Tesis Doctoral

Arquitectura de análisis de datos generados por el Internet de las cosas (IoT) en tiempo real

Departamento de Comunicaciones

Universidad Politécnica de Valencia

Autor: David Fernando Sarabia Jácome

Director: Dr. D. Carlos E. Palau

Valencia, Junio 2020

Dedicado a mis padres y hermanos.

A Yuliya mi eterna compañera.

Resumen

El Internet de las cosas (IoT, del inglés, *Internet of Things*) está revolucionando la manera en que percibimos todo lo que nos rodea. La gran cantidad de objetos conectados a Internet diariamente revela el grado de aceptación de las tecnologías habilitadoras de IoT en los diferentes entornos de aplicación. De la misma manera, el gran volumen de datos generado por estos objetos, conocidos como dispositivos IoT, está llegando a valores inimaginables. Como resultado, las metodologías y técnicas tradicionales presentan limitaciones para la gestión de los datos recolectados por los dispositivos IoT. Este problema es conocido como Big Data y ha sido analizado en las dos últimas décadas en otro tipo de ámbitos (buscadores de páginas web, redes sociales, entre otros.). Sin embargo, la falta de conocimientos y estrategias claras para integrar las metodologías, técnicas y soluciones de Big Data con entornos de IoT está afectando directamente a los potenciales beneficios de IoT.

La gestión del Big Data es uno de los desafíos que afrontan actualmente los entornos de IoT. La presente tesis doctoral especifica una arquitectura para la gestión del Big Data generado por entornos IoT. La arquitectura fue diseñada utilizando los requerimientos planteados en las recomendaciones de la Unión Internacional de Telecomunicaciones (ITU-T) y el Instituto Nacional de Estándares y Tecnologías (NIST) para la implementación de ecosistemas IoT y la interoperabilidad de *frameworks* de Big Data, respectivamente. De esta manera, la arquitectura es lo suficientemente genérica para adaptarse a cualquier entorno IoT. La arquitectura Big Data es capaz de recopilar datos de dispositivos IoT, *gateways* IoT, plataformas IoT y espacios de datos virtuales en entornos industriales. Además, la arquitectura brinda el soporte para la generación de servicios innovadores basados en las

tendencias actuales en el área de la Inteligencia Artificial. Finalmente, la arquitectura aprovecha los recientes avances en la tecnología de *fog computing* y los modelos de servicios de *cloud computing* para implementar sus funcionalidades.

La arquitectura presentada ha sido aplicada en tres casos de uso derivados de los proyectos Europeos ACTIVAGE y PixelPort financiados por la Unión Europea. El primero de ellos tiene el objetivo de monitorizar, controlar y guiar durante el tratamiento de la apnea del sueño en adultos mayores. El segundo persigue la detección temprana de caídas en adultos mayores basado en algoritmos de Inteligencia Artificial. Y el último tiene el objetivo de explotar el Big Data compartido en el espacio de datos industriales para entornos marítimos con el fin de proporcionar información relevante para la planificación de las operaciones de los buques de contenedores.

Resum

La Internet de les coses (IoT, del anglès, Internet of Things) està revolucionant la manera en que percebem tot el que ens rodeja. La gran quantitat d'objectes connectats diàriament a Internet revela el grau de acceptació de les tecnologies facilitadores de IoT en els diferents entorns de la aplicació. De la mateixa manera, el gran volum de dades generades per aquests objectes, coneguts com dispositius IoT, està arribant a valors inimaginables. Com a resultat, les metodologies i tècniques tradicionals presenten limitacions per a la gestió de les dades recol · lectades pels dispositius IoT. Aquest problema es conegut com a Big Data i ha sigut analitzat durant les dos últimes dècades en tot tipus d'àmbits (buscadors de pàgines web i xarxes socials entre altres). No obstant, la falta de coneixements i estratègies clares per a integrar les metodologies, tècniques i solucions de Big Data en entorns de IoT està afectant directament als potencials beneficis de IoT.

