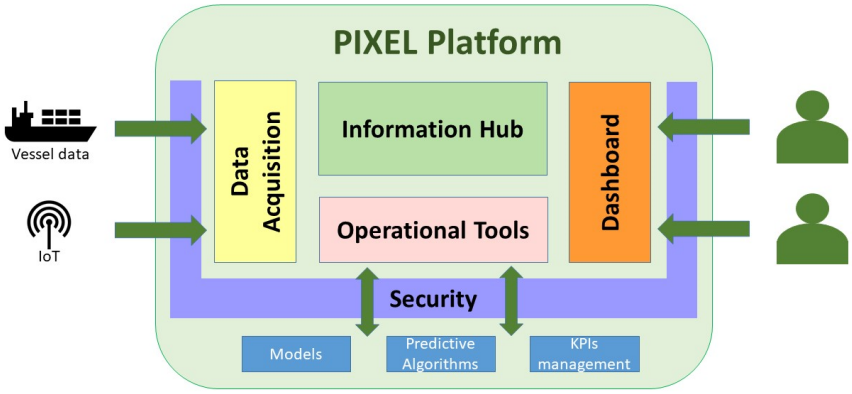


Figura 8.3: Arquitectura de componentes del proyecto PIXEL.

La Figura 8.3 muestra los principales bloques que componen la arquitectura PIXEL:

- **Modelos de datos:** Se basan en los modelos de datos de FIWARE, que han sido armonizados para permitir la portabilidad de datos para diferentes aplicaciones, incluyendo Smart Cities, Smart Agrifood, Smart Environment, Smart Sensoring, Smart Energy, Smart Water y otros dominios. El dominio de aplicación PIXEL se refiere a los puertos y a las operaciones portuarias, pero se sigue y se amplía la metodología del modelo de datos FIWARE.
- **Modelos:** Se ofrecen herramientas que permiten una adecuada modelización, simulación y análisis de datos. Basándose en la realidad de las actividades portuarias y en los datos disponibles, la plataforma PIXEL ofrece varios modelos como herramientas útiles [195]. Esto ayuda a los puertos a tener un mejor conocimiento de sus impactos ambientales. Estos modelos se centran en:
  - Modelización de la cadena de suministro y de las actividades del puerto.
  - Modelización del consumo y la producción de energía del puerto.
  - Modelización de la contaminación ambiental del puerto.

- **Algoritmos predictivos:** Los algoritmos de predicción de PIXEL son capaces de realizar las siguientes predicciones:
  - Predicción de los datos de las escalas de los buques a partir de formularios y otras fuentes.
  - Análisis y predicción de las condiciones de tráfico por carretera en relación con las operaciones portuarias.
  - Predicción de la producción de energía renovable.
- **Índice medioambiental portuario:** El Índice Medioambiental Portuario (PEI) es un indicador cuantitativo compuesto del rendimiento medioambiental global de un puerto. La idea central es concebir una metodología completa, estandarizada y transparente que sirva para integrar todos los aspectos medioambientales significativos de un puerto y los impactos correspondientes en una única métrica [196]. La métrica es utilizada por los puertos para:
  - Evaluar su propio rendimiento medioambiental de forma global e integradora.
  - Comparar sus resultados medioambientales con los de otros puertos que hayan utilizado la métrica.
- **Capa de Adquisición de Datos (Data Acquisition):** Consiste en varios componentes diseñados para enviar los datos desde las diversas fuentes de datos disponibles a un concentrador de información. La solución proporciona una forma estándar de adquirir datos de diferentes fuentes de datos que implementan diferentes tipos de protocolos y diferentes tipos de datos. La idea es proporcionar una forma estándar de importar datos al concentrador de información para permitir un fácil uso de cualquier tipo de fuentes de datos disponibles en los puertos.
- **Concentrador de información (Information Hub):** Consta de varias partes divididas conceptualmente en componentes que empujan los datos hacia la base de datos; componentes que intervienen en la recuperación de los datos almacenados y su posterior procesamiento y componentes responsables de la persistencia y el almacenamiento de los datos. Además, el sistema proporciona servicios de apoyo para configurar, gestionar y supervisar el concentrador de información.
- **Herramientas Operativas (Operational Tools):** Se encargan principalmente de acercar al usuario tanto los modelos como los algoritmos predictivos desarrollados en PIXEL. Por usuario se entiende aquí a los

administradores y gestores que analizan las operaciones portuarias mediante modelos de simulación y algoritmos predictivos. Para alcanzar este objetivo, se definen un conjunto de herramientas operativas de alto nivel, como la publicación, edición, ejecución y programación de modelos y algoritmos predictivos.

- **Cuadro de mandos y notificaciones (Dashboard):** Está diseñado para mostrar los datos disponibles de las Herramientas Operativas y también del Centro de Información. Estos datos son el resultado de:

  - Recuperar datos del IoT y de otros sensores de información.
  - Aplicar algoritmos y modelos predictivos.
  - Calcular el PEI.
  - Notificaciones del procesamiento de eventos. Cada vez que se cumpla una regla, se recibirá una notificación.

  Los datos se presentan al usuario de la plataforma a través de tres canales principales: gráficos y panel de control, vista de mapa de los datos geolocalizados como sensores, y notificaciones.

- **Solución de seguridad:** Se encarga de proporcionar una solución para identificar y autenticar a los usuarios, que podría conectarse a las soluciones de gestión de identidades existentes ya desplegadas en los puertos, y también de proporcionar una solución para controlar el acceso de los datos gestionados por la plataforma.

8.1.2. Componentes de la plataforma PIXEL relacionados con la solución propuesta

La solución propuesta en la Tesis Doctoral se incluye en la capa de adquisición de datos (DAL). La Figura 8.4 muestra su ubicación en los diferentes componentes de la arquitectura de PIXEL. Su objetivo es conectar las fuentes de datos externas al centro de información y al cuadro de mando. Este mecanismo de interoperabilidad se basa en convertir el formato de datos original y heterogéneo a un modelo de datos común. Por tanto, la solución tiene el rol de ser la capa habilitadora para el funcionamiento del resto de componentes de la plataforma. Ofrece el mecanismo principal de acceso común a los datos de fuentes heterogéneas.

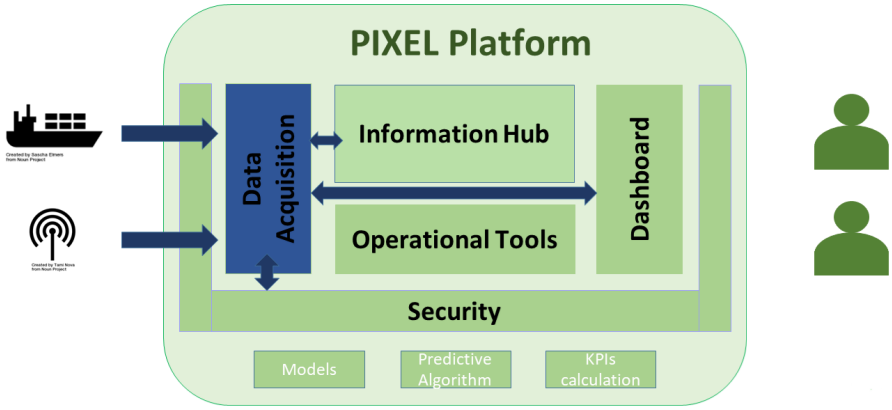


Figura 8.4: Componentes de la arquitectura e implementación involucradas en PIXEL.

Se trata de un componente clásico del IoT con la clara misión de reunir en un elemento central todas las entidades que proporcionan datos al marco de desarrollo. Un concepto crucial en este módulo es la consistencia del formato de los datos. El objetivo es proporcionar unas estructuras sintácticas y semánticas uniformes para todas las fuentes de datos entrantes. De ahí que haya que incorporar los agentes de interoperabilidad para hacer que los datos en bruto cumplan con el esquema esperado. Los formatos de datos han sido diseñados para seguir los modelos de datos Fiware NGSI de Fiware Smart Data Models. La tecnología seleccionada para el módulo de interoperabilidad es Fiware Orion Context Broker. La selección de ORION aporta la interoperabilidad IoT a todo el marco, garantizando el funcionamiento en tiempo real del sistema gracias al enfoque de publicación-suscripción.

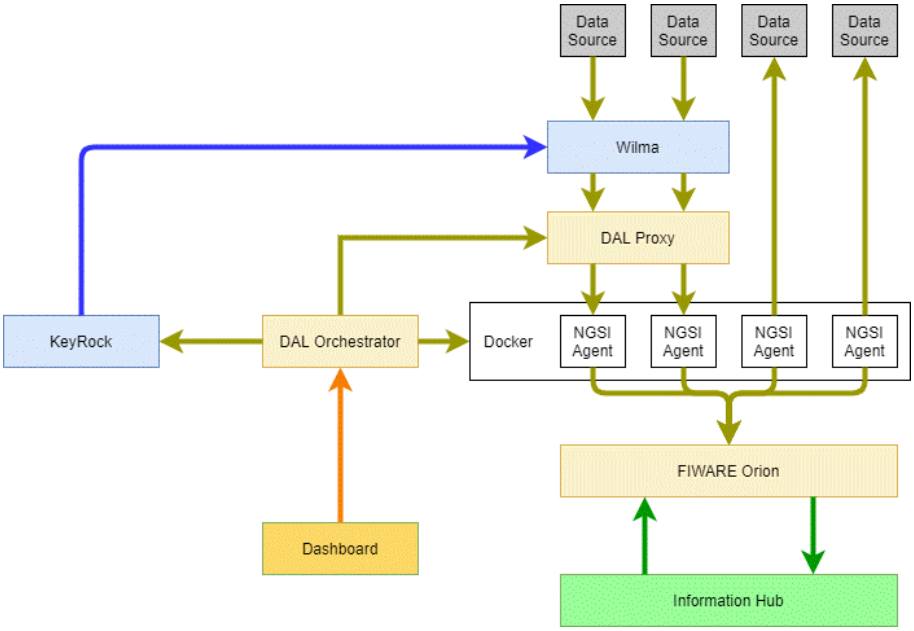


Figura 8.5: Descripción detallada del componente de acceso a datos de PIXEL.

La solución propuesta, representada en la Figura 8.5, se ha implementado y validado concretamente en las siguientes partes:

- Agentes NGSI: Para este caso particular se ha utilizado directamente el marco de desarrollo llamado pyngsi, implementado con el lenguaje Python, proporcionado por un socio del proyecto como herramienta de código libre para facilitar el desarrollo de los agentes y su interconexión con Orion Context Broker. Escribir un agente NGSI utilizando este marco de desarrollo evita desarrollarlos desde cero, porque ofrece una estructura de código limpia y un tutorial documentado. Por lo tanto, el desarrollador de un agente puede centrarse en escribir su propia lógica para construir entidades NGSI.

- Broker de Interoperabilidad Middleware: Como se ha indicado al principio de esta sección Orion Context Broker, es el mecanismo de interoperabilidad seleccionado. En este caso, ha sido extendido basado en los requisitos específicos del proyecto y validado en el entorno del proyecto. FIWARE sustituye al uso los brokers desplegados en los casos de uso previos (como

RabbitMQ) como elemento de soporte a la capa de abstracción middleware. Principalmente esto es debido a la experiencia previa de los socios participantes en este proyecto y a acceder a un tipo de plataformas IoT que principalmente ofrecen datos vía notificación o consultas contextuales. El desarrollo de agentes en este caso, es más accesible que crear una estructura completa de desarrollo de puentes de interoperabilidad.

- Mecanismos de seguridad: Incluye a los componentes Keyrock y Wilma del ecosistema Fiware. Wilma es un proxy PEP (proxy que mejora el rendimiento) que asegura el acceso a los agentes. KeyRock ofrece un sistema de gestión de identidad que incluye:
  - Un sistema de autenticación OAuth2 para aplicaciones y usuarios.
  - Una interfaz gráfica del sitio para la administración de la gestión de identidades.
  - Una API REST equivalente para la gestión de identidades a través de solicitudes HTTP.

Es importante resaltar, que se ha hecho uso de un cuadro de mandos y orquestador de agentes desarrollado por terceros. Ese tipo de orquestador e interfaz gráfica ha sido diseñado e implementado específicamente dentro del trabajo de la presente Tesis Doctoral. Sin embargo, este componente derivado de esta tesis doctoral no ha sido utilizado en el proyecto PIXEL, puesto que el presente autor no era el responsable de generar dicho subcomponente en el proyecto y se utilizó una solución previamente implementada por el grupo responsable del mismo. Por tanto, la solución que se ha diseñado dentro de este trabajo, ha sido utilizada y validada posteriormente en el caso de uso de DataPorts, correspondiente a la siguiente sección, donde se explica detalladamente.

8.1.3. Aportación de PIXEL a la solución propuesta

El proyecto destacó la necesidad de recopilar e intercambiar datos heterogéneos, incluidos los sensores de IoT, entre las distintas partes interesadas que operan en los puertos. Estos datos son difíciles de manejar debido a la falta de homogeneización y de herramientas para operarlos. Para satisfacer esta necesidad se ha desarrollado una solución middleware que usa una plataforma IoT ya existente como núcleo central de la interoperabilidad middleware. Concretamente el componente Orion Context Broker de la plataforma FIWARE [197][198] y el modelado de datos propuesto por la iniciativa Smart Data Models. Se trata de una decisión de diseño tomada después de analizar las necesidades específicas de los participantes en el proyecto.

A diferencia de los dos casos de uso anteriores, esta decisión hace que este mecanismo de interoperabilidad sólo pueda cubrir nativamente las funcionalidades de la plataforma seleccionada como mecanismo de abstracción.

- Por un lado, esto proporciona una solución mas sencilla y manejable en aquellos casos en los que se intenta conectar sencillas plataformas o servicios que ofrecen fuentes de datos heterogéneos.
- Por otro lado, esta decisión no será eficiente cuando se intenten conectar a dicha capa de abstracción complejas plataformas IoT, que tienen complejas descripciones en los datos o un amplio abanico de funcionalidades. No obstante, esto no ocurría en este caso de uso.

Por tanto, la principal lección aprendida de este caso de uso es que los mecanismos de interoperabilidad se deben adaptar a las necesidades técnicas reales de sus potenciales usuarios. Esto garantizará el éxito en su integración, despliegue y uso futuro. Los puntos claves obtenidos a partir de la validación y resultados en el proyecto son los siguientes:

- El despliegue y configuración de una plataforma como FIWARE como nucleo de la interoperabilidad es considerablemente más sencillo y de menor coste temporal y computacional que el despliegue de una solución como la de INTER-IoT. Además, a esta plataforma se puede conectar directamente aquellos sensores que no formen parte de ninguna plataforma IoT y estén produciendo información.
- El uso de FIWARE, como el núcleo las de interoperabilidad middleware limita las funcionalidades a las que tiene dicha plataforma disponibles. Por ejemplo, en este caso, se utiliza el registro de información contextual de FIWARE, frente a la solución de INTER-IoT que no almacenaba datos, sólo metadatos. No usar este registro complicaría algunas acciones, como, por ejemplo, los mecanismos de suscripciones o notificaciones. No obstante, almacenar esta información contextual no era un problema en el proyecto PIXEL.
- El uso de una plataforma como FIWARE permite aprovechar los resultados de la comunidad de desarrolladores que tiene detrás y los habilitadores disponibles, como los de persistencia de la información o los de seguridad. El uso de una plataforma como la ofrecida por INTER-IoT permite aprovechar las funcionalidades de todas las plataformas conectadas a ella, entre ellas FIWARE, lo que ofrece un mayor abanico de funcionalidades para aquellos que desarrollen sobre ella.

- El desarrollo de agentes tiene menos coste de desarrollo y complejidad que el de puentes. Un agente, independientemente de su naturaleza activa o pasiva, básicamente envía metadatos y datos a un puerto. En el proyecto DataPorts se ha desarrollado un SDK para facilitar esta tarea. Los puentes se encargan de ejecutar y gestionar funciones proporcionadas por las plataformas y devolver sus resultados. Esto implica que haga falta un desarrollador que adapte dicha lógica para una instancia de una plataforma nueva que no tenga un puente desarrollado previamente.
- Es importante analizar las fuentes de datos para analizar la complejidad del middleware a utilizar como capa de abstracción. En el caso de la solución ACTIVAGE se trataba con mas de cinco plataformas IoT de gran complejidad. Manejar este tipo de plataformas y la naturaleza del proyecto ACTIVAGE con la solución propuesta por PIXEL implicaria hacer demasiados cambios para adaptar la plataforma FIWARE. No seria óptimo porque se estaría reprogramando todo el diseño e implementación de FIWARE, para obtener finalmente una solución similar a la que ofrece INTER-IoT. En el caso de PIXEL no se estaban manejando plataformas que tuvieran una gran complejidad.
- No es necesario sobrecargar al desarrollador e integrador de plataformas con una solución que ofrece más funcionalidades de las que necesita. Esto puede provocar que no entiendan la necesidad de tener que implementar ciertas funcionalidades en los conectores de las que no perciben el recibir una ganancia directa. Esto puede llevar a que puedan sentir rechazo por el mecanismo de interoperabilidad que se les propone.
- La solución propuesta mediante FIWARE podría conectarse a una solución como la ofrecida en INTER-IoT. Se haría como se hace con cualquier otra plataforma mediante la adaptación de un puente de conexión de FIWARE. Por tanto, no se trata de dos soluciones opuestas, sino dos soluciones que pueden complementarse en caso de ser necesario.

El análisis de esta situación, ofrecida por el proyecto PIXEL, dió lugar a nuevas decisiones de diseño en la presente Tesis Doctoral, que finalmente fueron ejecutadas de manera completa en el proyecto DataPorts. Es por eso, que se proporcionan detalles más especificos en la sección de validación y resultados del caso de uso DataPorts. Allí se ofrece la instanciación completa del mecanismo de interoperabilidad y se proporcionan los enlaces y descripciones de las publicaciones en que se han presentado los resultados del trabajo conjunto entre ambos proyectos.

Por último, se ha participado en varios eventos y se han publicado dos artículos científicos, realizados en paralelo con el siguiente caso de uso de Da-

taPorts. En estos artículos y eventos se ha ofrecido una visión detallada de las implicaciones de algunos de los mecanismos de interoperabilidad ofrecidos por esta Tesis Doctoral para ofrecer soluciones que aporten un valor añadido y para integrase a nuevas tecnologías incipientes. Para ofrecer una visión completa del objetivo de dichos trabajos, serán brevemente presentados en la última sección del presente capítulo.