La gestió del Big Data es un dels desafius que afronten actualment els entorns de IoT. Aquesta tesis doctoral especifica una arquitectura per a la gestió del Big Data generat pels entorns IoT. L'arquitectura ha sigut dissenyada utilitzant els requeriments plantejats en les recomanacions de la Unió Internacional de Telecomunicacions (ITU-T) i el Institut Nacional d'Estàndards i Tecnologies (NIST) per a la implementació d'ecosistemes IoT i la interoperabilitat de frameworks de Big Data. D'aquesta manera, l'arquitectura es lo suficientment genèrica per a adaptar-se a qualsevol entorn IoT. L'arquitectura Big Data es capaç de recopilar dades de dispositius IoT, gateways IoT, plataformes IoT i espais de dades virtuals en entorns industrials. Així mateix, l'arquitectura brinda el suport per a la generació de serveis innovadors basats en les tendències actuals en l'àrea de la Intel · ligència Artificial. Finalment, l'arquitectura aprofita els recents

avanços en la tecnologia de *fog computing* i els models de serveis de *cloud computing* per a implementar les seues funcionalitats.

L'arquitectura presentada ha sigut aplicada a tres casos d'usos derivats dels projectes europeus ACTIVAGE i PixelPort finançats per la Unió Europea. El primer d'ells té l'objectiu de monitoritzar, controlar i guiar durant el tractament de la apnea del somni en adults majors. El segon persegueix la detecció primerenca de caigudes en adults majors basat en algoritmes de Intel · ligència Artificial. I l'Últim té l'objectiu de explotar el Big Data compartint en l'espai de dades industrials per a entorns marítims amb el fi de proporcionar informació rellevant per a la planificació de les operacions dels vaixells de contenidors.

Abstract

The Internet of Things (IoT) is revolutionizing the way we perceive everything around us. The large number of objects connected to the Internet reveals the degree of acceptance of IoT-enabling technologies in several domain applications. In the same way, the large volume of data generated by these objects, also known as IoT devices, is reaching unimaginable values. As a result, traditional methodologies and techniques are not capable of managing the large amount of data collected by IoT devices. This problem is known as Big Data, and it has been analyzed in the last two decades in other applications contexts (i.e., web page search engines, social networks, among others). However, the lack of clear knowledge and strategies to integrate Big Data methodologies, techniques and solutions with IoT environments is directly affecting the potential benefits of IoT.

Nowadays, Big Data management is one of the challenges that IoT environments are facing. For this reason, this doctoral thesis specifies an architecture for the management of Big Data generated by IoT environments. The Big Data architecture proposed was designed using the requirements outlined in the recommendations of the International Telecommunication Union (ITU-T) and the National Institute of Standards and Technologies (NIST) for the implementation of IoT ecosystems and the interoperability of Big Data frameworks. In this way, the architecture is generic enough for adapting to any IoT environment. Big Data architecture is capable of collecting data from IoT devices, IoT gateways, IoT platforms, and the industrial virtual data spaces. Also, the architecture provides support for the generation of innovative services based on current trends in Artificial Intelligence. Finally, the architecture takes advantage of the recent advances

in fog computing technology and the cloud computing model services for implementing its functionalities.

The architecture presented has been applied in three use cases derived from the European ACTIVAGE and PixelPort projects funded by the European Union. The first of these uses cases aims to monitor, control, and guide during the treatment of sleep apnea in elderly. The second one pursues the early detection of the elderly's fall based on Artificial Intelligence algorithms. The last one has the objective of exploiting shared Big Data in industrial data space for maritime environments to provide relevant information for the planning of shipping container operations.

Agradecimientos

En primer lugar, un agradecimiento sincero y profundo a mi director Dr. D. Carlos Enrique Palau por la oportunidad de aprender de su mano y por haber depositado en mí su entera confianza.

Al Estado Ecuatoriano y a la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) por haber apoyado la realización de esta tesis doctoral a través de su programa de Becas.

Con especial cariño, a mis padres y hermanos que siempre los sentí cerca de mí a pesar de la distancia que nos separaba.

Por último y no menos importante, a Yuliya quien con su infinito amor y paciencia me apoyo durante esta larga travesía.

El camino recorrido durante esta linda etapa de mi vida ha estado lleno de muchas dificultades y retos. No hubiese sido posible superar todos los obstáculos sin su incondicional apoyo.

Muchas gracias a todos.

David Sarabia-Jácome
Valencia, 2020

Índice general

Índice de figuras

Índice de tablas

Acrónimos

6LowPAN	*IPv6 over Low power Wireless Personal Area Network*
AAL	*Ambient Assisted Living*
AE	*Auxiliary Engine*
AHA	*Active and Healthy Ageing*
AIOTI	*Alliance for Internet of Things Innovation*
AIOTI HLA	*AIOTI High Level Architecture*
AIS	*Automatic Identification System*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
BDLM	*Big Data Lifecycle Management*
BDV	*Big Data Value*
BDVA	*Big Data Value Association*
BI	*Business Intelligence*
BSON	*Bynary Java Script Object Notation*
CEP	Procesador de Eventos Complejos
CKAN	*Comprehensive Knowledge Archive Network*
CNN	*Convolutional Neural Network*
CO	Monóxido de Carbono
CO_2	Dióxido de Carbono
CoAP	*Constrained Application Protocol*
COG	*Course Over Ground*
CPS	*Cyber Physical System*
CPU	*Central Processing Unit*
CQL	*Cassandra Query Language*
CRM	*Customer Relationship Management*
CRUD	*Create Read Update Delete*

CSV	*Comma Separated Values*
DAG	*Directed Acyclic Graph*
DBMS	*Database Management System*
DBSCAN	*Density-based Spatial Clustering of Applications with Noise*
EB	*Exabyte*
ECG	Electrocardiograma
EDI	*Electronic Data Interchange*
EDP	Espacio de Datos Portuario
ERP	*Enterprise Resource Planning*
GB	*Gigabyte*
GDPR	*General Data Protection Regulation*
GE	*Generic Enabler*
GFS	*Google File System*
GHz	*Gigahercio*
GIS	*Geographic Information System*
GPS	Sistema de Posicionamiento Global
GPU	*Graphics Processing Unit*
GRU	*Gated Recurrent Unit*
GSMA	*Global System for Mobile Communication Association*
GUI	*Graphical User Interface*
HDD	*Hard Disk Drive*
HDFS	*Hadoop Distributed File System*
HPC	*High Performance Computing*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IA	Inteligencia Artificial
IaaS	*Infrastructure as a Service*
IDC	*International Data Corporation*
IdM	*Identity Manager*
IDS	*Industrial Data Space*
IDSA	*International Data Spaces Association*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
IIC	*Industrial Internet Consortium*
IIRA	*Industrial Internet Reference Architecture*
IMO	*International Maritime Organization*
IoT	*Internet of Things*
IoTWF	*Internet of Things World Forum*
ISO	*International Organization for Standarization*
ITU-T	Unión Internacional de Telecomunicaciones
IU	Interfaz de Usuario
JSON	*Java Script Object Notation*

K-NN	*K-Nearest Neighbours*
KPI	*Key Performance Indicator*
KVM	*Kernel-based Virtual Machine*
LAN	*Local Area Network*
LSTM	*Long Short-Term Memory*
LTE	*Long Term Evolution*
LXC	*Linux Container*
M2M	*Machine to Machine*
MAE	*Mean Absolute Error*
MB	*Megabyte*
ME	*Main Engine*
MEC	*Mobile Edge Computing*
MLP	*Multilayer Perceptron*
MMSI	*Maritime Mobile Service Identity*
MQTT	*Message Queue Telemetry Transport*
MSE	*Mean-Squared Error*
NBD-PWG	*NIST Big Data Working Group*
NBDRA	*NIST Big Data Reference Architecture*
NGSI	*Next Generation Service Interface*
NIST	Instituto Nacional de Estándares y Tecnologías
NO	Monóxido de Nitrógeno
NO_2	Dióxido de Nitrógeno
NoSQL	*No Structured Query Language*
NO_x	Óxidos de Nitrógeno
O_3	Ozono
OMS	Organización Mundial de la Salud
OPC UA	*Open Platform Communications Unified Architecture*
PaaS	*Platform as a Service*
PAP	*Policy Administration Point*
PB	*Petabyte*
PCA	*Principal Component Analysis*
PCS	*Port Community Systems*
PDP	*Policy Decision Point*
PE	*Processing Element*
PEP	*Policy Enforcement Point*
PLM	*Product Lifecycle Management*
PM_{10}	Partículas en Suspensión 2.5 - 10 μm
PM_{25}	Partículas en Suspensión < 2.5 μm
PortCDM	*Port Collaboration for Decision Making*
PSG	Polisomnografía
QoE	*Quality of Experience*
QoS	*Quality of Service*
RAM	*Random Access Memory*