8.2. Visión general del proyecto

Tras haber analizado brevemente el proyecto PIXEL y sus resultados desde el punto de vista de la interoperabilidad middleware, se toman las principales conclusiones como referencia para llevar a cabo la implementación de la solución de interoperabilidad necesaria en el proyecto DataPorts. Principalmente, se toman como punto de partida de los componentes de acceso a datos y de interoperabilidad que se describirán en las siguientes secciones.

A continuación se retoma la descripción del proyecto DataPorts. El principal resultado del mismo es su plataforma de datos, que interactuará con los puertos de la siguiente manera:

- Adquiriendo, procesando y almacenando datos procedentes de las diferentes fuentes de datos e infraestructuras digitales existentes en los puertos marítimos digitales.
- Obteniendo de los propietarios de dichos datos las reglas para compartirlos y comercializarlos, ofreciéndoles una propuesta de valor clara. Los consumidores de datos tendrán acceso a ellos bajo las reglas mencionadas, con la aplicación de contratos inteligentes, si es necesario.
- Proporcionando, además de los servicios de análisis de datos ofrecidos por la plataforma, un conjunto de aplicaciones de Inteligencia Artificial (IA) y cognitivas a la comunidad portuaria [199], destinadas a resolver problemas específicos del puerto e impulsar la evolución hacia puertos inteligentes y cognitivos.

La arquitectura de DataPorts también sigue el enfoque de los Espacios de Datos Internacionales (IDS), en el sentido de que ofrece: i) una conectividad infinita; ii) confianza entre diferentes dominios de seguridad; y iii) gobernanza para la economía de datos. Al mismo tiempo, también está en consonancia con la declaración de objetivos de la Asociación de Espacios de Datos Internacionales (IDSA) sobre: i) intercambio seguro de datos; ii) casos de uso; y iii) modelos de negocio.

En pocas palabras, IDS es un entorno virtual que aprovecha las normas y tecnologías existentes, así como modelos de gobernanza bien aceptados en la economía de los datos, para facilitar el intercambio y la vinculación de datos de forma segura y estandarizada en un ecosistema de confianza [200][201].

Los principales requisitos estratégicos a los que apunta IDS son:

- Confianza: Cada participante es evaluado y certificado antes de que se le conceda acceso al ecosistema empresarial de confianza.
- Seguridad y soberanía de los datos: Todos los componentes de IDS se apoyan en medidas de seguridad de última generación.
- Ecosistema de datos: El diseño de la arquitectura de IDS no requiere una capacidad central de almacenamiento de datos. En su lugar, persigue la idea de un almacenamiento de datos descentralizado, lo que significa que los datos permanecen físicamente con el respectivo propietario de los mismos hasta que se transfieren a una parte de confianza.
- Interoperabilidad estandarizada: El Conector IDS, al ser un componente central de la arquitectura, se implementa en diferentes variantes y puede adaptarse a distintos fines.
- Aplicaciones de valor añadido: Las aplicaciones permiten prestar servicios sobre los procesos de intercambio de datos. Esto incluye servicios de procesamiento de datos, alineación de formatos de datos y protocolos de intercambio de datos. Además, se pueden proporcionar servicios de análisis de datos mediante la ejecución remota de algoritmos.
- Mercado de datos: El Espacio Internacional de Datos permite la creación de nuevos servicios basados en datos que hacen uso de las aplicaciones de datos. También fomenta nuevos modelos de negocio para estos servicios, proporcionando mecanismos de compensación y funciones de facturación, y creando soluciones de intermediación y mercados específicos para cada dominio. Además, los Espacios Internacionales de Datos proporcionan plantillas y otros apoyos metodológicos para que los participantes los utilicen al especificar la información sobre restricciones de uso y solicitar información legal.
- Reutilización de tecnologías existentes: La iniciativa IDS pretende utilizar tecnologías existentes (del ámbito del código abierto), y estándares (por ejemplo, los estándares semánticos del W3C).
- Contribución a la estandarizacción: La iniciativa IDS apoya la idea de las pilas de arquitectura estandarizadas.

Al proponer una arquitectura para el intercambio seguro de datos y la compartición de datos de confianza, el IDS contribuye al diseño de arquitecturas empresariales en escenarios de digitalización comercial e industrial. La arquitectura está diseñada con el objetivo de superar las diferencias entre los enfoques descendentes y los ascendentes.

La plataforma de intercambio de datos DataPorts incluirá análisis, exploración, descubrimiento y adaptabilidad y apoyo al desarrollo de un mercado de datos. Para garantizar la compatibilidad con el modelo de referencia del IDS, los componentes ofrecerán lo siguiente:

- Los propietarios y proveedores de datos tendrán la posibilidad de describir el conector para indicar claramente el tipo y las condiciones de los datos que se ofrecerán a otros actores cuando accedan a ellos. Esta operación se realizará a través de los métodos de la cadena de bloques (Blockchain) y será implementada por los bloques de abstracción y virtualización de datos y el de gobernanza de datos.
- El bloque Blockchain actuará como Broker, registrando la descripción de todos los datos y proporcionando a los consumidores de datos aquellos datos más adecuados en función de sus atributos.
- Los conectores estarán a disposición de los consumidores de datos para permitir el acceso a los diferentes conjuntos de datos en la plataforma, de acuerdo con las reglas definidas por los propietarios y a través de Smart Contracts si es necesario. Al mismo tiempo, el bloque de análisis de datos utilizará esos conectores para acceder a los datos con el fin de permitir la explotación de la plataforma por parte de los beneficiarios.
- El bloque de abstracción y virtualización de datos actuará como proveedor de vocabulario, gestionando las ontologías y los metadatos para proporcionar un vocabulario específico del dominio del puerto.
- Las diferentes aplicaciones y servicios proporcionados por la plataforma se someterán a la certificación de los organismos de certificación aprobados por la IDSA.

La figura 8.6 presenta la arquitectura de la plataforma de datos DataPorts desde la perspectiva de alto nivel que describe los principales componentes de la plataforma incluyendo las principales funcionalidades.

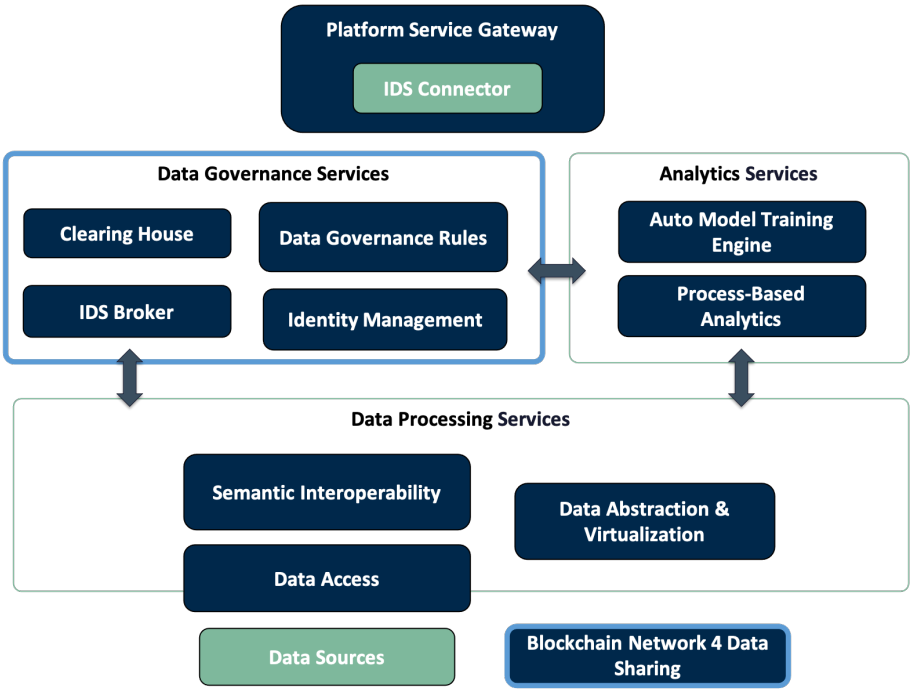


Figura 8.6: Visión general de la arquitectura de la plataforma DataPorts.

De manera más detallada la Plataforma de Datos incluye los siguientes bloques:

Componente de acceso a datos: Es el punto de entrada a DataPorts. Es el componente responsable de acceder a las distintas fuentes de datos conectadas a la plataforma DataPorts. Entre estas fuentes de datos se encuentran las siguientes:

- Datos internos. Datos de diferentes fuentes (diferentes actores del ecosistema portuario). Por ejemplo, datos del Port Community System, dispositivos IoT, o también aplicaciones de terceros que se ejecutan en el puerto.
- Datos externos federados. Datos procedentes de diferentes plataformas DataPorts. Datos procedentes de otras plataformas o ecosistemas con un modelo de datos común.
- Datos abiertos. Fuentes de datos públicas. Por ejemplo, una API de terceros.

Componente de Interoperabilidad Semántica: La plataforma DataPorts garantizará la interoperabilidad semántica con el fin de proporcionar una visión virtualizada unificada de los datos para su uso por parte de los diferentes consumidores de datos y los servicios de análisis de datos. En el momento de la escritura de esta Tesis Doctoral, el proyecto DataPorts está desarrollando un componente semántico para describir los datos de los puertos junto con mapeos a vocabularios estándar con el fin de simplificar la reutilización de las aplicaciones de datos para la analítica y la previsión. En particular, este componente semántico codificará el conocimiento del dominio de los expertos en la materia y, por tanto, podrá ser reutilizado y explotado por los expertos en datos directamente, potenciando así la construcción de aplicaciones portuarias cognitivas.

El componente de interoperabilidad semántica expone una API unificada para acceder a los datos de las diferentes fuentes de datos conectadas a la plataforma DataPorts, proporcionando datos históricos en tiempo real y por lotes a los consumidores de datos. En colaboración con el Componente de Acceso a Datos, proporciona una solución de semántica compartida de datos para la interoperabilidad de diversas fuentes de datos utilizando ontologías existentes. Para garantizar que el consumidor de datos sea capaz de entender la estructura y el significado de los datos, los datos de salida siguen el formato común y la ontología desarrollada, independientemente de la fuente de datos que los genere. En cuanto a los metadatos, el Componente de Interoperabilidad Semántica obtiene la información sobre las diferentes fuentes de datos del Gestor de Acceso a Datos y la almacena en un Registro de Metadatos para que esté disponible para los demás subcomponentes de esta capa. Además, los metadatos se envían al broker IDS para proporcionar información a los consumidores de datos sobre las fuentes de datos disponibles.

Componente de Abstracción y Virtualización de Datos: Dada la gran variedad de fuentes, la plataforma tendrá que hacer frente a la variedad de datos, la diferente velocidad según los sistemas heredados y la veracidad según el tipo de fuente. La plataforma de datos utilizará las ventajas de la interoperabilidad semántica y su interacción con el componente de Abstracción y Virtualización de Datos (DAV) con el fin de proporcionar una visión virtualizada unificada de los datos para su uso por los diferentes consumidores de datos y los servicios de análisis de datos. Este componente tiene como objetivo preparar adecuadamente las entradas de datos de varias fuentes dentro de la arquitectura genérica de DataPorts, manteniendo los metadatos de todos los flujos y finalmente exportando los conjuntos de datos limpios/procesados a través de las APIs REST expuestas, poniéndolos así a disposición de cualquier cliente potencial. Los flujos de datos persistentes serán la fuente básica de carga del componente.

Componente de gobernanza de los datos: La gobernanza de datos es el mecanismo que permite manejar datos de alta calidad, supervisando su ciclo de vida completo. Aumenta la coherencia y la confianza de los datos registrados, mejorando la seguridad de estos y minimizando el riesgo de no cumplir con la normativa pertinente. La plataforma de datos proporcionará todas las herramientas para compartir y comercializar los datos de forma segura y fiable, teniendo en cuenta las normas propuestas por los proveedores de datos para los consumidores de datos, y ofreciendo una propuesta de valor clara a los propietarios de los datos, implementando contratos inteligentes (smart contracts) si es necesario. Además, la gobernanza de los datos mantendrá la información sobre quién es el propietario de los datos, cuál es la calidad de estos y cuáles son los usos potenciales de los datos.

Componente de Blockchain: Para mantener la procedencia de los datos que entran en la plataforma e implementar las funcionalidades de la gobernanza de datos, la plataforma DataPorts implementa la tecnología Blockchain tomando como referencia Hyperledger Fabric [202], la implementación de Blockchain con permisos más madura y de código abierto disponible en la actualidad.

Componente de privacidad y seguridad: La plataforma de datos proporciona un entorno seguro y fiable de extremo a extremo para las actividades en cuestión mediante un enfoque integral de ciberseguridad que al mismo tiempo incluye la privacidad. Sobre esta base, la plataforma ofrece un entorno seguro y de confianza en el que la mayoría de los riesgos se mitiguen hasta un nivel aceptable, proporcionando una gobernanza de los esfuerzos de seguridad con el fin de crear un modelo dinámico capaz de ajustarse a las nuevas amenazas.

Componente analítica avanzada de Big Data: DataPorts ofrece Big Data Analytics as a Service (BDAaaS), proporcionando un nivel de abstracción a los desarrolladores de aplicaciones sobre los detalles de implementación y puesta en marcha de la plataforma de datos en cuanto a configuración, servicios, adaptabilidad y despliegue. Este módulo aprovecha la enorme cantidad de datos disponibles en la plataforma, y proporciona servicios independientes de la aplicación, como el reconocimiento de patrones, el análisis predictivo y prescriptivo, la previsión de tendencias, etc. Hay dos tipos de subcomponentes de este tipo: el componente de análisis basado en procesos y el componente de motor de formación de modelos automáticos [203][204][205][206].

Como parte de la ejecución del proyecto, el despliegue, las pruebas y la evaluación de la plataforma de datos en la vida real se llevarán a cabo en dos sitios de demostración locales (Valencia y Tesalónica), donde la plataforma se desplegará y se conectará a las infraestructuras digitales existentes de cada puerto, se le dará acceso a las diferentes fuentes de datos y será operada por las partes interesadas pertinentes, a las que la plataforma de datos ofrecerá

servicios impulsados por datos para abordar problemas concretos. Además, la plataforma DataPorts se probará y mostrará a escala paneuropea en una base interportuaria en casos de uso global que implican indirectamente a múltiples puertos y actores. En resumen, los casos de uso son:

Puerto de valencia

- Seguimiento de las operaciones de transporte.
- Servicios de análisis e intercambio de datos de la autoridad portuaria.
- Compartir masa bruta verificada.
- Nota de envío digital.

Puerto de Thessaloniki

- Aplicación basada en datos para decisiones estratégicas y en tiempo real.
- Identificación de permiso para la recogida de contenedores.
- Estadísticas para la predicción.
- Predicciones de colas.
- Facilitación de Pasajeros, Profesionales y Visitantes del Puerto.
- Estadísticas para pasajeros / visitantes.

Caso de uso global: contenedores inteligentes

- Seguimiento de contenedores mediante dispositivos IoT.
- Uso de datos de contenedores inteligentes en el puerto de Thessaloniki.

Casos de uso de global: integración del sistema de gestión de puertos

- Notificaciones de eventos de embarcaciones.

DataPorts contribuye al objetivo de *Apoyar la aparición de mercados de datos y la economía de los datos.* Este proyecto aborda tanto los objetivos estratégicos de alto nivel establecidos por la Unión Europea como las necesidades prácticas de las partes interesadas y los usuarios finales relacionadas con las plataformas de datos, los mercados de datos y la economía de los datos. DataPorts refleja la voluntad de Europa de crear y ofrecer plataformas de datos integradas dentro y a través de diferentes dominios de aplicación. La plataforma, la metodología y las herramientas propuestas tienen la ambición de ayudar

a los agentes relevantes del transporte de los puertos marítimos de la UE (por ejemplo, las autoridades portuarias, las compañías navieras o los integradores y operadores de infraestructuras) a proporcionar servicios relacionados con los datos integrados, seguros, privados e interoperables. DataPorts proporcionará APIs, habilitadores y herramientas para ampliar aún más el uso de datos escalables que tendrán el potencial de ser extendidos a otros sectores verticales. Los resultados del proyecto DataPorts se alinean con uno de los pilares de la estrategia de la Big Data Value Association (BDVA) centrada en las Plataformas de Datos de Próxima Generación e Inteligencia Artificial. Estas son consideradas como un activo crítico para obtener valor real de los datos y contribuir a una posición de liderazgo de Europa en el desarrollo de plataformas de datos de próxima generación. Por último, la plataforma DataPorts contribuirá al desarrollo de los sectores marítimo y navieros europeos, fomentando la transición de los puertos marítimos de digitales y conectados a inteligentes y cognitivos, proporcionando nuevos servicios y modelos de negocio basados en datos y mejorando las operaciones.

8.3. Componentes de la arquitectura e implementación involucradas

Los componentes de la arquitectura e implementación involucradas se corresponden con los elementos señalados en rojo en la Figura 8.7.

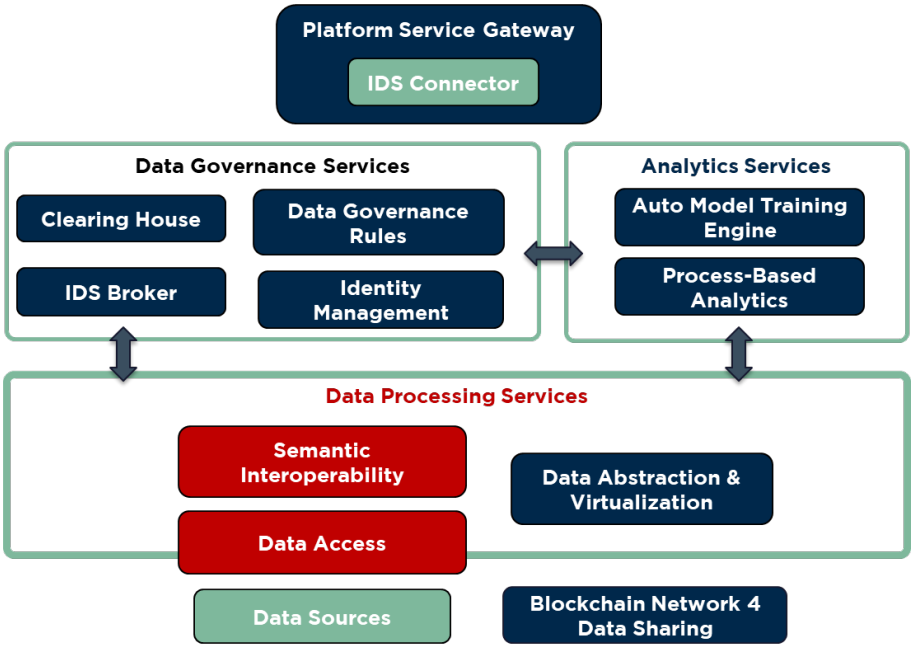


Figura 8.7: Componentes de la arquitectura de la tesis involucrados en la plataforma DataPorts.

El componente de acceso a datos es la capa más baja de la plataforma DataPorts. Accede y se conecta a los datos y alimenta el componente responsable de la interoperabilidad. Es el único componente de la plataforma DataPorts que es capaz de interactuar con las fuentes de datos y proporcionar acceso a sus datos sin procesar. Esta tarea es responsabilidad de los agentes, quienes están a cargo de acceder a las diferentes fuentes de datos, convertir estos datos en modelos de datos entendidos por DataPorts y enviar estos datos a capas superiores (la funcionalidad de los agentes se detallará en secciones posteriores).

El componente tiene una interfaz de programación de aplicaciones con APIs del tipo Transferencia de Estado Representacional (REST) que administra las diferentes operaciones CRUD sobre los agentes. Este componente también cuenta con una Interfaz de Usuario (UI) encargada de mostrar información relacionada con las diferentes imágenes (imágenes Docker de los agentes) y agentes que trabajan en la plataforma. Esta interfaz de usuario también tiene un Kit de Desarrollo de Software (SDK) para facilitar la creación de agentes para desarrolladores.

Además, este componente también está conectado al componente de virtualización de datos, que recibe los datos transformados bajo demanda a través de su Localizador de Recursos Uniforme (URL) de devolución de llamada declarado en ese momento para ejecutar los agentes (tipo bajo demanda). El componente de interoperabilidad semántica actúa como una capa de middleware entre el componente de acceso a datos y las capas superiores (componente de virtualización y abstracción de datos y motor de entrenamiento de modelos automáticos). Por lo tanto, en la arquitectura, el componente de interoperabilidad semántica está encima del componente de acceso a datos y debajo de los componentes que obtienen los datos de su API, como el componente de virtualización y abstracción de datos y el motor de entrenamiento de modelos automáticos. La API de este componente ofrece acceso a los datos y metadatos disponibles. Hay dos modos de acceso a los datos: el primero se basa en el uso de suscripciones para recibir datos casi en tiempo real, mientras que el segundo permite consultas bajo demanda para recibir datos históricos o datos casi en tiempo real. Además, el componente de interoperabilidad semántica proporciona un repositorio y una descripción para el modelo de datos de DataPorts y la ontología de DataPorts. El componente de interoperabilidad semántica está conectado al componente Blockchain / Data Governance a través de un cliente Blockchain, y al IDS Broker a través de un conector IDS.

El componente de Abstacción de Datos y virtualización (DAV) recibe datos de las capas de acceso a datos e interoperabilidad semántica, así como de cualquier posible proveedor de datos que resida fuera de estos dos componentes. Por otro lado, DAV proporciona datos al motor de entrenamiento de modelos automáticos y a cualquier otro destinatario potencial de datos, a través de API RESTful expuestas. En resumen, DAV sirve como enlace entre los principales proveedores de datos de DataPorts y los consumidores, obteniendo datos de entrada persistentes (principalmente de la capa de interoperabilidad semántica) y reenviando estos flujos a cualquier destinatario interesado (principalmente el motor de entrenamiento de modelos automáticos).

8.4. Descripción de la solución implementada

Los siguientes componentes (Figura 8.8) de la solución propuesta en la Tesis Doctoral se han validado en el proyecto:

- El componente de acceso a los datos ofrece una aplicación Web para gestionar y ejecutar agentes que integran los datos en la plataforma DataPorts.

- El componente de interoperabilidad, que proporciona una API común y un modelo de datos para acceder a los datos y metadatos disponibles de las respectivas fuentes de datos (sistemas, sensores, plataformas...).

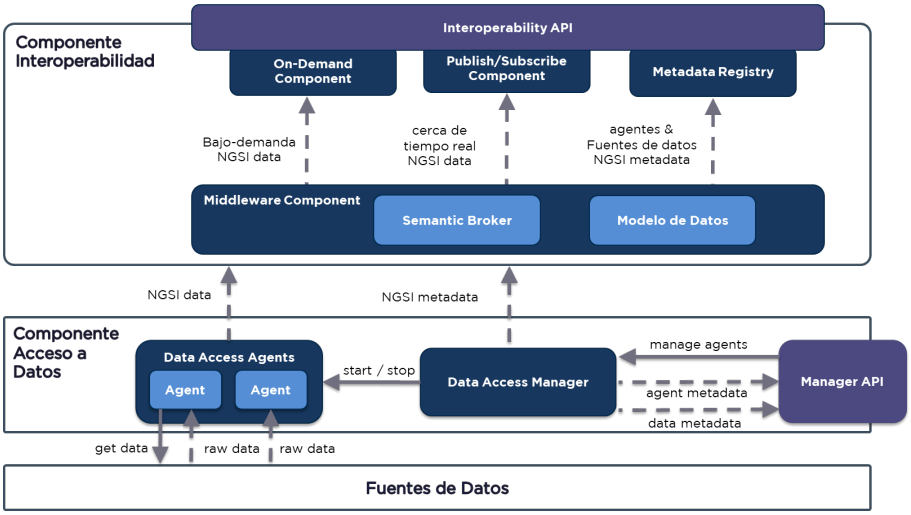


Figura 8.8: Diseño componentes interoperabilidad y acceso a datos.

El **componente de acceso a datos:** Es el responsable de recopilar, transformar y publicar datos de diferentes fuentes de datos a la plataforma. Admite muchos tipos de fuentes, como análisis de archivos, consultas de bases de datos o llamadas a API. El componente es extensible y, por lo tanto, no se limita a acceder a las diversas fuentes de datos que se comparten en la plataforma en este proyecto.

Para ello, proporciona diferentes funcionalidades que se pueden dividir en dos secciones: procesamiento de datos y gestión de componentes.

- En primer lugar, el componente tiene la capacidad de obtener datos de fuentes de datos de muchos tipos diferentes, gracias a la implementación de los agentes, que se desarrollarán para acceder a cada uno de los tipos de fuentes con una periodicidad personalizable, procesando y transformando la información recibida en modelos de datos soportados por la plataforma. Una vez que los datos sean comprensibles por la plataforma, los agentes interactuarán con otros componentes, publicando, compartiendo o enviando el resultado de los pasos anteriores.

- En segundo lugar, se facilitan las herramientas y mecanismos para gestionar la ejecución y el estado de los procesos. El componente proporciona una API y una interfaz de usuario para la gestión de los procesos. Los agentes se pueden crear, eliminar, iniciar y detener, o también ejecutar según el tipo de agente. Los registros, las propiedades y el estado se pueden verificar a través de las funcionalidades del componente. El componente de acceso a datos también proporciona las herramientas para ayudar al desarrollo de nuevos agentes y la posterior integración en el componente.

Hay dos tipos de agentes: publicación-suscripción y bajo demanda. Los primeros se ejecutan una vez o periódicamente según su configuración. Los agentes bajo demanda se ejecutan cuando se realiza una llamada externa. El componente expone una API para la gestión de los agentes y también para llamar a la ejecución bajo demanda de los mismos. La información que se genera como resultado de la ejecución de los agentes se puede publicar en un intermediario de contexto, que se comparte con el componente de interoperabilidad semántica o se envía a un punto final de devolución de llamada. La ejecución de los agentes y las suscripciones se gestionan desde la capa de Interoperabilidad, pero los resultados se pueden devolver tanto a ese componente como a un destino determinado. Entonces, el componente de Acceso a Datos interactúa principalmente con un componente en la plataforma DataPorts que es el Componente de Interoperabilidad Semántica, otros componentes o sistemas como receptores de las respuestas bajo demanda y con todas las diferentes fuentes de datos fuera de la plataforma.

El **componente de interoperabilidad middleware:** Este componente proporciona un repositorio con el modelo de datos de DataPorts y la ontología de DataPorts. Ofrece una definición del modelo de datos usando un esquema JSON y documentos de contexto JSON-LD. Su objetivo es integrar las siguientes ontologías y modelos de datos: Fiware Data Models, IDSA Information Model, UN / CEFACT model y SAREF ontology [207][208]. Además, este componente actúa como una capa de middleware entre las fuentes de datos y el componente de virtualización y abstracción de datos. Recibe la información del componente de acceso a datos en un modelo de datos común e interactúa con este componente a través de un intermediario semántico.

El componente realiza las siguientes acciones:

- Gestiona y almacena localmente la información relativa a las fuentes de datos conectadas. Las fuentes de datos (y sus agentes correspondientes en el componente de acceso a datos) se registran localmente en el registro de metadatos. De esta forma, los metadatos están disponibles para todos los subcomponentes de esta capa.

- El Registro de Metadatos ofrece la descripción de las fuentes de datos disponibles para los componentes internos de la Plataforma DataPorts.
- Cada instancia local del Componente de Interoperabilidad Semántica registra en el broker IDS las fuentes de datos que gestiona para que la descripción de las fuentes de datos esté disponible para los consumidores de datos de otras organizaciones.
- Proporciona una interfaz para publicar y suscribir flujos de datos.
- Proporciona datos bajo demanda. Esto incluye tanto los datos históricos como los últimos valores.
- Proporciona interacción API REST con datos vinculados.
- Valida que los datos enviados a los consumidores de datos estén en el formato correcto.
- Distribuye los datos a los componentes de las capas superiores, como el componente Acceso a datos y virtualización, el motor de predicciones automáticas y el análisis basado en procesos.

El componente de interoperabilidad semántica se compone de los siguientes subcomponentes:

- Componente de capa middleware:
  - Semantic Broker: Middleware Broker para recibir información e interactuar con la capa de acceso a datos
  - Modelo de Datos: Repositorio con la Ontología y la descripción del Modelo de Datos.
- Registro de metadatos: Recupera la información sobre las fuentes de datos disponibles y la semántica. Incluye información sobre las funciones de los agentes.
- Componente Publicar / Suscribir: Gestiona las suscripciones con las fuentes de datos y habilita el acceso a datos en tiempo real, interactuando con los agentes implementados como publicación / suscripción.
- Componente On-Demand: Gestiona el acceso a datos bajo demanda (por ejemplo, consultas históricas de datos por lotes), interactuando con los agentes implementados como bajo demanda.

- API de Interoperabilidad Semántica: API común para acceder a las funcionalidades del Componente de Interoperabilidad Semántica. Se implementa como API Gateway para facilitar la integración con otros componentes como Blockchain y Data Governance.

Las tecnologías seleccionadas para implementar la solución aparecen en la Figura 8.9.

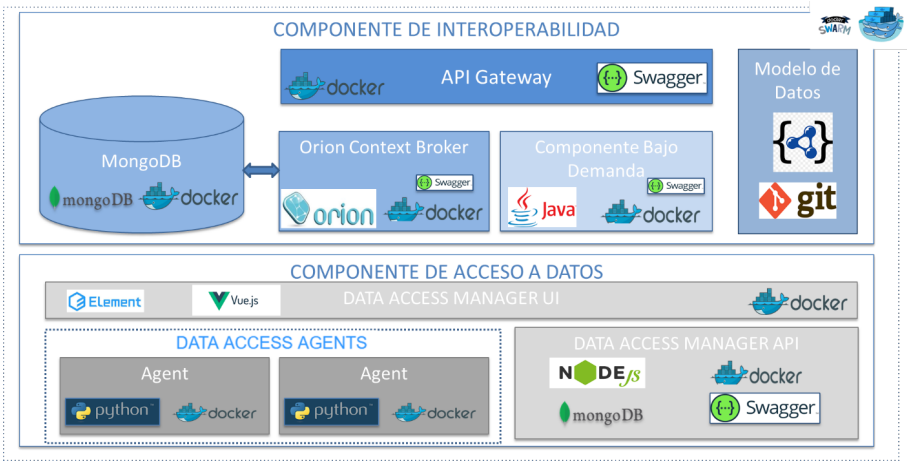


Figura 8.9: Tecnologías seleccionadas para la implementación de los mecanismos de interoperabilidad en DataPorts.

En cuanto a la implementación, Orion Context Broker proporciona las funciones de Interoperabilidad Middleware, Metadata Registry y una interfaz de componente de publicación / suscripción. Orion recibe los datos de los agentes en el componente de acceso a datos en formato Next Generation Service Interfaces (NGSI), mantiene un registro de las fuentes de datos conectadas y envía notificaciones a los consumidores de datos en las capas superiores. Orion Context Broker se desarrolló como parte de la plataforma FIWARE. Orion es una implementación de Publish / Subscribe Context Broker GE, que proporciona una interfaz NGSI. Orion permite al usuario administrar todo el ciclo de vida de la información de contexto, incluidas actualizaciones, consultas, registros y suscripciones.

Dado que Orion Context Broker sólo almacena el último valor de cada entidad, se ha incluido un módulo personalizado bajo demanda. Este módulo proporciona una API de consulta para recuperar datos históricos y se comunica

con Orion Context Broker para recuperar información sobre las fuentes de datos y el modelo de datos.

Para garantizar que los datos enviados a los consumidores de datos sigan el modelo de datos común, se incluye un módulo de validación del modelo de datos.

La opción elegida para construir estos agentes es usar pygnsi, un marco de desarrollo en Python para escribir entidades NGSI en Fiware Orion [209]. Los agentes se integran en el componente como contenedores docker, por lo que se integrarán en el componente siguiendo la metodología y herramientas definidas y proporcionadas.

Finalmente, respecto al administrador de acceso a datos. Los subcomponentes definidos para este bloque son:

- Interfaz de gestión. Desarrollado usando Element o vue-element-admin. Es una biblioteca de componentes para desarrolladores basada en Vue 2.0.

- API de gestión. API REST desarrollada en Node.js siguiendo el patrón MVC. Internamente, hace uso de la API de Docker para administrar agentes como contenedores. La especificación de la API se realiza en Swagger. La API hace uso de MongoDB para almacenar las plantillas utilizadas para crear los agentes mediante el SDK.

Por otro lado, la relación entre el componente de acceso a datos y el componente de interoperabilidad merece una mención especial. Las siguientes características permiten implementar ambos componentes juntos:

- La base de datos común (MongoDB) se utiliza para almacenar las plantillas que se distribuyen de forma predeterminada al implementar DataPorts. Lo mismo se aplica a los modelos de datos que son compatibles con la plataforma de forma predeterminada.

- Interfaz gráfica común. Se ha incluido una opción en la interfaz de usuario de gestión para administrar las fuentes de datos y las suscripciones disponibles en ORION.

- Uso del mismo modelo de datos interno. El modelo de datos de entidad desarrollado a partir del Componente de Interoperabilidad Semántica para registrar agentes, fuentes de datos y modelos de datos disponibles en ORION, se comparte con el Componente de Acceso a Datos. Entonces, cada vez que se inicia un agente, genera un registro de la entidad definida.

Con respecto al despliegue de los componentes: los agentes, el gestor gráfico, Orion Context Broker, el componente On Demand y la API de interoperabilidad se despliegan y se ofrecen mediante Docker. Estos componentes se pueden implementar juntos usando docker-compose. Con respecto a las imágenes de Docker, se proporciona una imagen precompilada de los componentes diseñados especificamente, como por ejemplo, el componente bajo demanda. Las imágenes oficiales de Docker de Orion Context Broker y Traefik (utilizado como API Gateway) están disponibles en DockerHub.

8.5. Validación y resultados

Los principales resultados son:

- Abordar casos de uso del mercado de datos de la vida real en dos puertos marítimos europeos relevantes, dos casos de uso global y comunidades relacionadas.
- Diseñar y validar la próxima generación de servicios interoperables avanzados relacionados con los datos para dar soporte a soluciones de Inteligencia Artificial.
- Definir una metodología de ingeniería que facilite la aplicación de la arquitectura y las herramientas para apoyar los procesos de intercambio de datos cognitivos, conscientes de la privacidad y seguros.
- Definir, diseñar e incorporar un enfoque interoperable novedoso, escalable y resistente para el intercambio de datos.

Los componentes han ofrecido los siguientes resultados concretos:

- Componente de acceso a los datos: Identificar las diferentes fuentes de datos que se integran en la plataforma de datos DataPorts. Incluir los mecanismos para facilitar la gestión de los datos.
- Componente de interoperabilidad: Definir mecanismos habilitadores de la interoperabilidad semántica con plataformas de datos, dispositivos IoT, robots y otras fuentes de datos, y desarrollar las herramientas para facilitar la generación de interfaces de acceso a los datos, además de definir mecanismos para la gestión de los datos.

Concretamente, el principal valor de integrar la solución propuesta en DataPorts, es que se hace uso de ella en un entorno en el que se combina una arquitectura basada en IDS, utiliza Blockchain y soluciones de soberanía de

datos, proporciona funcionalidades de Big Data e Inteligencia Artificial. Por tanto, es un marco perfecto para evolucionar y ser un mecanismo habilitador de interoperabilidad que facilite el uso de nuevas tecnologías.

Los resultados principales son habilitar el acceso interoperable a los datos a los componentes previamente descritos de la plataforma y a herramientas gráficas, servicios o aplicaciones, tal y como se muestra a continuación.

Fuentes de datos

Como muestra la Figura 8.10 en la solución se consideran diferentes fuentes de datos heterogéneas como:

- Plataformas IoT.
- Servicios y Aplicaciones de plataformas IoT.
- FTP.
- Archivos (como CSV, TXT...).
- API.
- Base de datos SQL.
- Base de datos que no es SQL (por ejemplo, Elasticsearch, MongoDB...).

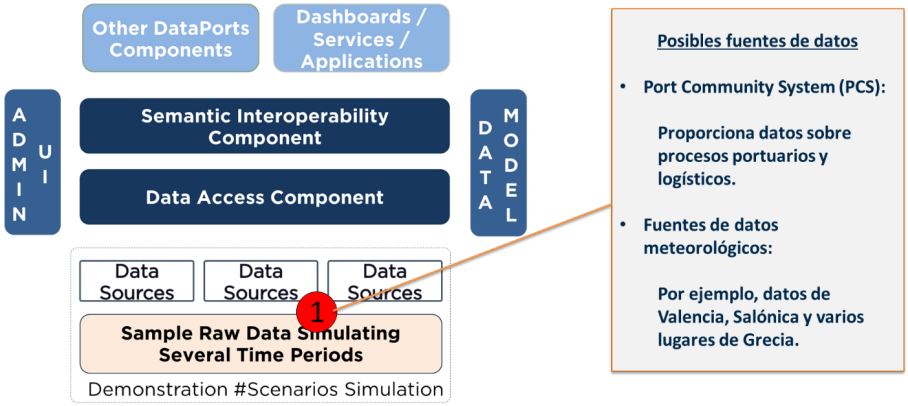


Figura 8.10: Escenarios de validación (I).