RAMI	*Reference Architectural Model Industrie*
RDD	*Resilient Distributed Dataset*
ReLU	*Rectified Linear Unit*
REST	*Representational State Transfer*
REST	*Representational State Transfer*
RFID	*Radio Frecuency Identification*
RMSE	*Root-Mean-Squared Error*
RNN	*Recurrent Neural Network*
ROT	*Rate of Turn*
RVVCCA	Red Valenciana de Vigilancia y Control de la Contaminación Atmosférica
S3	*Simple Storage Service*
SaaS	*Software as a Service*
SATRD	Sistemas y Aplicaciones de Tiempo Real Distribuido
SDK	*Software Development Kit*
SenaaS	*Sensing as a Service*
SFC	*Specific Fuel Consumption*
SG	*Study Group*
SO_2	Dióxido de Azufre
SOG	*Speed Over Ground*
SOM	*Self-Organizing Maps*
SQL	*Structured Query Language*
SSD	*Solid State Drive*
SSL	*Secure Sockets Layer*
STM	*Sea Traffic Management*
SVM	*Support Vector Machine*
TB	*Terabyte*
TLS	*Transport Layer Security*
UPV	Universidad Politécnica de Valencia
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
VHF	*Very High Frequency*
VLCi	Valencia Ciudad Inteligente
WiFi	*Wireless Fidelity*
WSAN	*Wireless Sensor and Actuator Network*
XACML	*eXtensible Access Control Markup Language*
XML	*Extensible Markup Language*
YARN	*Yet Another Resource Negotiator*
ZB	*Zettabyte*

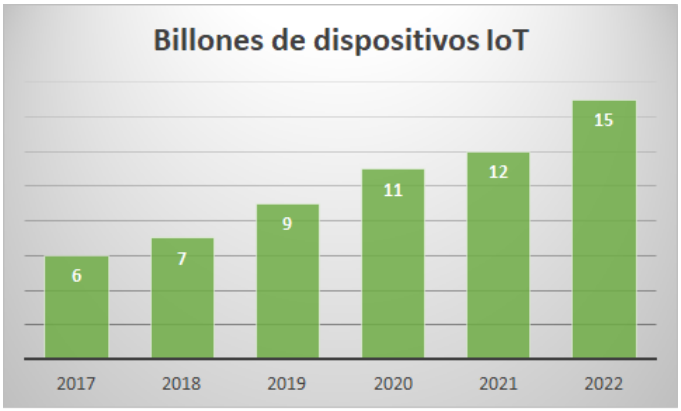
Capítulo 1

Introducción

1.1 Introducción

El que un día fue el sueño en que las cosas puedan interactuar con los humanos se ha vuelto realidad con la rápida evolución tecnológica de los últimos años. El concepto del Internet de las cosas (IoT, por sus siglas en inglés, *Internet of Things*) engloba esta idea de interconexión entre el mundo físico y el virtual permitiendo dicha interacción en cualquier momento y en cualquier lugar [1]. Por ello, IoT se ha convertido en uno de los pilares de la transformación digital de todo lo que nos rodea. Esta transformación digital lleva consigo cambios en la forma en que la industria gestiona sus operaciones y sus activos, como las personas controlan su salud y bienestar, incluso como las ciudades gestionan sus recursos para afrontar las necesidades de sus ciudadanos [2].

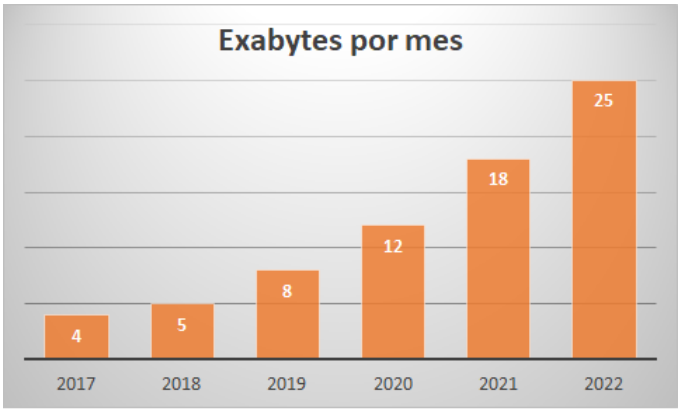
En los últimos años, el número de objetos conectados a Internet ha evolucionado exponencialmente debido a la amplia aceptación en sus diferentes dominios de aplicación. Cada día, millones de objetos están siendo conectados a Internet, entre los que se encuentran vehículos, monitores de salud, luces, televisiones, termostatos, entre otros [3]. En 2017, 6 billones de dispositivos IoT fueron conectados a Internet y se prevé un crecimiento 2.4 veces mayor llegando a 15 billones de conexiones de dispositivos IoT para el 2022, como muestra la Figura 1.1 [3].