Agentes

Los agentes (Figura 8.11) acceden a las fuentes de datos y pueden ser:

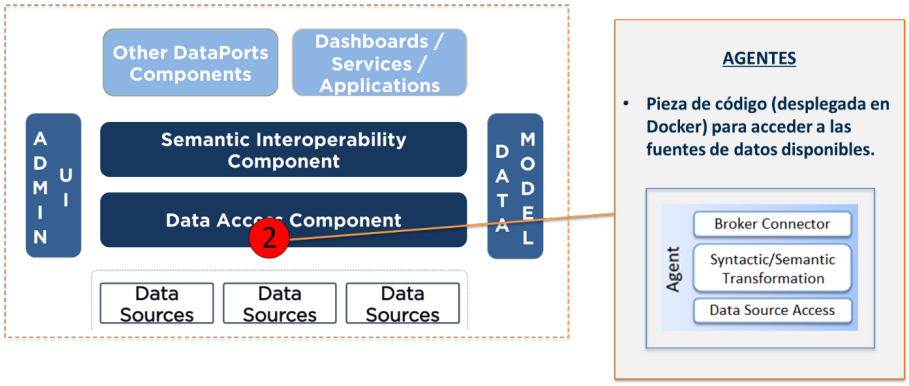


Figura 8.11: Escenarios de validación (II).

- Publicación-suscripción. Este agente se utiliza para suscribirse a una fuente de datos para obtener información periódicamente.
- Bajo demanda. Este tipo de agente se utiliza cuando se requiere acceso a una fuente de datos de forma única y para grandes volúmenes de datos, como, por ejemplo, acceso a datos históricos. En este caso, el agente no es un mecanismo de publicación-suscripción y, por tanto, los datos no se devuelven a ORION. Se devuelve a una URL de devolución de llamada que debe indicarse en el momento de crear el agente.

Interfaz de administración y SDK de creación de agentes de interoperabilidad

La API de gestión del componente de acceso a datos proporciona las funcionalidades de dicho componente y la forma de ejecutarlas (Figura 8.12). Estas funcionalidades están enfocadas a la creación y gestión de agentes, como son:

- Crear plantilla de agente.
- Importar nuevo agente. Y una vez integrados los agentes en el componente:
- Crear instancia de instancia.
- Iniciar agente.
- Detener agente.
- Ejecutar agente.
- Eliminar agente.
- Verificar los registros de los agentes.
- Verificar las propiedades del agente.

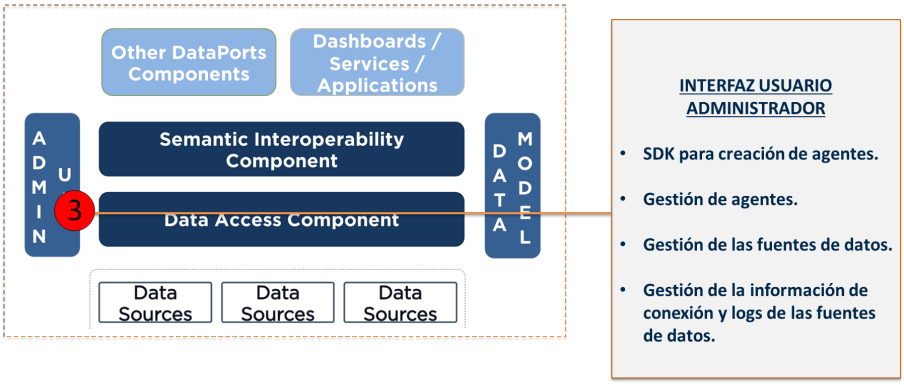


Figura 8.12: Escenarios de validación (III).

La interfaz de usuario admite la ejecución y gestión de las siguientes funcionalidades:

- Crear instancias de los agentes mediante la aplicación gráfica.

- Ofrecer una interfaz interactiva para gestionar los agentes.
- Ofrecer capacidades para cargar agentes.
- Proporcionar un asistente y plantillas para crear agentes.

Por ejemplo, en las siguientes Figuras 8.13, 8.14, 8.15 y 8.16 se muestra el asistente de creación de agentes y la Figura 8.17 muestra el gestor de agentes:

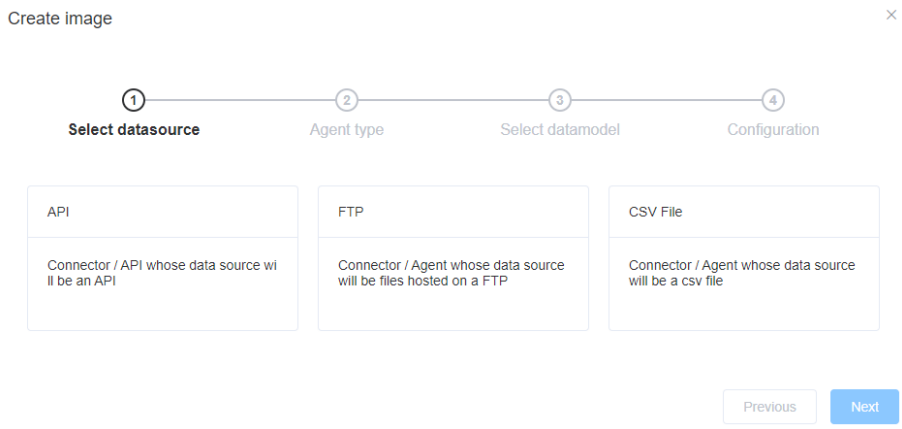


Figura 8.13: Asistente creación agentes de DataPorts (I).

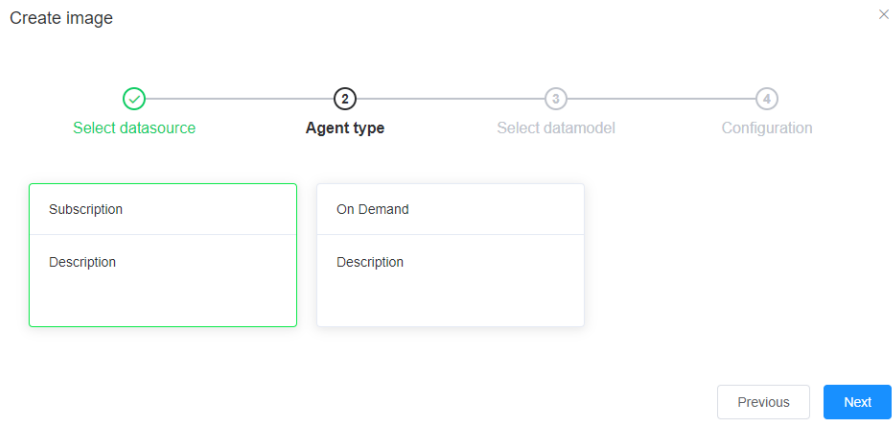


Figura 8.14: Asistente creación agentes de DataPorts (II).

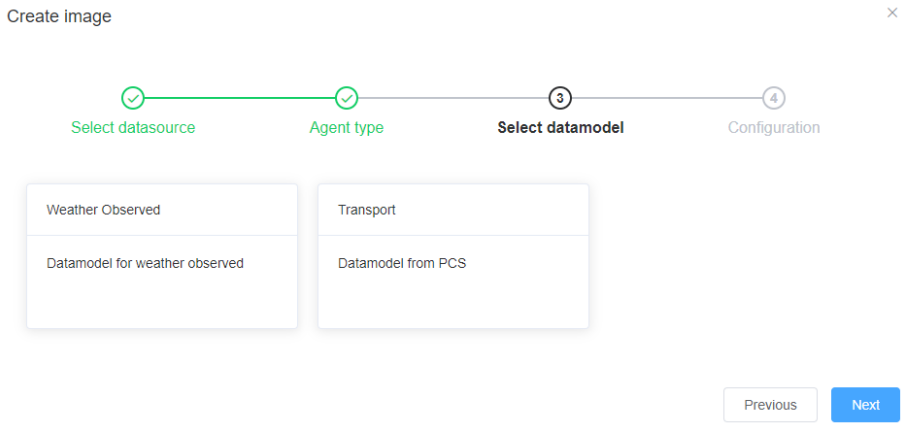


Figura 8.15: Asistente creación agentes de DataPorts (III).

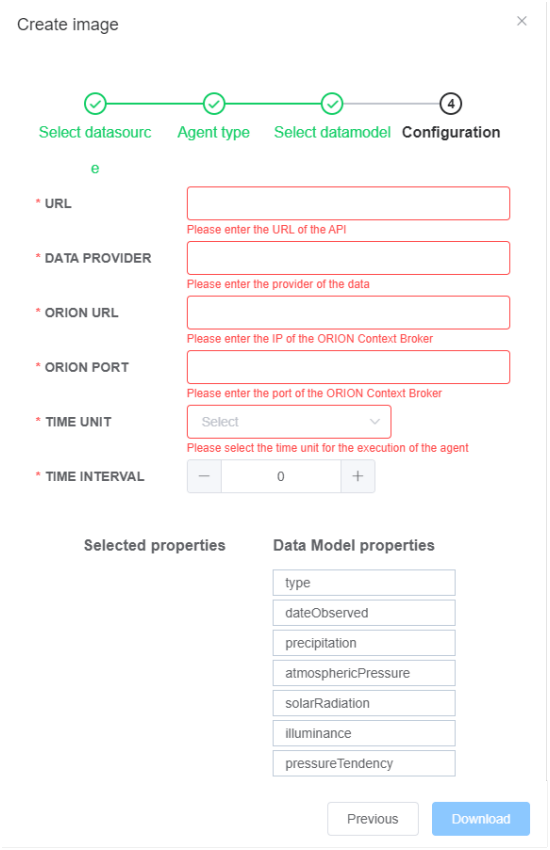


Figura 8.16: Asistente creación agentes de DataPorts (IV).

Figura 8.17: Gestor de agentes de DataPorts.

Modelo de datos

El modelo de datos de DataPorts (Figura 8.18) integra conceptos de ontologías y modelos de datos existentes, incluidos los modelos de datos inteligentes de FIWARE, el modelo de datos UN / CEFACT, la ontología SAREF y el modelo de información IDSA. La versión inicial del modelo de datos se alojará en un repositorio de Git, que contendrá los documentos de esquema JSON correspondientes que describen la sintaxis en NGSI v2, así como la documentación y los ejemplos. Para cada clase del modelo de datos, se proporcionarán ejemplos normalizados (NGSI v2) y simplificados (JSON).

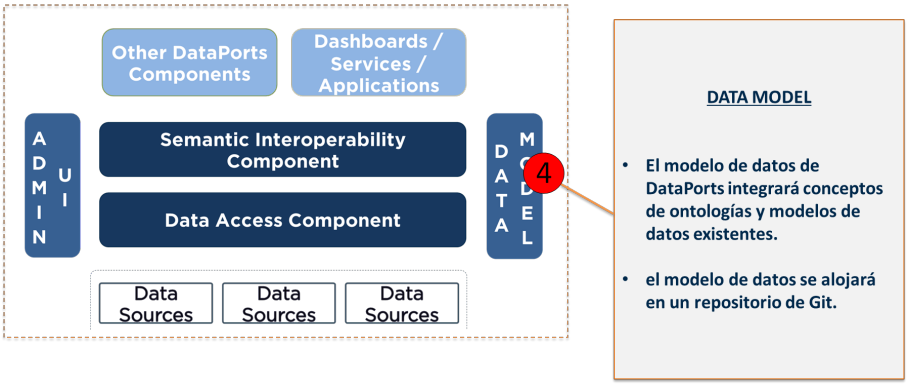


Figura 8.18: Escenarios de validación (IV).

Los conceptos definidos en el Modelo de Datos están organizados por dominio y tema, y esta jerarquía se ve reflejada en la organización del repositorio del Modelo de Datos (Figura 8.19). El repositorio Git tiene una estructura de árbol, en la que la raíz (repositorio Umbrella) proporciona acceso a los repositorios de los diferentes dominios y contiene información general y pautas, así como definiciones y recursos comunes. Cada dominio proporciona acceso a sus diferentes temas y contiene información y recursos compartidos del dominio. De manera similar, cada tema contiene carpetas para cada uno de sus tipos de entidad, así como información y recursos compartidos del tema. Finalmente, se proporcionan las especificaciones, ejemplos e información de cada tipo de entidad en su carpeta correspondiente. Esta estructura ha sido diseñada con el objetivo de facilitar la futura integración del DataPorts Data Model con la iniciativa Smart Data Models, que es una iniciativa relacionada con FIWARE para la publicación de los modelos que describen los datos que los participantes de los espacios de datos pueden intercambiar. Finalmente, cada tipo de enti-

dad definido en el Modelo de Datos pasará por dos estados: incubado (es decir, con especificaciones existentes pero no oficialmente adoptadas por los pilotos) y adoptado (definiciones de entidad consolidadas que se utilizan activamente en DataPorts).

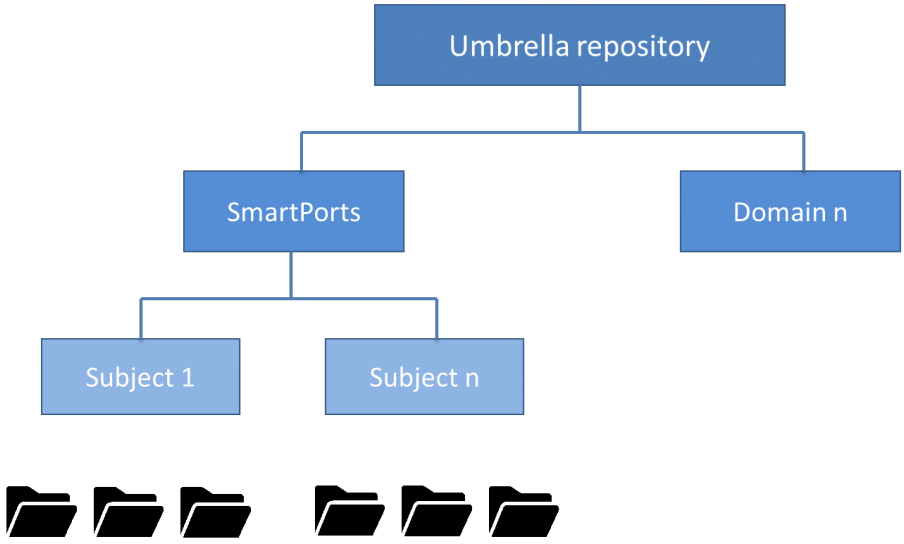


Figura 8.19: Organización del repositorio del modelo de datos.

Registro de Metadatos

Con respecto al Registro de Metadatos (Figura 8.20), los metadatos se almacenan en una base de datos separada en MongoDB, denominada "metadatos", que se corresponde con el valor del Servicio Fiware "metadatos". El Modelo de Datos (Figura 8.21) de dicho registro incluye tres tipos de datos: "DataSource", "AgentImage" (para los metadatos de los agentes bajo demanda) y "AgentContainer" (para los metadatos de los agentes de publicación / suscripción). La Figura 8.22 ofrece una captura de pantalla de cómo se ve dicho registro de metadatos en la aplicación gráfica.

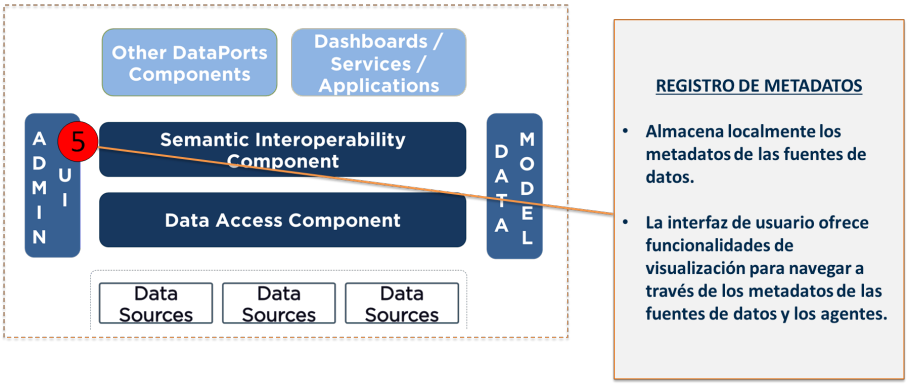


Figura 8.20: Escenarios de validación (V).

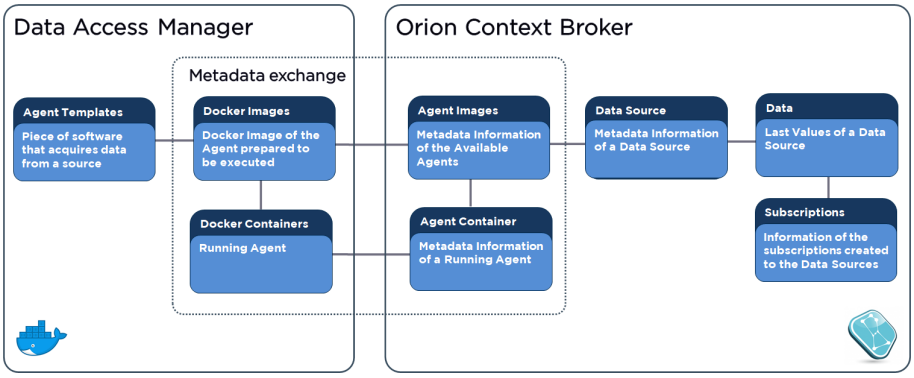


Figura 8.21: Intercambio de metadatos entre agentes Docker y Orion Context Broker.

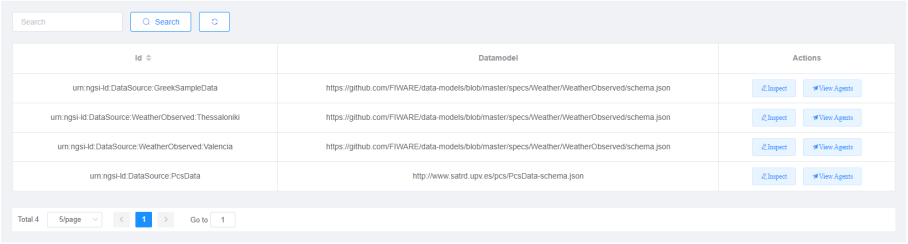


Figura 8.22: Interfaz gráfica del registro de metadatos.

API de Interoperabilidad Middleware

Los principales métodos que proporcionará la API de interoperabilidad semántica (Figura 8.23) son los siguientes:

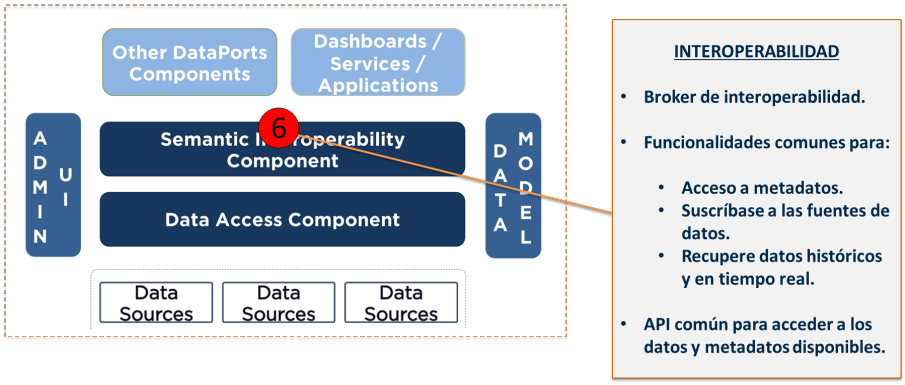


Figura 8.23: Escenarios de validación (VI).

- Relacionados con el registro de metadatos:
  - Crear / eliminar / actualizar la fuente de datos.
  - Obtener información de la fuente de datos.
  - Enumerar las fuentes de datos / Consultar por identificación de la fuente de datos / Consultar por metadatos.
  - Obtener descripción de los agentes.
  - Recuperar atributos de fuente de datos.
  - Lista de tipos de datos.
- Relacionados con el componente Publicar-Suscribir:
  - Crear / eliminar / actualizar suscripción.
  - Lista de suscripciones.
  - Recuperar información de suscripción.
- Relacionados con el componente bajo demanda:
  - Obtener datos históricos.
  - Obtener los últimos valores.

Consumo de la API por otros componentes de la plataforma

La Figura 8.24 muestra como la API del componente de interoperabilidad se relaciona con otros componentes de la plataforma.

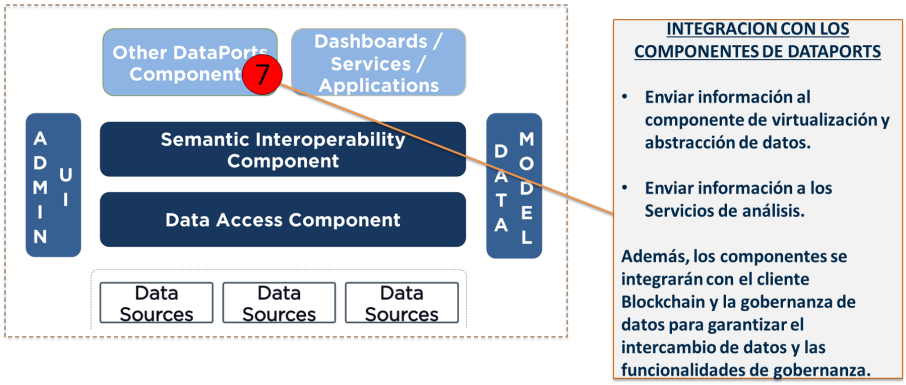


Figura 8.24: Escenarios de validación (VII).

Consumo de la API por servicios y aplicaciones

La Figura 8.25 muestra cómo esta API del componente de interoperabilidad también puede ser consumida por otras aplicaciones o servicios nuevos o ya existentes.

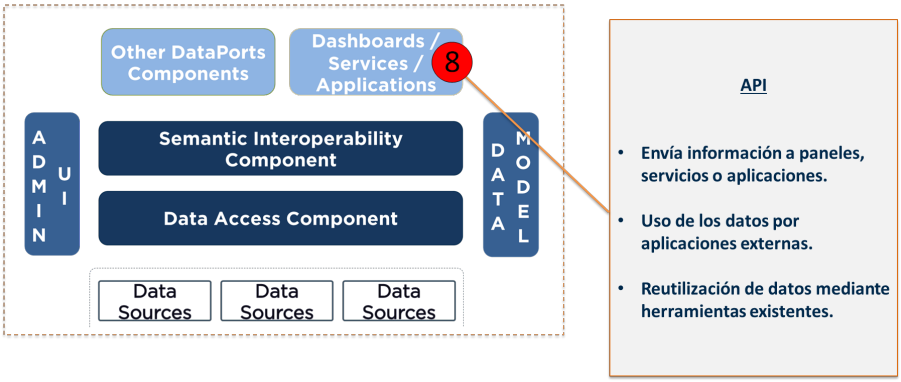


Figura 8.25: Escenarios de validación (VIII).

Se adjuntan diferentes capturas de pantalla de aplicaciones desarrolladas utilizando la API común de interoperabilidad:

- Aplicación que recibe notificaciones, a las cuales se ha suscrito (Figura 8.26).
- Aplicación para obtener datos de contexto o actuales (Figura 8.27).
- Aplicación para realizar consultas de datos históricos (Figura 8.28).
- También es posible reutilizar el API en aplicaciones de composición de servicios como Wirecloud (Figura 8.29).

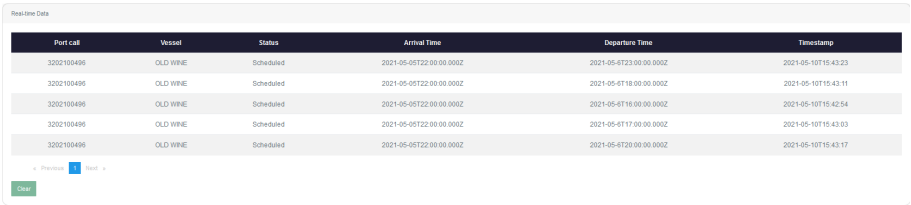


Figura 8.26: Captura de pantalla de aplicación receptora de notificaciones.

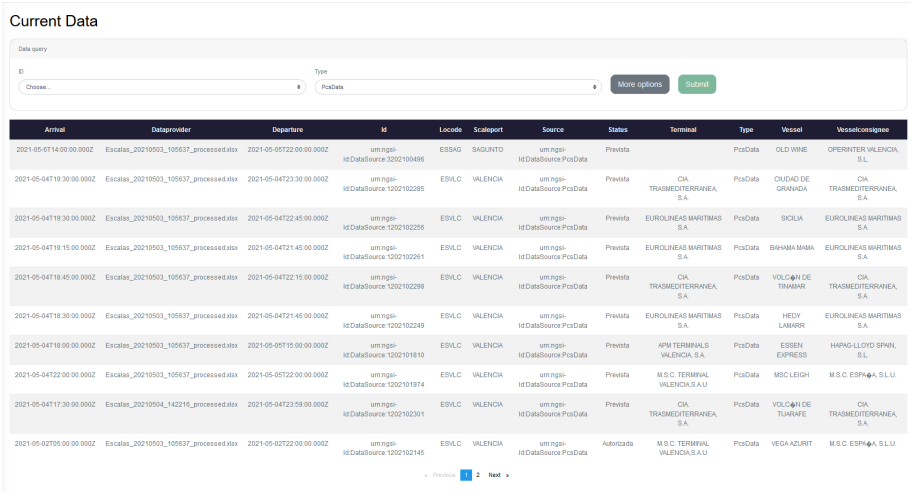


Figura 8.27: Captura de pantalla de aplicación petición a datos contextuales.

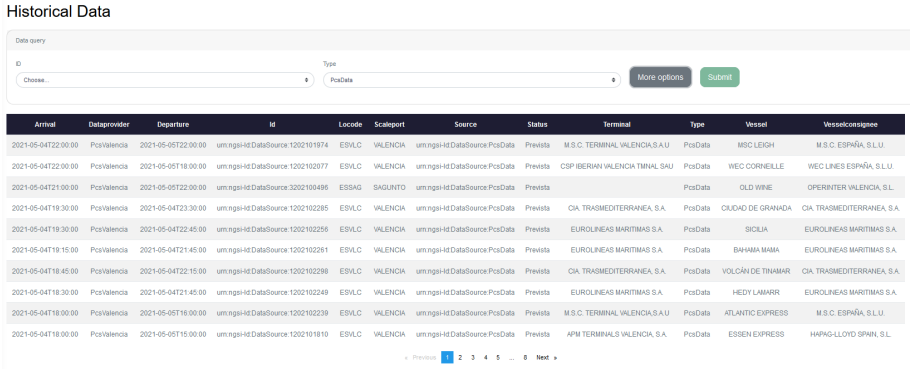


Figura 8.28: Captura de pantalla de aplicación petición a datos historicos.

Figura 8.29: Captura de pantalla de integración con Wirecloud.

Del mismo modo, otras herramientas existentes que admiten NGSI, como Node-RED, CKAN, Kibana o Apache NiFi, podrían hacer uso de la API de interoperabilidad semántica para recuperar datos de las fuentes de datos, lo que permite una forma sencilla de mostrar los datos y desarrollar nuevos servicios.

8.6. Alcance de la solución dentro del proyecto

La solución y arquitecturas propuesta en esta Tesis Doctoral tienen el siguiente alcance y relación con DataPorts:

- Interoperabilidad Middleware
  - Despliegue y configuración de plataformas IoT Middleware y fuentes de datos heterogéneas. Concretamente, las fuentes de datos del entorno portuario y de los casos de uso globales.
  - Implementación de nuevos mecanismos para extender la solución Middleware existente, mediante el desarrollo de la capa de acceso a datos, con los correspondientes agentes y mecanismos compatibles con el ecosistema Fiware. Utilización y extensión de las capacidades del Orion Context Broker como núcleo de interoperabilidad del proyecto.
  - Diseño e implementación de la solución y validación en casos de uso. Se traduce en ser un mecanismo habilitador de soluciones Big Data y Cognitivas, encuadrar la solución en un marco de seguridad y privacidad, integrarla en un entorno de soberanía de datos y hacerla compatible con soluciones Blockchain.
- Interoperabilidad de Aplicaciones y Servicios
  - Despliegue y configuración de servicios y aplicaciones de plataformas IoT. Se extendieron las capacidades mediante el desarrollo de agentes bajo demanda y un componente que proporcionaba una API para su manejo.
  - Diseño e implementación de la solución y validación en casos de uso. Los agentes bajo demanda alimentan los sistemas de Abstracción y Virtualización de contenedores de Datos, lo que requiere utilizar mecanismos para asegurar la transmisión y persistencia de estos datos, como Apache Flume.
- Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos
  - Despliegue y configuración de servicios y aplicaciones de plataformas IoT. Se aprovechan las funcionalidades desarrolladas para i) desarrollar nuevos nodos y funcionalidades, ii) crear nuevos flujos de interoperabilidad y iii) probar nuevas herramientas de Mashup, como Wirecloud.

- Validación de la solución en casos de uso como las funciones de valor añadido desarrolladas sobre la plataforma que permite integrar los datos traducidos al formato común a soluciones existentes.

Finalmente, es interesante indicar que el trabajo realizado se plasmó en varias presentaciones en eventos:

- Reunión del Grupo de Actividad 44 [210] de la BDVA/DAIRO [211] durante abril de 2021: Los principales temas tratados fueron Datos/IA, estándares e interoperabilidad, Espacios de Datos y Fiabilidad de la IA Industrial. Se realizó una presentación sobre la solución de interoperabilidad de la presente Tesis Doctoral, principalmente, sobre los detalles de la elaboración del registro de metadatos.

- Grupo de Sinergia de Espacios de Datos en la reunión inicial durante mayo de 2021 [212]: La reunión inicial del Grupo de Sinergia de Espacios de Datos tenía como objetivo iniciar la colaboración entre organizaciones y proyectos que comparten la visión descrita anteriormente sobre cómo materializar una infraestructura blanda basada en estándares abiertos, disponible en código abierto y compatible con el CEF para la creación de espacios de datos en Europa. Se presentó la solución desde el punto de vista de la interacción de la solución propuesta con los habilitadores de la plataforma Fiware.

- Tercer seminario web del proyecto PIXEL durante junio de 2021 [213]: Se participó en un webinar organizado por el Proyecto PIXEL centrado principalmente en describir la Plataforma PIXEL en detallle. Se presentó la solución propuesta para DataPorts y sus sinergias con Pixel en un espacio de este webinar llamado: "Fertilización cruzada entre proyectos de investigación: sinergias con DataPorts y el uso de FIWARE en las iniciativas de innovación de los puertos marítimos".

- Simposio Internacional sobre Computación Distribuida Inteligente (IDC) durante septiembre de 2021 [214]: Se presentó un trabajo realizado en colaboración con el proyecto PIXEL, que propuso una solución basada en el desarrollo realizado en PIXEL y DataPorts, para calcular, de forma colaborativa, y compartir indicadores cuantitativos compuestos entre actores industriales asegurando la soberanía de los datos.

- Foro Europeo de Valor de Big Data (EBDVF 2021) durante noviembre de 2021 [215]: El evento reúne a profesionales de la industria, desarrolladores de negocios, investigadores y responsables políticos de toda Europa y otras regiones del mundo para avanzar en las acciones políticas y en

las actividades industriales y de investigación en las áreas de Datos e IA. Se realizó una demostración del funcionamiento del componente de interoperabilidad de la presente Tesis Doctoral.

Además, hasta el momento, ha dado lugar a tres publicaciones. Las dos primeras en colaboración con PIXEL y la tercera exclusiva de DataPorts:

- *Marco y metodología para el establecimiento de políticas de ciudad puerto basadas en indicadores compuestos en tiempo real e IoT, un caso de uso práctico* [216]

  Este articulo propone un marco de software basado en IoT, acompañado de una metodología para definir, calcular y predecir indicadores compuestos que representan fenómenos del mundo real en el contexto de una ciudad portuaria inteligente. El experimento consiste en implementar un índice compuesto para monitorizar la congestión del tráfico en la interfaz puerto-ciudad en Salónica (Grecia). Esta investigación involucra al puerto de Thessaloniki, responsable de uno de los casos de uso de DataPorts. El documento se centra en la interoperabilidad de la ciudad y el puerto de Salónica en el tráfico de mercancías. El objetivo es optimizar el tráfico entre la ciudad y la zona portuaria, además de reducir la contaminación, las colas, el consumo de energía, etc., y finalmente actuar, como un único punto de referencia para la explotación de datos en beneficio de la Autoridad Portuaria, los pasajeros y la comunidad local. La solución propuesta ayuda a las autoridades a tener una mejor planificación y optimización de recursos, mientras que los pasajeros tendrán un traslado más cómodo y fluido desde su hogar hasta su embarque. Desde un punto de vista técnico, la interoperabilidad se implementa mediante el uso de Fiware Data Models, agentes NGSI, Orion Context Broker y componentes Fiware.

- *Un enfoque novedoso para calcular indicadores compuestos en tiempo real basándose en Internet de las cosas y los espacios de datos industriales* (Publicación aceptada y presentada en el congreso 14th International Symposium on Intelligent Distributed Computing (IDC 2021). Actualmente pendiente de ser publicada)

  Este articulo propone una solución partiendo de la solución de interoperabilidad propuesta como marco habilitador de la interoperabilidad en IoT para calcular de manera colaborativa y compartir indicadores compuestos cuantitativos entre los actores industriales mientras se asegura la soberanía de los datos y se alinea con la arquitectura propuesta por IDS.

- *Hacia los puertos cognitivos del futuro* (Capítulo de libro aceptado y pendiente de ser publicado en un libro titulado Tecnologías y aplicaciones para el valor de Big Data (TABDV 2021))

  Este capítulo del libro aborda las características de escalabilidad, interoperabilidad y estandarización de las plataformas de datos desde el punto de vista empresarial en un caso práctico de puerto inteligente. Describe cómo DataPorts intenta abordar estos retos y proporcionar un ecosistema en el que las autoridades portuarias y las partes interesadas externas, como las empresas de transporte y logística, puedan cooperar y crear la base para ofrecer servicios cognitivos a los usuarios finales.

Se espera durante el ultimo año del proyecto, obtener un mayor número de publicaciones en revistas y participar en nuevos eventos y congresos.

Capítulo 9

Conclusiones y líneas futuras de investigación

En el trabajo presentado en esta Tesis Doctoral se ha intentado revertir la problemática asociada a la heterogeneidad de las plataformas IoT y la falta de un estándar de interoperabilidad predominante en el mercado. Por tanto, la Tesis Doctoral se enfoca como una gran oportunidad que parte de todas las diferentes ventajas que ofrecen las plataformas, aplicaciones y servicios IoT disponibles, para ofrecer una serie de mecanismos de interoperabilidad y un marco común que permita poder acceder, interactuar e intercambiar información y funcionalidades en las diferentes plataformas IoT. Partiendo de dicha situación, la presente Tesis Doctoral ha cubierto las necesidades de interoperabilidad de plataformas IoT en las capas de Middleware y Aplicación y Servicios mediante el desarrollo e implementación de diferentes mecanismos de interoperabilidad localizados en dichas capas.

9.1. Conclusiones generales

Se proporcionan, en primer lugar, las conclusiones obtenidas del análisis del estado del arte realizado en la presente Tesis Doctoral. En segundo lugar, se recapitulan los elementos más destacados y ventajas de la solución propuesta. En tercer lugar, se explican los detalles principales del diseño y la implementación de la solución propuesta. En cuarto lugar, se resumen las lecciones aprendidas de instanciar dichas soluciones en los diferentes casos de uso. Finalmente, se finaliza con las conclusiones clave a modo de recapitulación final.

Estado del Arte:

Las principales conclusiones obtenidas basadas en el análisis realizado del estado del arte son las siguientes:

- Las Plataformas IoT proporcionan funcionalidades para interactuar con las pasarelas, dispositivos y sensores que tienen conectados a ellas. Las plataformas ofrecen las herramientas para garantizar el acceso a todos estos elementos. Además, las plataformas proporcionan aplicaciones y servicios para procesar los datos obtenidos y ofrecer un valor añadido a estos.

- Debido a la gran variedad y cantidad de soluciones IoT que existen en el mercado, que han crecido sin seguir un estándar claro, han surgido barreras que impiden la interoperabilidad entre las distintas soluciones existentes. Esta heterogeneidad produce un obstáculo y desconfianza en los usuarios en la implantación y despliegue de estas tecnologías en sus organizaciones.