Fuente: Cisco, 2019 [3]

Figura 1.1: Evolución de la cantidad de dispositivos IoT

Conjuntamente con este crecimiento, la cantidad de datos generados por los dispositivos IoT se incrementará exponencialmente para los próximos años, como se observa en la Figura 1.2. Los dispositivos IoT conectados a Internet generaron 3.7 *Exabyte* (EB) por mes en el 2017 y se estima un crecimiento de más de 25 EB para finales del 2022 [3]. Además, el volumen de datos generados por las personas, máquinas y cosas alcanzará 847 *Zettabyte* (ZB) en el 2021, superando los 218 ZB generados en el 2016 [4]. Aquellas cifras eran impensables en otros tiempos, pero se están convirtiendo en un serio desafío lidiar con ellas en la actualidad.



Fuente: Cisco, 2019 [3]

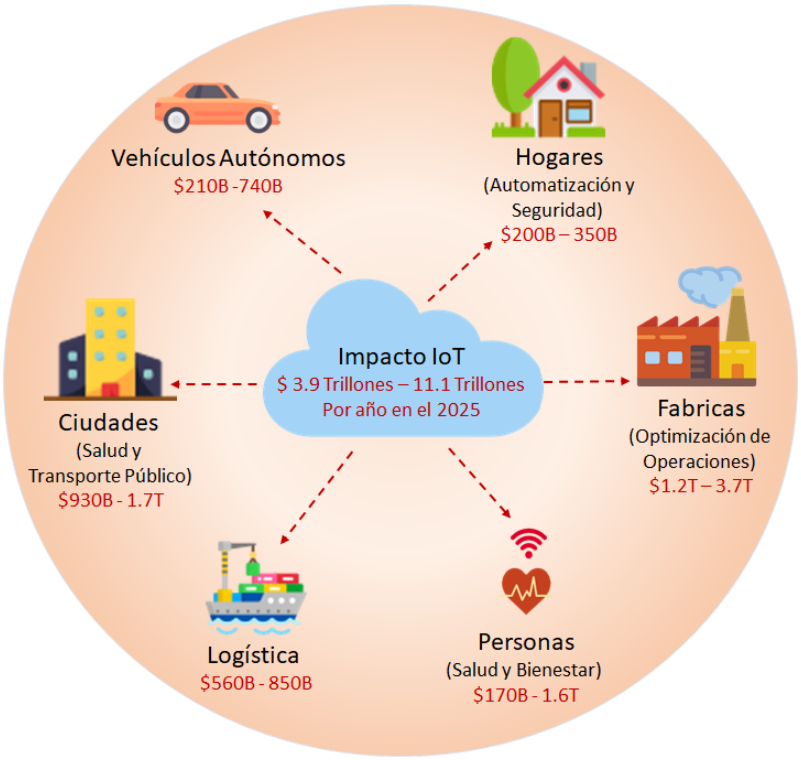
Figura 1.2: Evolución de la cantidad de datos generados por IoT

Batallar acertadamente con el progresivo aumento de los datos generados por IoT es una tarea tecnológica desafiante y que requiere de nuevas e innovadoras perspectivas. Los enfoques tradicionales utilizados en el proceso de almacenado, procesado y análisis son ineficientes, poco escalables y poco prácticos para el manejo de un gran volumen de datos [5]. Con el fin de afrontar las limitaciones de los enfoques tradicionales se han diseñado nuevas arquitecturas, tecnologías y técnicas orientadas al manejo de grandes volúmenes de datos y que engloban un concepto tecnológico conocido como Big Data [6]. El conjunto de metodologías y técnicas de Big Data nacieron con el objetivo de manejar los grandes volúmenes de datos generados por las páginas web y las redes sociales para extraer información útil.

La extracción de información de los datos generados por los dispositivos IoT es importante debido a que aportará un sin número de beneficios económicos, tecnológicos y sociales. La extracción del conocimiento de los datos generados por los dominios de aplicación de IoT permitirá entender los diferentes procesos industriales con el fin de optimizarlos para favorecer el crecimiento económico, como resume la Figura 1.3. De hecho, el potencial crecimiento económico producido por IoT se estima en 11.1 trillones de dólares por año para el 2025 [2]. Además, el gran volumen de datos impulsará el avance de otras áreas tecnológicas como por ejemplo la Inteligencia Artificial (IA), donde se podrá generar nuevos algoritmos y modelos útiles para la explotación de datos. En la actualidad, un bajo porcentaje de los datos recolectados por los dispositivos IoT es utilizado para el desarrollo de aplicaciones relacionadas a la detección de anomalías, sin aún considerar aplicaciones con gran potencial como la predicción y optimización [2]. No obstante, los beneficios serán más notorios en la mejora de la calidad de vida de las personas. El conocimiento extraído de los datos permitirá entender los factores y problemas envueltos en el entorno en el que vivimos con el fin de optimizarlo para proveer una mejor calidad de vida. Los beneficios del Big Data generados por los dominios de aplicación de IoT serán el resultado de un adecuado manejo de las características de los datos de IoT y sus requerimientos.