- Las soluciones IoT analizadas son diversas, desde el punto de vista del despliegue pueden ser soluciones software disponibles para descargar y desplegar en local o soluciones software desplegadas en una nube pública o privada. Estas soluciones pueden clasificarse como soluciones a medida del usuario, soluciones propietarias, proyectos de código abierto mantenidos por una comunidad, desarrolladas por entidades privadas con diferentes versiones (gratuita, con menos funcionalidades, o de pago, que incluye un mayor numero de funcionalidades dependiendo del plan contratado), financiadas por entidades publicas, resultados o derivadas de los proyectos de investigación.

- El despliegue de cada plataforma es un aspecto importante a considerar, puede ser un proceso casi automático, o, por el contrario, en otros casos puede implicar seguir diferentes pasos y una configuración más compleja. Los recursos solicitados por la plataforma deben adaptarse a las necesidades del usuario. En algunos casos, un despliegue en local dentro de la organización puede ser suficiente. No obstante, en otros casos el despliegue en la nube proporciona a los usuarios una forma eficiente de desplegar sus soluciones, ofreciendo una alta escalabilidad y reducción de costes. Las Plataformas IoT utilizan esta tecnología permitiendo que sus recursos estén disponibles en una nube privada o pública.

- Cada plataforma IoT del mercado ha seguido su propio enfoque particular. En el estado del arte se presenta una muestra representativa de las

diferentes Plataformas IoT, resaltando la heterogeneidad entre las soluciones, con el fin de encontrar las funcionalidades básicas que debe tener una plataforma que quiere ofrecer interoperabilidad entre ellas.

- Existen diferentes clasificaciones de los distintos aspectos de la interoperabilidad, también llamadas niveles de interoperabilidad y diferentes arquitecturas de referencia. Por tanto, la primera tarea necesaria para desarrollar una solución como la que se presenta en esta tesis doctoral ha sido partir de un análisis del material existente en la literatura. para acotar y clasificar los elementos que se deben abordar en la solución de interoperabilidad.

- Las soluciones disponibles para lograr la interoperabilidad entre plataformas IoT, también, son variadas y tienen definidas diferentes aproximaciones para lograrlo. Principalmente, cada una de ellas esta centrada en una serie de objetivos concretos relacionados con la interoperabilidad. El estado del arte se centra en analizar las diferentes plataformas, proyectos, organismos de estandarización o asociaciones centradas en tal fin. La finalidad de la presente Tesis Doctoral ha sido seleccionar y seguir las tecnologías y estándares predominantes para garantizar la interoperabilidad.

- La conclusión clave es que, a pesar de existir múltiples plataformas, tecnologías, organismos y estándares es necesaria una solución que pueda ofrecer mecanismos de interoperabilidad entre estos elementos. Esta no debe ser una solución más en esa lista, sino actuar como el nexo que garantice que los usuarios puedan intercambiar la información y gestionar sus plataformas haciendo uso de la solución de interoperabilidad.

Solución propuesta:

Los principales aspectos a destacar de la solución propuesta en la Tesis Doctoral son los siguientes:

- El primer paso necesario para lograr una solución como la propuesta es identificar, desplegar, configurar y realizar un análisis las distintas plataformas IoT disponibles y sus correspondientes aplicaciones/servicios.

- Los mecanismos propuestos en la presente Tesis Doctoral proporcionan una solución de interoperabilidad para las capas middleware y aplicación y servicios de las Plataformas IoT. Cada uno de los mecanismos de interoperabilidad ofrecen una API de acceso a sus funcionalidades.

- Los mecanismos de interoperabilidad de la capa middleware establecen un mecanismos de abstracción para acoplar en él las diferentes plataformas IoT. Esto proporciona las funcionalidades para acceder a las principales funcionalidades e información de las diferentes plataformas (y sus correspondientes dispositivos, sensores o entidades) conectadas a este mecanismo de abstracción.

- Los mecanismos de interoperabilidad de la capa de aplicación y servicio proporcionan una solución para el acceso común y la interacción entre sí de los distintos servicios heterogéneos ofrecidos por las plataformas IoT.

- Además de los mecanismos de interoperabilidad middleware y de aplicación y servicios es necesario considerar aquellos elementos transversales para ofrecer una solución completa. Estos afectan a más de una capa de interoperabilidad y se pueden dividir en tres áreas principales: seguridad, interacciones entre componentes y virtualización de las soluciones.

- El marco común de interoperabilidad proporciona una forma de homogeneizar los diferentes mecanismos de interoperabilidad presentados. Mediante el uso de este marco, cualquier plataforma IoT puede hacerse interoperable con respecto a sus dispositivos, middleware y capa de servicios. Esta herramienta ofrece un marco visual completo para configurar y gestionar de forma segura y desarrollar nuevas aplicaciones de software aprovechando los datos de múltiples plataformas IoT heterogéneas.

- La solución propuesta se compone de los mecanismos de interoperabilidad, componentes habilitantes y el marco común de interoperabilidad. Este ha sido el conjunto de herramientas diseñadas e implementadas para solucionar los problemas de interoperabilidad en las capas de Middleware y Aplicación/Servicios de las plataformas IoT en la presente Tesis Doctoral.

Diseño e implementación:

Las principales conclusiones relacionadas con el diseño y la implementación de la solución propuesta en la presente Tesis Doctoral son los siguientes:

- El objetivo de esta Tesis no era crear una nueva plataforma IoT, sino una estructura de interoperabilidad para interconectar diferentes plataformas IoT, dispositivos, aplicaciones y otros artefactos IoT.

- La arquitectura presentada se fundamenta en la arquitectura global que se ofrece en el caso de uso INTER-IOT: Dicha arquitectura incluye varias

soluciones de interoperabilidad dedicadas a capas específicas Dispositivo a Dispositivo (D2D), Red a Red (N2N), Middleware a Middleware (MW2MW), Aplicación y Servicios a Aplicación y Servicios (AS2AS), Datos y Semántica a Datos y Semántica (DS2DS). Cada capa de infraestructura de interoperabilidad tiene un fuerte acoplamiento con las capas adyacentes y proporciona una interfaz. Las interfaces son controladas por un marco de metaniveles para proporcionar una interoperabilidad global. Se puede acceder a cada mecanismo de interoperabilidad a través de una API. Las capas de la infraestructura de interoperabilidad pueden comunicarse e interoperar a través de las interfaces. Esta estratificación cruzada permite lograr una integración más profunda y completa.

- Partiendo de los fundamentos de INTER-IoT la solución propuesta está enmarcada en las capa de middleware (MW2MW) y la capa de aplicaciones y servicios (AS2AS). Por tanto, en la presente Tesis Doctoral, se han desarrollado una serie de mecanismos que han extendido y ampliado la propuesta genérica tomada como base. El resultando han sido cuatro mecanismos de interoperabilidad:

  - Interoperabilidad Middleware (MW)
  - Interoperabilidad de Plataformas Middleware (MW2MW)
  - Interoperabilidad de Aplicaciones y Servicios (AS2AS)
  - Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos (ASFLOW)

  Todos los mecanismos están Dockerizados y, ofrecen una API de acceso a sus funcionalidades, descrita mediante Swagger.

- En cuanto a la Interoperabilidad Middleware, se apoya en una plataforma middleware de referencia como mecanismo de interoperabilidad. Unos adaptadores llamados agentes son el elemento clave para conectar las diferentes plataformas y dispositivos a este mecanismo. La plataforma IoT Fiware es el principal elemento habilitador de la interoperabilidad de este mecanismo, concretamente su componente Orion Context Broker. Además, se han desarrollo diferentes componentes específicos para complementar las necesidades del mecanismo de interoperabilidad que no son cubiertas por Orion.

- En cuanto a la Interoperabilidad de Plataformas Middleware, se apoya en una capa de abstracción específicamente desarrollada como mecanismo de referencia para proporcionar el componente middleware que abstrae las funcionalidades de interoperabilidad. Unos adaptadores llamados

puentes son el elemento clave para conectar las diferentes plataformas y dispositivos a este mecanismo. Se han implementado los componentes del mecanismo de interoperabilidad, la interfaz API REST que expone las funcionalidades y los puntos de acceso a las plataformas, mediante piezas de software en el lenguaje de programación Java. Además, se hace uso de un broker para la gestión de mensajes llamado RabbitMQ y una base de datos semántica llamada Parliament Triple Store.

- En cuanto a la Interoperabilidad de Aplicaciones y Servicios, se Ofrece una API común para definir las llamadas a los servicios y aplicaciones que existen en las plataformas conectadas a este mecanismo. Hay diferentes instanciaciones de esta solución que hacen uso, como elemento central, de MW, MW2MW o ASFLOW. Por tanto, según la instanciación elegida los agentes, puentes o nodos son los elementos clave para conectar con los servicios y aplicaciones accedidos de las plataformas IoT. Esta solución necesita partir del resto de mecanismos (Fiware, solución desarrollada en MW2MW con RabbitMQ o Node-RED), y utilizar sus respectivos registros (MongoDB, Parliament o Node-RED) y conectores (agentes, puentes o nodos). Finalmente, se ha implementado específicamente (en Java o Node.js según el caso) el software necesario para actuar como gestor de servicios y peticiones.

- En cuanto a la Interoperabilidad de Aplicaciones y Servicios mediante Programación orientada a Flujos, se fundamenta en un motor gráfico de diseño y ejecución de flujos de servicios conectados. Se ha seguido el paradigma llamado programación orientada a flujos. Para acceder a los servicios y aplicaciones, se utilizan unos adaptadores llamados nodos. La conexión entre sí de estos nodos define los flujos de ejecución entre los servicios. Los nodos, además, facilitan la descripción de las funcionalidades de los servicios y aplicaciones accedidos. Se ha seleccionado Node-RED como motor de composición de servicios heterogéneos de plataformas IoT siguiendo el patrón de programación orientada a flujos.

- Los mecanismos diseñados e implementados pueden interactuar con el mediador semántico ofrecido en el proyecto INTER-IoT para obtener facilidades de interoperabilidad semántica.

- La solución propuesta cubre, mediante los componentes habilitantes, los aspectos no funcionales, como la confianza, seguridad, privacidad, extensibilidad o escalabilidad. La implementación propuesta se ha centrado en los componentes que ofrece la plataforma WSO2 relacionadas con la gestión de identidades, para tratar con los diferentes roles de los usuarios,

y la centralización y gestión de las APIs ofrecidas por cada uno de los componentes. Se ha seleccionado Docker como mecanismo para el uso de contenedores para envolver componentes y servicios.

- Se ha diseñado e implementado el marco común de interoperabilidad, que tiene como objetivo proporcionar mecanismos, herramientas y contenidos de ayuda para hacer un uso adecuado de los mecanismos de interoperabilidad y las APIs que estos ofrecen. Este elemento se ha desarrollado mediante una aplicación web especifica que implementa la interfaz gráfica mediante Vue.js y el backend en Node.js, utilizando MongoDB como base de datos no relacional.
- La solución esta diseñada e implementada de manera que se puede extender mediante el desarrollo de nuevos conectores (agentes, puentes o nodos) a nuevas plataformas y servicios, permitiendo una compatibilidad mayor con las soluciones existentes y futuras.

Casos de uso:

Los instanciación de la solución propuesta en diferentes casos de uso ha dado lugar a una serie de lecciones aprendidas y conclusiones como las siguientes:

- La Unión Europea ha respaldado el ecosistema y la comunidad IoT mediante diferentes acciones de investigación e innovación durante los últimos años. Los proyectos de investigación son una pieza fundamental para dar soporte a esta tesis doctoral. Los proyectos europeos H2020 INTER-IoT, ACTIVAGE, PIXEL y DataPorts han sido los que han servido para validar los mecanismos de interoperabilidad desarrollados en esta Tesis Doctoral.
- El proyecto INTER-IoT se centró en diseñar, implementar y experimentar una solución basada en un marco de desarrollo abierto dividido en múltiples capas (dispositivos, red, middleware, aplicación y servicios y datos y semántica), una metodología asociada y herramientas para permitir la interoperabilidad entre plataformas heterogéneas IoT. La presente Tesis Doctoral ha tomado como referencia la arquitectura fundamental de dicho proyecto y se ha centrado en las capas de middleware y aplicación y servicios. Este proyecto ha sido clave para analizar las plataformas y servicios IoT disponibles y para especificar el marco teórico, diseñar la arquitectura e implementar la solución software de la Tesis Doctoral. Lo más destacable de la interacción con el proyecto ha sido lograr la construcción de la solución propuesta, sentando así las bases de los trabajos desarrollados en los siguientes casos de uso.

- En el proyecto ACTIVAGE se construyó un ecosistema europeo de IoT a través de 9 ubicaciones de despliegue localizadas en siete países europeos. Desde el punto de vista técnico se reutilizaron, interconectaron y ampliaron las plataformas, tecnologías y estándares de IoT abiertos y propietarios disponibles. Además, se integraron nuevas interfaces necesarias para proporcionar interoperabilidad a través de estas plataformas heterogéneas. Este proyecto parte de los componentes implementados y se centra en su integración y despliegue en diferentes ubicaciones cada una de ellas con diferentes plataformas y servicios. ACTIVAGE fue clave para consolidar la solución desarrollada en esta Tesis Doctoral y conocer las necesidades técnicas que surgen en un escenario tan amplio. Entre ellos, cabe destacar los diferentes despliegues de plataformas, las necesidades especificas de rendimiento, la integración con el soluciones externas o considerar los recursos disponibles. Además, también permitió conocer las necesidades desde un punto de vista del dominio de aplicación de la salud, en el que, por ejemplo, la privacidad y seguridad de los datos es un hecho fundamental.

- El proyecto PIXEL tenía como objetivo facilitar las operaciones en los puertos y cubrir varios desafíos de los puertos del futuro, como, por ejemplo, el de reducir las emisiones contaminantes. Para tal fin, en dicho proyecto fue necesario recopilar e intercambiar datos heterogéneos, incluidos los sensores de IoT, entre las distintas partes interesadas que operan en los puertos. En este proyecto quedó claro que estos datos son difíciles de manejar debido a la falta de homogeneización y de herramientas para operarlos. En este caso se desarrolló una solución middleware que partía de una plataforma IoT ya existente como núcleo central de la interoperabilidad middleware. Esto se trata de una decisión de diseño después de analizar las necesidades específicas de los participantes en el proyecto. Esta decisión hace que este mecanismo de interoperabilidad sólo pueda cubrir nativamente las funcionalidades de la plataforma seleccionada como mecanismo de abstracción. Por un lado, esto proporciona una solución más sencilla y manejable en aquellos casos en los que se intenta conectar plataformas sencillas o servicios que simplemente ofrecen fuentes de datos heterogéneos, Por otro lado, puede que no sea eficiente cuando se intenta conectar a la capa de abstracción complejas plataformas IoT, que tienen diversas funcionalidades o complejas descripciones en los datos. Esto último no se correspondía con la situación de este caso de uso. Por tanto, la principal lección aprendida fue que los mecanismos de interoperabilidad se deben adaptar a las necesidades técnicas reales de sus

potenciales usuarios. Esto garantiza el éxito en su integración, despliegue y futuro uso.

- El proyecto DataPorts, que aún no finalizado, tiene como objetivo proporcionar una plataforma de datos en la que las empresas de transporte y logística de un puerto marítimo puedan gestionar los datos como cualquier otro activo de la empresa, con el fin de crear la base para ofrecer servicios cognitivos. El proyecto tiene un marcado interés en la creación de una plataforma de datos segura que permita compartir la información no sólo entre agentes portuarios sino también entre otros puertos. Ello implica implementar un entorno seguro de intercambio de datos de manera fiable, con permisos de acceso y contratos. Este proyecto ha tomado como referencia los mecanismo realizados en el caso de uso de PIXEL y los ha extendido para cubrir nuevas funcionalidades que no eran nativas, como el acceso a datos historicos. Ha puesto también énfasis en realizar un marco de interoperabilidad y herramientas de desarrollo que acerquen la solución a los usuarios menos técnicos. En este proyecto los mecanismos de interoperabilidad desarrollados son el eje central de la solución, es decir, los elementos principales sin los cuales la solución no puede funcionar. Dichos mecanismos están encargados de proporcionar los datos y metadatos que consumirán el resto de componentes de la plataforma. Estos pueden ser, por un lado, aplicaciones o servicios consuman los datos y proporcionen soluciones de valor añadido, Por otro lado, los consumidores pueden ser componentes tecnologicos más complejos, debido a que la solución propuesta se ha enfrentado a una integración con soluciones de virtualización y abstracción de datos, tecnologías de gobernanza de datos, soluciones de IA y cognitivas, tecnologías Blockchain y una arquitectura relacionada con los espacios de datos. Es un proyecto clave porque muestra claramente que es posible partir de los logros alcanzados en la presente Tesis Doctoral y utilizar la solución propuesta en presentes y futuras lineas de investigación tecnológicas.

Conclusiones clave:

Finalmente, los principales aspectos clave a destacar de la Tesis Doctoral, como recapitulación final de las conclusiones generales, son los siguientes:

- En la actualidad hay una apuesta firme en los proyectos de investigación y en el mercado por las soluciones IoT. En los últimos años la interoperabilidad entre fuentes de datos, plataformas, aplicaciones y servicios ha pasado de ser una finalidad, a ser un elemento necesario y habilitador en la construcción de nuevas soluciones software y proyectos. Por tanto,

los resultados de esta Tesis Doctoral son una base solida sobre la que construir futuros proyectos que utilicen nuevas tecnologías y necesiten abstraer el acceso a los datos de las plataformas y servicios.

- La interoperabilidad es la pieza clave para crear un ecosistema activo alrededor de los datos, plataformas, servicios y aplicaciones IoT. Un mayor grado de interoperabilidad implica un mayor reaprovechamiento de toda la información y funcionalidad disponible. Por tanto, ofrecer una solución de interoperabilidad eficaz y eficiente conduce a un mayor grado de satisfacción e interés de los potenciales usuarios en usarla y construir nuevas soluciones a partir de ella.

- Los mecanismos habilitadores transversales son también clave para transmitir confianza a los usuarios. Por un lado, la seguridad y privacidad de la información son tan importantes para las empresas como el propio funcionamiento de la solución en sí. Si el propietario de los datos no tiene confianza en los mecanismos de interoperabilidad, la solución no tendra exito. Por otro lado, la virtualización de los componentes facilitará el uso y despliegue de los mismos. Cuando mas manejable y sencilla sea la solución software mejor aceptación tendrá entre los potenciales usuarios.