Desde el punto de vista de IoT, las metodologías y técnicas de Big Data representan el aliado tecnológico que necesitaba para lidiar con el incremento progresivo de sus datos. Sin embargo, IoT como fuente de Big Data presenta requerimientos más complejos de resolver en relación a otras fuentes de datos. El Big Data generado por IoT presenta características como la heterogeneidad, la correlación espacio-tiempo, la efectividad de los datos y la recolección de datos a gran escala que dificultan el procesamiento de los mismos [7]. Actualmente, varias arquitecturas han sido propuestas para el manejo del Big Data generado por IoT considerando las características mencionadas. Estas arquitecturas emplean un enfoque basado en *cloud*

computing para el procesamiento en tiempo real de los datos de IoT. No obstante, el enfoque basado en *cloud computing* ha mostrado varias limitaciones para el manejo adecuado de los altos volúmenes de datos de IoT. El manejo de datos en *cloud computing* provoca una alta latencia (tiempo transcurrido desde la toma de medidas por los sensores, envío al *cloud*, preprocesamiento, extracción de patrones, toma de decisiones), gran consumo de ancho de banda, alto costo de almacenamiento en la nube, y problemas de seguridad y privacidad de datos. Estos problemas dificultan el proceso de extracción de valor en tiempo real.



Fuente: Mckinsey, 2015 [2]

Figura 1.3: Impacto económico de IoT

Este primer capítulo presenta una breve introducción a la temática y resalta la importancia del manejo del Big Data generado por los dominios de aplicación de IoT para la extracción de valor; así como también, presenta las principales motivaciones que originaron la presente investigación y detalla los objetivos que condujeron su desarrollo. Finalmente, el capítulo enlista las principales aportaciones que respaldan este trabajo y detalla la estructura del documento.

1.2 Motivación

En los últimos años, el desarrollo de arquitecturas de Big Data para solucionar los problemas de manejo del gran volumen de datos generados por los dominios de aplicación de IoT ha despertado gran interés en la comunidad científica [8]. Las arquitecturas presentadas tienen el objetivo de investigar las estrategias del manejo de los datos de los dominios de IoT empleando las metodologías y técnicas de Big Data con soporte en *cloud computing*. En la actualidad, nuevos obstáculos han sido identificados en la literatura actual que dificultan la extracción de conocimiento del Big Data generado por las aplicaciones de IoT. Por ello, la presente tesis se enmarca en una investigación aplicada y enfocada en solventar los problemas de Big Data en dominios de aplicación de IoT para brindar conocimientos que faciliten la adopción exitosa de las metodologías y técnicas de Big Data.

Si bien los dominios de aplicación de IoT son amplios, la presente tesis se enfoca en analizar los requerimientos particulares de dos dominios de aplicación: la salud y transporte. El análisis de estos dominios de aplicación evidenció ciertas carencias que deben ser superadas para generar valor. Estas carencias son las principales motivaciones para el desarrollo de esta tesis y se resumen a continuación:

- **Arquitectura integral**

  El componente funcional que más atención ha generado es el encargado del análisis de los datos a través de algoritmos de IA. Varios modelos han sido desarrollados para predecir o detectar patrones en áreas como la salud, transporte, medio ambiente, energía, entre otros. Sin embargo, este tipo de soluciones aisladas en un solo componente no proveen conocimientos sobre la interacción con el resto de componentes funcionales de la arquitectura de Big Data. Por ello, una arquitectura integral que presente el funcionamiento de todos los componentes involucrados en el análisis de los datos es relevante para contrarrestar la desigualdad en el avance de la técnica.

- **Arquitectura flexible y adaptable**

  Las arquitecturas propuestas se enfocan en resolver los requerimientos particulares de un determinado dominio de aplicación de IoT. El problema se genera cuando se requiere adaptar esa arquitectura a los requerimientos de otra aplicación, provocando un fracaso, pérdida de

tiempo y recursos. El diseño de una arquitectura genérica, en la medida de lo posible y que no dependa de un dominio de aplicación proporciona un punto de partida en el diseño de una arquitectura capaz de adaptarse a los requerimientos particulares de cada aplicación.

- **Distribución de la inteligencia cerca a la fuente de datos**

  El actual enfoque de procesamiento basado en *cloud computing* propuesto para manejar el gran volumen de datos presenta problemas en el proceso de recolección de los datos. El alto volumen de datos generados por una gran cantidad de dispositivos IoT produce saturación de ancho de banda y considerables retardos, ocasionando una baja calidad de servicio (QoS, del inglés, *Quality of Service*). El uso de tecnologías de *fog computing* ha demostrado ser adecuado para reducir las limitaciones de *cloud computing*. Además, el análisis de los datos utilizando nodos *fog* ubicados en el borde de la red (también conocido como *edge analytics*) provee mejores prestaciones en el manejo del Big Data, a la vez que combate las limitaciones del procesamiento en el *cloud*.

- **Compartición de datos en la industria**

  Las limitaciones en mantener la propiedad y control de los datos de las actuales arquitecturas inhiben la integración de múltiples fuentes en el sector industrial. La industria está considerando a los datos como su activo más valioso y por ello no concibe la idea de compartir sus datos. Como consecuencia, la información extraída del Big Data no tiene la suficiente relevancia para optimizar los procesos industriales. Un entorno seguro y confiable para compartir datos brindará al Big Data más fuentes de datos de explotación para generar mejores y más precisos análisis que conduzcan a una mejor toma de decisiones en la industria.

- **Desarrollo de servicios innovadores y extracción de información**

  Los beneficios del Big Data son notorios cuando se extrae información relevante para la toma de decisiones o extracción de conocimiento para generar servicios innovadores. Recientemente, las técnicas basadas en la IA así como las herramientas de análisis han permitió el desarrollo de modelos más precisos y confiables de forma rápida. La aplicación de estas nuevas técnicas permite el desarrollo de servicios innovadores y la extracción de información relevante en la toma de decisiones.

La presente tesis doctoral expone una arquitectura integral, flexible y adaptable de Big Data para extraer el conocimiento de los datos generados por los dominios de aplicación de IoT. La utilidad y viabilidad de la arquitectura ha sido demostrada con tres casos de uso de IoT dentro del ámbito de la salud y transporte. Los casos de uso están enmarcados en el contexto de dos proyectos de investigación financiados por la Unión Europea a través de programa Horizonte 2020 (H2020):

- **ACTIVAGE Project**[1] que tiene como objetivo presentar un ecosistema basado en IoT para soportar un envejecimiento activo y saludable de los adultos mayores.
- **PIXEL-PORT (Port IoT for Environmental Leverage)**[2] que tiene como objetivo presentar una solución flexible y escalable para reducir el impacto ambiental y permitir la optimización de las operaciones en puertos marítimos a través de IoT.

1.3 Objetivos

A partir de las motivaciones enunciadas, la presente tesis atiende a la resolución del problema del manejo de grandes volúmenes de datos generados en aplicaciones IoT, contribuyendo de esta forma al futuro despliegue de aplicaciones IoT a gran escala y sobretodo su adopción en entornos industriales.

1.3.1 Objetivo general

En tal virtud, el objetivo principal de la presente tesis doctoral se enfoca en:

> Diseñar una arquitectura para el manejo eficiente de los datos generados por IoT, utilizando el conjunto de metodologías y técnicas de Big Data, *cloud computing*, e IA para generar innovadores servicios basados en la explotación de los datos masivos recolectados en el ámbito de la salud y transporte.

[1] https://www.activageproject.eu/
[2] https://pixel-ports.eu/

1.3.2 Objetivos específicos

Los siguientes objetivos específicos son planteados para alcanzar el objetivo principal:

O1. Identificar el problema de manejo de grandes volúmenes de datos y los retos que afrontan actualmente los ecosistemas IoT, así como su relevancia económica y social con el fin de aportar mejoras en el estado del arte.

O2. Analizar las metodologías, estrategias y las tecnologías habilitadoras relacionadas con el manejo de grandes volúmenes de datos por medio de la compilación de la bibliografía necesaria de soporte y el estudio de conceptos relacionados con la temática para sentar las bases en el desarrollo de la solución al problema planteado.