- Los marcos de desarrollo son necesarios para atraer a una audiencia más allá de los desarrolladores técnicos. Ofrecer un robusto entorno en el que manejar los componentes facilita la presentación de los componentes, el desarrollo de los conectores, la integración de la seguridad y la utilización y gestión de los componentes.

- La capacidad de extender la solución integrando nuevas plataformas o tecnologías que vayan apareciendo es crucial para que las soluciones ofrecidas por la presente Tesis Doctoral se puedan adaptar a los nuevos retos y desafíos que surjan en el futuro.

9.2. Futuras líneas de investigación

Las lineas futuras de trabajo pueden seguir diferentes alternativas, como, por ejemplo, las siguientes:

- Soluciones que parten del objeto de la presente investigación. El objetivo es que partiendo de los mecanismos de interoperabilidad ofrecidos por esta Tesis Doctoral sean capaces de proporcionar soluciones nuevas e innovadoras. Por ejemplo, pueden estar centradas en:

  - Aumentar el porfolio de plataformas y servicios compatibles con los mecanismos de interoperabilidad de la solución desarrollada.
  - Ofrecer nuevos mecanismos que faciliten el desarrollo de agentes, puentes o nodos para transformar la información y acceder a los servicios de las plataformas y servicios.
  - Soluciones o proyectos que tomen los mecanismos de interoperabilidad del presente trabajo como sus elementos centrales y habilitadores de interoperabilidad entre sus fuentes de datos, plataformas, aplicaciones y servicios heterogéneos.
  - Diferentes implementaciones de los mecanismos y habilitadores propuestos con otras tecnologías existentes o futuras.
  - Diseño, extensión o implementación de nuevas interfaces gráficas, herramientas de gestión o herramientas para el desarrollo, para satisfacer nuevas funcionalidades o mejorar la usabilidad de los mecanismos y habilitadores propuestos.

- Integración de los mecanismos propuestos con las nuevas tecnologías. De esta manera se puede ofrecer un valor añadido a los datos adquiridos con la solución propuesta en la presente Tesis Doctoral. Esto es fundamental para que la solución se adapte a las nuevas tendencias y se siga considerando su utilización en el futuro. Se consideran tecnologías como:

  - Cloud computing.
  - Soluciones edge.
  - Espacios de datos.
  - Big Data, IA y Servicios Cognitivos.
  - NG-IoT.
  - Blockchain.
  - Gemelo digital.

- Finalmente, a pesar de que la solución se haya validado en los entornos de la salud y de los puertos marítimos, en su concepción fue básicamente creada para ser compatible en cualquier dominio de aplicación. Actualmente, se considera enfocar el trabajo en los siguientes dominios:

  - Dominio académico y teórico.
  - Proyectos de investigación regionales, nacionales e internacionales.
  - Iniciativas de estandarización y asociaciones.

- Consolidar la solución en los dominios de aplicación en los que se ha validado la solución, como el entorno médico y portuario. Además de apoyar a las soluciones enfocadas en reducir las emisiones y mejorar el medio ambiente.
- Analizar el valor que ofrecen los datos proporcionados por las ciudades inteligentes, para implantar la solución propuesta en el dominio de las Smart Cities. Además, analizar y potenciar la interoperabilidad entre las aplicaciones y plataformas localizadas en diferentes ciudades.

Referencias

[1] Alex Gluhak *et al.*, "Report on IoT platform activities," oct 2016. [Online]. Available: http://www.internet-of-things-research.eu/pdf/D03_01_WP03_H2020_UNIFY-IoT_Final.pdf

[2] R. van Kranenburg and A. Bassi, "IoT Challenges," *Communications in Mobile Computing 2012 1:1*, vol. 1, no. 1, pp. 1–5, nov 2012. [Online]. Available: https://link.springer.com/articles/10.1186/2192-1121-1-9https://link.springer.com/article/10.1186/2192-1121-1-9

[3] M. U. Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A review on internet of things (iot)," *International journal of computer applications*, vol. 113, no. 1, pp. 1–7, 2015.

[4] A. Čolaković and M. Hadžialić, "Internet of things (iot): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, 2018.

[5] S. Madakam *et al.*, "Internet of Things (IoT): A Literature Review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, may 2015. [Online]. Available: http://www.scirp.org/Html/56616_56616.htmhttp://www.scirp.org/Journal/Paperabs.aspx?paperid=56616

[6] T. Joseph *et al.*, "IoT middleware for smart city: (An integrated and centrally managed IoT middleware for smart city)," *TENSYMP 2017 - IEEE International Symposium on Technologies for Smart Cities*, oct 2017.

[7] C. Srinivasan, B. Rajesh, P. Saikalyan, K. Premsagar, and E. S. Yadav, "A review on the different types of internet of things (iot)," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 11, no. 1, pp. 154–158, 2019.

[8] G. Gardašević *et al.*, "The iot architectural framework, design issues and application domains," *Wireless personal communications*, vol. 92, no. 1, pp. 127–148, 2017.

[9] S. Haller, A. Serbanati, M. Bauer, and F. Carrez, "A domain model for the internet of things," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 411–417.

[10] J. ho Park *et al.*, "CIoT-Net: a scalable cognitive IoT based smart city network architecture," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, pp. 1–20, dec 2019. [Online]. Available: https://hcis-journal.springeropen.com/articles/10.1186/s13673-019-0190-9

[11] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and Open Challenges," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 796–809, jun 2019. [Online]. Available: https://link.springer.com/article/10.1007/s11036-018-1089-9

[12] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 414–454, mar 2014.

[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, oct 2015.

[14] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the Internet of Things," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2015-October, oct 2015.

[15] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1294–1312, jul 2015.

[16] V. Gazis *et al.*, "A survey of technologies for the internet of things," *IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference*, pp. 1090–1095, oct 2015.

[17] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Computer Communications*, vol. 89-90, pp. 5–16, sep 2016.

[18] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, feb 2017.

[19] T. Teixeira, S. Hachem, V. Issarny, and N. Georgantas, "Service Oriented Middleware for the Internet of Things: A Perspective," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6994 LNCS, pp. 220–229, oct 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-24755-2_21

[20] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti, and Subhajit Dutta, "Role Of Middleware For Internet Of Things: A Study," *International Journal of Computer Science & Engineering Survey*, vol. 2, no. 3, pp. 94–105, aug 2011.

[21] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Cla, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, feb 2016.

[22] P. Wegner, "Interoperability," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 285–287, mar 1996. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/234313.234424

[23] G. da Silva Serapiao Leal, W. Guedria, and H. Panetto, "Interoperability assessment: A systematic literature review," *Computers in Industry*, vol. 106, pp. 111–132, apr 2019.

[24] "What is Interoperability? Definition and FAQs." [Online]. Available: https://www.omnisci.com/technical-glossary/interoperability

[25] "Y.2060 : Overview of the Internet of things." [Online]. Available: https://www.itu.int/rec/T-REC-Y.2060-201206-I

[26] O. Vermesan, "Advancing IoT Platforms Interoperability," *River Publishers Series in Information Science and Technology*, pp. 1–92, 2018.

[27] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Towards Semantic Interoperability Between Internet of Things Platforms," *Internet of Things*, vol. 0, no. 9783319612997, pp. 103–127, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-61300-0_6

[28] P. Annicchino, A. Brékine, F. M. Facca, A. Heijnen, and F. M. Castro, "Next Generation Internet of Things Deliverable number D3.1," 2018. [Online]. Available: https://www.ngiot.eu/wp-content/uploads/sites/26/2020/09/D3.1.pdf

[29] H. van der Veer and A. Wiles, "Achieving Technical Interoperability-the ETSI Approach," *ETSI White Paper No. 3*, 2008. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/IOPwhitepaperEdition3final.pdf

[30] "CREATE-IoT LSP." [Online]. Available: https://european-iot-pilots.eu/create-iot/

[31] CREATE-IOT, "IoT Assessment of convergence and interoperability in LSP platforms." [Online]. Available: https://european-iot-pilots.eu/wp-content/uploads/2017/10/D06_01_WP06_H2020_CREATE-IoT_Final.pdf

[32] CREATE-IOT, "Recommendations for commonalities and interoperability profiles of IoT platforms." [Online]. Available: https://european-iot-pilots.eu/wp-content/uploads/2018/11/D06_02_WP06_H2020_CREATE-IoT_Final.pdf

[33] CREATE-IOT, "IoT Strategy and coordination plan for IoT interoperability and standard approaches." [Online]. Available: https://european-iot-pilots.eu/wp-content/uploads/2020/06/D06_03_WP06_H2020_CREATE-IoT_Final.pdf

[34] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[35] A. Deohate and D. Rojatkar, "Middleware Challenges and Platform for IoT-A Survey," *Proceedings of the 5th International Conference on Trends in Electronics and Informatics, ICOEI 2021*, pp. 463–467, jun 2021.

[36] W. Wang, A. Tolk, and W. Wang, "The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S," *SpringSim 2009 Proceedings*, 2009.

[37] Idabc, "European Interoperability Framework for Pan-European eGovernment Services," *European Commission*, 2004.

[38] A. f. I. o. T. Innovation, "High Level Architecture (HLA) Release 4.0 AIOTI WG03-IoT Standardisation," 2018. [Online]. Available: https://aioti.eu/wp-content/uploads/2018/06/AIOTI-HLA-R4.0.7.1-Final.pdf

[39] IoT-A, "Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0." [Online]. Available: https://www.researchgate.net/publication/272814818_Internet_of_Things_-_Architecture_IoT-A_Deliverable_D15_-_Final_architectural_reference_model_for_the_IoT_v30

[40] ACTIVAGE, "D3.1 Report on IoT European Platforms." [Online]. Available: http://www.activageproject.eu/docs/downloads/activage_public_deliverables/ACTIVAGE_D3.1_M3_ReportonIoTEuropeanPlatforms_V1.0.pdf

[41] C. E. Palau *et al.*, "Introduction to Interoperability for Heterogeneous IoT Platforms," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 1–26. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_1

[42] A. Bassi, M. A. Llorente, M. Montesinos, and R. Gravina, "INTER-IoT Architecture for Platform Interoperability," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 49–94. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_3

[43] D. C. Y. Vargas and C. E. P. Salvador, "Smart IoT gateway for heterogeneous devices interoperability," *IEEE Latin America Transactions*, vol. 14, no. 8, pp. 3900–3906, aug 2016.

[44] P. Papcun *et al.*, "Edge-enabled IoT gateway criteria selection and evaluation," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 13, p. e5219, jul 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/cpe.5219https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5219https://onlinelibrary.wiley.com/doi/10.1002/cpe.5219

[45] C. López *et al.*, "Reviewing SDN Adoption Strategies for Next Generation Internet of Things Networks," *Smart Innovation, Systems and Technologies*, vol. 235, pp. 619–631, 2022. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-16-2877-1{_}57

[46] J. S. De Puga, C. E. Salvador, and A. B. Pellicer, "Architecture and use case for an IoT deployment with SDN at the edge and dual physical and

virtual gateway," *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, vol. 2019-July, jul 2019.

[47] I. Grønbæk, "Architecture for the Internet of Things (IoT): API and interconnect," *Proceedings - 2nd Int. Conf. Sensor Technol. Appl., SENSORCOMM 2008, Includes: MESH 2008 Conf. Mesh Networks; ENOPT 2008 Energy Optim. Wireless Sensors Networks, UNWAT 2008 Under Water Sensors Systems*, pp. 802–807, 2008.

[48] S. Pulparambil and Y. Baghdadi, "Service oriented architecture maturity models: A systematic literature review," *Computer Standards & Interfaces*, vol. 61, pp. 65–76, jan 2019.

[49] C. M. Sosa-Reyna, E. Tello-Leal, and D. Lara-Alabazares, "Methodology for the model-driven development of service oriented IoT applications," *Journal of Systems Architecture*, vol. 90, pp. 15–22, oct 2018.

[50] A. Palavalli, D. Karri, and S. Pasupuleti, "Semantic Internet of Things," *Proceedings - 2016 IEEE 10th International Conference on Semantic Computing, ICSC 2016*, pp. 91–95, mar 2016.

[51] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska, "Semantic technologies for the IoT - An Inter-IoT perspective," *Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pp. 271–276, may 2016.

[52] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska, "Identifier management in semantic interoperability solutions for IoT," *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018 - Proceedings*, pp. 1–6, jul 2018.

[53] M. Violette, "IoT Standards," *IEEE Internet of Things Magazine*, vol. 1, no. 1, pp. 6–7, nov 2018.

[54] V. Bhuvaneswari and R. Porkodi, "The internet of things (IOT) applications and communication enabling technology standards: An overview," *Proceedings - 2014 International Conference on Intelligent Computing Applications, ICICA 2014*, pp. 324–329, nov 2014.

[55] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, "Internet of Things (IoT): A Review of Its Enabling Technologies in Healthcare Applications, Standards Protocols, Security, and Market Opportunities," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 474–10 498, jul 2021.

[56] L. L. Dhirani, E. Armstrong, and T. Newe, "Industrial IoT, Cyber Threats, and Standards Landscape: Evaluation and Roadmap," *Sensors 2021, Vol. 21, Page 3901*, vol. 21, no. 11, p. 3901, jun 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3901/htmhttps://www.mdpi.com/1424-8220/21/11/3901

[57] ASSIST-IoT, "D3.1-State-of-the-Art and Market Analysis Report," 2021. [Online]. Available: https://assist-iot.eu/wp-content/uploads/2021/12/ASSIST-IoT_D3.1_State-of-the-art-And-Market-Analysis-Report_v1.0.pdf

[58] "ETSI." [Online]. Available: https://www.etsi.org/

[59] "IEEE." [Online]. Available: https://www.ieee.org/

[60] "IETF." [Online]. Available: https://www.ietf.org/

[61] "ISO." [Online]. Available: https://www.iso.org/home.html

[62] "ITU." [Online]. Available: https://www.itu.int/en/

[63] "oneM2M." [Online]. Available: https://www.onem2m.org/

[64] "W3C." [Online]. Available: https://www.w3.org/

[65] "Qué es middleware: definición y ejemplos — Microsoft Azure." [Online]. Available: https://azure.microsoft.com/es-es/overview/what-is-middleware/

[66] INTER-IoT, "D3.1. Methods for Interoperability and Integration," EC, Tech. Rep., 2016. [Online]. Available: https://files.inter-iot.eu/deliverables/accepted/D3.1-MethodsforInteroperabilityandIntegrationv.1.pdf

[67] M. A. da Cruz, J. J. Rodrigues, A. K. Sangaiah, J. Al-Muhtadi, and V. Korotaev, "Performance evaluation of IoT middleware," *Journal of Network and Computer Applications*, vol. 109, pp. 53–65, may 2018.

[68] A. Karim Mohamed Ibrahim, R. A. Rashid, A. H. F. A. Hamid, M. Adib Sarijari, and M. A. Baharudin, "Lightweight IoT middleware for rapid application development," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 17, no. 3, pp. 1385–1392, 2019.

[69] J. Cardoso, C. Pereira, A. Aguiar, and R. Morla, "Benchmarking IoT middleware platforms," *18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017 - Conference*, jul 2017.

[70] "¿Qué es el middleware?" [Online]. Available: https://www.redhat.com/es/topics/middleware/what-is-middleware

[71] F. Nammour and N. Mansour, "Comparative evaluation of object request broker technologies," in *ACS/IEEE International Conference on Computer Systems and Applications*. Institute of Electrical and Electronics Engineers (IEEE), jan 2004, p. 66.

[72] D. Happ, N. Karowski, T. Menzel, V. Handziski, and A. Wolisz, "Meeting IoT platform requirements with open pub/sub solutions," *Annales des Telecommunications/Annals of Telecommunications*, vol. 72, no. 1-2, pp. 41–52, feb 2017. [Online]. Available: https://link.springer.com/article/10.1007/s12243-016-0537-4

[73] "RabbitMQ." [Online]. Available: https://www.rabbitmq.com/

[74] "Apache Kafka." [Online]. Available: https://kafka.apache.org/

[75] "ActiveMQ." [Online]. Available: https://activemq.apache.org/

[76] "AmazonMQ." [Online]. Available: https://aws.amazon.com/es/amazon-mq/

[77] "ZeroMQ." [Online]. Available: https://zeromq.org/

[78] "Mosquitto." [Online]. Available: https://mosquitto.org/

[79] "HiveMQ." [Online]. Available: https://www.hivemq.com/

[80] "INTER-IoT ." [Online]. Available: https://inter-iot.eu/

[81] "ACTIVAGE." [Online]. Available: https://activageproject.eu/

[82] "DataPorts." [Online]. Available: https://dataports-project.eu/

[83] CREATE-IoT, "IoT FA strategy and coordination plan." [Online]. Available: https://european-iot-pilots.eu/wp-content/uploads/2017/10/D01_01_WP01_H2020_CREATE-IoT_Final.pdf

[84] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Computer Networks*, vol. 192, p. 108040, jun 2021.

[85] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, dec 2016.

[86] A. Ismail and H. Hamza, "Performance evaluation of open source iot platforms," *ieeexplore.ieee.org*, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8620130/

[87] H. Hejazi, H. Rajab, and T. Cinkler, "Survey of platforms for massive IoT," *ieeexplore.ieee.org*, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8325598

[88] G. Keramidas, N. Voros, and M. Hübner, "Components and Services for IoT Platforms," *Springer*, 2016. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-319-42304-3.pdf

[89] S. Lucero, "IoT platforms: enabling the Internet of Things," *cdn.ihs.com*, 2016. [Online]. Available: https://cdn.ihs.com/www/pdf/enabling-IOT.pdf

[90] A. A. Ismail, H. S. Hamza, and A. M. Kotb, "Performance Evaluation of Open Source IoT Platforms," *2018 IEEE Global Conference on Internet of Things, GCIoT 2018*, jan 2019.

[91] P. Ganguly, "Selecting the right IoT cloud platform," *2016 International Conference on Internet of Things and Applications, IOTA 2016*, pp. 316–320, sep 2016.

[92] J. Guth, U. Breitenbucher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of IoT platform architectures: A field study based on a reference architecture," *2016 Cloudification of the Internet of Things, CIoT 2016*, mar 2017.

[93] "FIWARE." [Online]. Available: https://www.fiware.org/

[94] "UniversAAL." [Online]. Available: https://www.universaal.info/

[95] "ONESAIT." [Online]. Available: https://www.onesait.com/

[96] "OPENIOT." [Online]. Available: https://github.com/OpenIotOrg

[97] "SENSINACT." [Online]. Available: https://wiki.eclipse.org/SensiNact

[98] "WSO2." [Online]. Available: https://wso2.com/

[99] "OneM2M." [Online]. Available: https://www.onem2m.org/

[100] "IBM Cloud Bluemix." [Online]. Available: https://cloud.ibm.com/login

[101] "Microsoft Azure IoT." [Online]. Available: https://azure.microsoft.com/es-es/overview/iot/

[102] "Google Cloud IoT." [Online]. Available: https://cloud.google.com/solutions/iot

[103] "AWS IoT." [Online]. Available: https://aws.amazon.com/es/iot/

[104] "SEAMS." [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_9

[105] "I3WSN." [Online]. Available: https://m.riunet.upv.es/handle/10251/54652

[106] "Body Cloud." [Online]. Available: https://projects.dimes.unical.it/spine-bok/body-of-knowledge/bodycloud/

[107] "KAA." [Online]. Available: https://www.kaaiot.com/

[108] "Thingspeak." [Online]. Available: https://thingspeak.com/

[109] "Eclipse OM2M." [Online]. Available: https://www.eclipse.org/om2m/

[110] "AllJoyn." [Online]. Available: https://openconnectivity.org/

[111] "OpenHab." [Online]. Available: https://www.openhab.org/

[112] "IoTivity." [Online]. Available: http://iotivity.org/

[113] "ThingWorx." [Online]. Available: https://www.ptc.com/es/products/thingworx

[114] "Developers Catalogue - FIWARE." [Online]. Available: https://www.fiware.org/developers/catalogue/

[115] G. Fortino, R. Gravina, W. Russo, and C. Savaglio, "Modeling and Simulating Internet-of-Things Systems: A Hybrid Agent-Oriented Approach," *Computing in Science and Engineering*, vol. 19, no. 5, pp. 68–76, 2017.

[116] A. Bröring *et al.*, "Enabling IoT Ecosystems through Platform Interoperability," *IEEE Software*, vol. 34, no. 1, pp. 54–61, jan 2017.

[117] A. L. Lemos, F. Daniel, and B. Benatallah, "Web Service Composition," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, dec 2015. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2831270

[118] T. Szydlo, R. Brzoza-Woch, J. Sendorek, M. Windak, and C. Gniady, "Flow-based programming for IoT leveraging fog computing," *Proceedings - 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2017*, pp. 74–79, aug 2017.

[119] C. S. Shih, C. C. Chuang, and H. Y. Yeh, "Federating public and private intelligent services for IoT applications," *2017 13th International Wireless Communications and Mobile Computing Conference, IWCMC 2017*, pp. 558–563, jul 2017.

[120] C. Savaglio, G. Fortino, and M. Zhou, "Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach," *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pp. 58–63, feb 2017.

[121] "UDDI — Online community for the Universal Description, Discovery, and Integration." [Online]. Available: http://uddi.xml.org/

[122] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language." [Online]. Available: https://www.w3.org/TR/wsdl20/

[123] "Hypercat." [Online]. Available: http://www.hypercat.io

[124] "iServe." [Online]. Available: http://iserve.kmi.open.ac.uk/

[125] "Introduction - WireCloud." [Online]. Available: https://wirecloud.readthedocs.io/en/stable/

[126] J. Soldatos *et al.*, "OpenIoT: Open source internet-of-things in the cloud," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9001, pp. 13–25, 2015.

[127] A. Belsa, D. Sarabia-Jacome, C. E. Palau, and M. Esteve, "Flow-based programming interoperability solution for IoT platform applications," *Proceedings - 2018 IEEE International Conference on Cloud Engineering, IC2E 2018*, pp. 304–309, may 2018.

[128] J. P. J. P. Morrison, *Flow-based programming : a new approach to application development.* J.P. Morrison, 2011. [Online]. Available: https://books.google.com/books/about/Flow_based_Programming.html?hl=es&id=5GFCnQAACAAJ

[129] "Node-RED." [Online]. Available: https://nodered.org/

[130] "Apache Nifi." [Online]. Available: https://nifi.apache.org/

[131] "IFTTT." [Online]. Available: https://ifttt.com/

[132] "Flogo." [Online]. Available: https://www.flogo.io/

[133] "Noflo." [Online]. Available: https://noflojs.org/

[134] "Node-RED." [Online]. Available: https://nodered.org/

[135] M. Blackstock and R. Lea, "Toward a distributed data flow platform for the Web of Things (Distributed Node-RED)," *ACM International Conference Proceeding Series*, vol. 08-October-2014, pp. 34–39, oct 2014. [Online]. Available: http://dx.doi.org/10.1145/2684432.2684439

[136] M. Blackstock and R. Lea, "FRED: A hosted data flow platform for the IoT," *Proceedings of the 1st International Workshop on Mashups of Things and APIs, MOTA 2016*, dec 2016.

[137] R. Kleinfeld, S. Steglich, L. Radziwonowicz, and C. Doukas, "glue.things-a Mashup Platform for wiring the Internet of Things with the Internet of Services," in *5th International Workshop on the Web of Things*, 2014. [Online]. Available: http://dx.doi.org/10.1145/2684432.2684436

[138] "Apache Flume." [Online]. Available: https://flume.apache.org/

[139] O. Vermesan and P. Friess, "Digitising the industry internet of things connecting the physical, digital and virtual worlds," *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*, pp. 1–338, aug 2016.

[140] "Download resources — IoT-EPI." [Online]. Available: https://iot-epi.eu/download-doc/

[141] "AGILE." [Online]. Available: http://agile-iot.eu/

[142] "Big IoT." [Online]. Available: http://big-iot.eu/

[143] "BioTope." [Online]. Available: https://biotope-project.eu/

[144] "TagItSmart." [Online]. Available: https://www.tagitsmart.eu/

[145] "SymbioTe." [Online]. Available: https://www.symbiote-h2020.eu/

[146] M. Schneider, B. Hippchen, S. Abeck, M. Jacoby, and R. Herzog, "Enabling IoT platform interoperability using a systematic development approach by example," *2018 Global Internet of Things Summit, GIoTS 2018*, nov 2018.

[147] "VICINITY." [Online]. Available: https://www.vicinity2020.eu/vicinity/

[148] AIOTI, "Advancing EU IoT Research and Innovation AIOTI's position on Horizon Europe and Digital Europe." [Online]. Available: https://aioti.eu/wp-content/uploads/2018/09/AIOTI_Position_Paper_HEU_2018_for_publishing.pdf

[149] "IoF2020." [Online]. Available: https://www.iof2020.eu/

[150] "MONICA." [Online]. Available: https://www.monica-project.eu/

[151] "AUTOPILOT." [Online]. Available: https://autopilot-project.eu/

[152] "SYNCHRONICITY ." [Online]. Available: https://synchronicity-iot.eu/

[153] LSP, "LARGE-SCALE PILOTS PROJECTS." [Online]. Available: https://european-iot-pilots.eu/wp-content/uploads/2018/03/220315_SD_IoT_Brochure_A4_Brief_LowRes_final-2.pdf

[154] LSP, "CREATE-IOT e-book." [Online]. Available: https://european-iot-pilots.eu/create-iot-e-book/

[155] LSP, "CREATE-IoT Deliverables." [Online]. Available: https://european-iot-pilots.eu/create-iot/deliverables-2/

[156] J. Y. Yu and Y. G. Kim, "Analysis of IoT Platform Security: A Survey," *2019 International Conference on Platform Technology and Service, PlatCon 2019 - Proceedings*, mar 2019.

[157] "Keycloack." [Online]. Available: https://www.keycloak.org/

[158] "Keyrock Identity Manager." [Online]. Available: https://fiware-idm.readthedocs.io/en/latest/

[159] "Kong." [Online]. Available: https://konghq.com/kong/

[160] "API Umbrella." [Online]. Available: https://apiumbrella.io/

[161] "Express Gateway." [Online]. Available: https://www.express-gateway.io//

[162] "Traefik." [Online]. Available: https://doc.traefik.io/traefik/

[163] "Istio." [Online]. Available: https://istio.io/

[164] M. Shao, K. Lu, and W. Zhang, "Survey on Virtualization of High Performance Computing," *2021 International Conference on Intelligent Computing, Automation and Applications (ICAA)*, pp. 39–41, jun 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9653557/

[165] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, sep 2014.

[166] D. Reis, B. Piedade, F. F. Correia, J. P. Dias, and A. Aguiar, "Developing Docker and Docker-Compose Specifications: A Developers' Survey," *IEEE Access*, vol. 10, pp. 2318–2329, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9658534/

[167] "Awesome Docker: A curated list of Docker resources and projects." [Online]. Available: https://github.com/veggiemonk/awesome-docker

[168] C. I. Valero *et al.*, "INTER-Framework: An Interoperability Framework to Support IoT Platform Interoperability," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 167–193. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_6

[169] A. Kumar Muppalla *et al.*, "Efficient Practices and Frameworks for Cloud-Based Application Development," *Computer Communications and Networks*, pp. 305–329, 2013. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4471-5031-2{_}14

[170] B. De, "API Management," in *API Management*. Apress, Berkeley, CA, 2017, pp. 15–28. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-1305-6_2

[171] J. Ofoeda, R. Boateng, and J. Effah, "Application programming interface (API) research: A review of the past to inform the future," *International Journal of Enterprise Information Systems*, vol. 15, no. 3, pp. 76–95, jul 2019.

[172] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating Web APIs on the world wide Web," *Proceedings - 8th IEEE European Conference on Web Services, ECOWS 2010*, pp. 107–114, 2010.

[173] N. Kratzke, "A Brief History of Cloud Application Architectures," *Applied Sciences 2018, Vol. 8, Page 1368*, vol. 8, no. 8, p. 1368, aug 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/8/1368/htmhttps://www.mdpi.com/2076-3417/8/8/1368

[174] R. C. Basole, "On the Evolution of Service Ecosystems: A Study of the Emerging API Economy," in *Handbook of Service Science, Volume II*, 2019, pp. 479–495.

[175] B. De, "API Documentation," in *API Management*. Apress, Berkeley, CA, 2017, pp. 59–80. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-1305-6_4

[176] P. Giménez, M. Llop, R. Gonzalez-Usach, and M. A. Llorente, "INTER-IoT Requirements," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 27–47. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_2

[177] P. Szmeja, M. Ganzha, M. Paprzycki, W. Pawlowski, and K. Wasielewska, "Declarative Ontology Alignment Format for Semantic Translation," *Proceedings - 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU 2018*, nov 2018.

[178] A. Belsa *et al.*, "INTER-Layer: A Layered Approach for IoT Platform Interoperability," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 95–132. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_4

[179] G. Fortino *et al.*, "INTER-Meth: A Methodological Approach for the Integration of Heterogeneous IoT Systems," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 195–230. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_7

[180] P. Giménez, M. Llop, J. Meseguer, F. Martin, and A. Broseta, "INTER-LogP: INTER-IoT for Smart Port Transportation," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 257–277. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_9

[181] G. Ibáñez-Sánchez, A. Fides-Valero, J.-L. Bayo-Monton, M. Gulino, and P. Pace, "Interoperability Application in e-Health," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 231–256. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_8

[182] R. Gonzalez-Usach *et al.*, "IoT Ecosystem Building," in *Interoperability of Heterogeneous IoT Platforms*. Springer, Cham, 2021, pp. 279–305. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-82446-4_10

[183] "AS2AS demostración IoT Week." [Online]. Available: https://www.youtube.com/watch?v=bvwuwiO8vQc

[184] D. Sarabia-Jacome, A. Belsa, C. E. Palau, and M. Esteve, "Exploiting IoT data and smart city services for chronic obstructive pulmonary diseases risk factors monitoring," *Proceedings - 2018 IEEE International Conference on Cloud Engineering, IC2E 2018*, pp. 351–356, may 2018.

[185] D. Yacchirema, A. Belsapellicer, C. Palau, and M. Esteve, "OneM2M Based-Interworking Architecture for Heterogeneous Devices Interoperability in IoT," *2018 IEEE Conference on Standards for Communications and Networking, CSCN 2018*, dec 2018.

[186] C. Panagiotou, C. Antonopoulos, G. Keramidas, N. Voros, and M. Hübner, "IoT in Ambient Assistant Living Environments: A View from Europe," *Components and Services for IoT Platforms: Paving the Way for IoT Standards*, pp. 281–298, jan 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-42304-3{_}14

[187] S. Stavrotheodoros, N. Kaklanis, K. Votis, and D. Tzovaras, "A Smart-Home IoT Infrastructure for the Support of Independent Living of Older Adults," *IFIP Advances in Information and Communication Technology*, vol. 520, pp. 238–249, may 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-92016-0{_}22

[188] C. I. Valero *et al.*, "AIoTES: Setting the principles for semantic interoperable and modern IoT-enabled reference architecture for Active and Healthy Ageing ecosystems," *Computer Communications*, vol. 177, pp. 96–111, sep 2021.

[189] A. M. Medrano-Gil *et al.*, "Definition of technological solutions based on the Internet of Things and Smart Cities paradigms for active and healthy ageing through cocreation," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[190] Sebastien Ziegler, P. Annicchino, and D. Stefanatou, "Personal Data Protection for Internet of Things Deployments," 2020. [Online]. Available: http://www.activageproject.eu/docs/publications/PersonalDataProtectionforIoTDeployments-final(MI).pdf

[191] K. L. A. Yau, S. Peng, J. Qadir, Y. C. Low, and M. H. Ling, "Towards Smart Port Infrastructures: Enhancing Port Activities Using Information and Communications Technology," *IEEE Access*, vol. 8, pp. 83 387–83 404, 2020.

[192] T. Inkinen, R. Helminen, and J. Saarikoski, "Port digitalization with open data: Challenges, opportunities, and integrations," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 5, no. 2, jun 2019.

[193] "PIXEL." [Online]. Available: https://pixel-ports.eu/

[194] J. P. Costa *et al.*, "Advantage of a Green and Smart Port of the Future," *WIT Transactions on The Built Environment*, vol. 204, pp. 203–217, sep 2021. [Online]. Available: http://www.witpress.com/elibrary/wit-transactions-on-the-built-environment/204/38062

[195] E. Simon, C. Garnier, I. Lacalle, J. P. Costa, and C. E. Palau, "Small and Medium Ports Activities Modelling: Introduction to the PIXEL Approach," *WIT Transactions on The Built Environment*, vol. 187, pp. 149–163, nov 2019. [Online]. Available: http://www.witpress.com/elibrary/wit-transactions-on-the-built-environment/187/37370

[196] M. Široka, S. Piličić, T. Milošević, I. Lacalle, and L. Traven, "A novel approach for assessing the ports' environmental impacts in real time – The IoT based port environmental index," *Ecological Indicators*, vol. 120, p. 106949, jan 2021.

[197] V. Araujo, K. Mitra, S. Saguna, and C. Åhlund, "Performance evaluation of FIWARE: A cloud-based IoT platform for smart cities," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 250–261, oct 2019.

[198] I. Zyrianoff, A. Heideker, L. Sciullo, C. Kamienski, and M. DI Felice, "Interoperability in Open IoT Platforms: WoT-FIWARE Comparison and Integration," *Proceedings - 2021 IEEE International Conference on Smart Computing, SMARTCOMP 2021*, pp. 169–174, aug 2021.

[199] P. Palaiogeorgou *et al.*, "AI: Opportunities and Challenges - The Optimal Exploitation of (Telecom) Corporate Data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12896 LNCS, pp. 47–59, sep 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-85447-8_5

[200] D. Sarabia-Jacome, I. Lacalle, C. E. Palau, and M. Esteve, "Enabling Industrial Data Space Architecture for Seaport Scenario," *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*, pp. 101–106, apr 2019.

[201] D. Sarabia-Jacome, C. E. Palau, M. Esteve, and F. Boronat, "Seaport Data Space for Improving Logistic Maritime Operations," *IEEE Access*, vol. 8, pp. 4372–4382, 2020.

[202] "Hyperledger Fabric." [Online]. Available: https://www.hyperledger.org/use/fabric

[203] A. Metzger, T. Kley, and A. Palm, "Triggering Proactive Business Process Adaptations via Online Reinforcement Learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12168 LNCS, pp. 273–290, sep 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58666-9_16

[204] E. Curry, A. Metzger, A. J. Berre, A. Monzón, and A. Boggio-Marzet, "A Reference Model for Big Data Technologies," *The Elements of Big Data Value*, pp. 127–151, 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-68176-0_6

[205] A. Palm, A. Metzger, and K. Pohl, "Online Reinforcement Learning for Self-adaptive Information Systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12127 LNCS, pp. 169–184, jun 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-49435-3_11

[206] A. Metzger, J. Franke, and T. Jansen, "Ensemble Deep Learning for Proactive Terminal Process Management at the Port of Duisburg "duisport"," *Business Process Management Cases Vol. 2*, pp. 153–164, 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-63047-1_12

[207] Á. Alonso, A. Pozo, J. M. Cantera, F. de la Vega, and J. J. Hierro, "Industrial Data Space Architecture Implementation Using FIWARE," *Sensors 2018, Vol. 18, Page 2226*, vol. 18, no. 7, p. 2226, jul 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/7/2226/htmhttps://www.mdpi.com/1424-8220/18/7/2226

[208] A. Munoz-Arcentales *et al.*, "Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE," *Sustainability 2020, Vol. 12, Page 3885*, vol. 12, no. 9, p. 3885, may 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/9/3885/htmhttps://www.mdpi.com/2071-1050/12/9/3885

[209] "PYNGSI." [Online]. Available: https://pypi.org/project/pyngsi/

[210] "BDVA/DAIRO Activity Group 44." [Online]. Available: https://www.bdva.eu/node/1783

[211] "Big Data Value Association." [Online]. Available: https://www.bdva.eu/

[212] "Standards, Data Platforms/Spaces and Industrial AI at the AG43." [Online]. Available: https://www.bdva.eu/node/1763

[213] "Webinar The PIXEL platform." [Online]. Available: https://www.youtube.com/watch?v=EA57OY3hDWU

[214] "IDC 2021: The 14th International Symposium on Intelligent Distributed Computing." [Online]. Available: http://www.idc2021.unirc.it/cfp.html

[215] "Big Data Value Forum, Data Spaces." [Online]. Available: https://www.youtube.com/watch?v=kUWl4-8p620&t=4447s

[216] I. Lacalle, A. Belsa, R. Vaño, and C. E. Palau, "Framework and methodology for establishing port-city policies based on real-time composite indicators and iot: A practical use-case," *Sensors (Switzerland)*, vol. 20, no. 15, pp. 1–41, aug 2020.