O3. Diseñar una arquitectura para la gestión eficiente de los grandes volúmenes de datos generados por los dispositivos IoT siguiendo los estándares y recomendaciones técnicas para que sea capaz de extenderse a cualquier dominio de aplicación. La modularidad de la arquitectura permitirá la adaptación a los requerimientos funcionales y no funcionales de los diferentes entornos de IoT.

O4. Aplicar la arquitectura propuesta en un entorno controlado enfocado en la integración con un sistema de monitorización de la salud remoto basado en IoT, a través de una instancia de la arquitectura empleando *software open source* con el fin de proveer información útil para la toma de decisiones.

O5. Aplicar la arquitectura propuesta en un entorno controlado enfocado en el uso de algoritmos de IA como técnica de explotación de los grandes volúmenes de datos recolectados por los dispositivos IoT en un escenario AAL (del inglés, *Ambient Assisted Living*), con el fin de mejorar tiempos de respuesta en caso de emergencia.

O6. Aplicar la arquitectura propuesta en un entorno controlado enfocado en la monitorización de las operaciones de transporte marítimo basado en IoT, mediante la integración de múltiples fuentes de datos empleando un entorno seguro y confiable para compartir datos, con el fin de generar información útil para la mejora y optimización de las operaciones portuarias.

1.4 Principales aportaciones

1.4.1 Artículos

- **Sarabia-Jácome, D.**, Usach, R., Palau, C. E., Esteve, M. (2020). "Highly-Efficient Fog-Based Deep Learning Aal Fall Detection System". Internet of Things, 100185. [9]
- **Sarabia-Jácome, D.**, Palau, C. E., Esteve, M., Boronat, F. (2019). "Seaport Data Spaces for improving logistic maritime operations". IEEE Access, 8, 4372-4382. [10]
- Yacchirema, D., **Sarabia-Jácome, D.**, Palau, C. E., Esteve, M. (2018). "System for monitoring and supporting the treatment of sleep apnea using IoT and big data". Pervasive and Mobile Computing, 50, 25-40. [11]
- Yacchirema, D., **Sarabia-Jácome, D.**, Palau, C. E., Esteve, M. (2018). "A smart system for sleep monitoring by integrating IoT with big data analytics". IEEE Access, 6, 35988-36001. [12]

1.4.2 Congresos

- **Sarabia-Jácome, D.**, Belsa, A., Palau, C. E., Esteve, M. (2018, April). "Exploiting IoT Data and Smart City Services for Chronic Obstructive Pulmonary Diseases Risk Factors Monitoring". In 2018 IEEE International Conference on Cloud Engineering (IC2E) (pp. 351-356). IEEE. [13]
- Belsa, A., **Sarabia-Jácome, D.**, Palau, C. E., Esteve, M. (2018, April). "Flow-based programming interoperability solution for IoT Platform Applications". In 2018 IEEE International Conference on Cloud Engineering (IC2E) (pp. 304-309). IEEE. [14]
- **Sarabia-Jácome, D.**, Lacalle, I., Palau, C. E., Esteve, M. (2019, April). "Efficient Deployment of Predictive Analytics in Edge Gateways: Fall Detection Scenario". 2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (WF-IoT 2019). [15]
- **Sarabia-Jácome, D.**, Lacalle, I., Palau, C. E., Esteve, M. (2019, April). "Enabling Industrial Data Space Architecture for Seaport Scenario". 2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (WF-IoT 2019). [16]

1.4.3 Capítulos de libro

- **Sarabia-Jácome, D.**, Gonzalez-Usach, R., Palau, C. E. (2019). "IoT Big Data Architectures, Approaches, and Challenges: A Fog-Cloud Approach. In Handbook of Research on Big Data and the IoT (pp. 125-148). IGI Global. [17]

1.4.4 Participación en proyectos de investigación

La presente tesis doctoral se ha desarrollado en el contexto del grupo de investigación Sistemas y Aplicaciones de Tiempo Real Distribuido (SATRD) del Departamento de Comunicaciones de la Universidad Politécnica de Valencia (UPV).

La tesis doctoral ha realizado colaboraciones en los siguientes proyectos de investigación y desarrollo, y algunas publicaciones han sido financiadas por los mismos:

- Proyecto ACTIVAGE: "*Supporting Active and Healthy Ageing through IoT technologies*". Programa de investigación e innovación Horizon 2020 de la Unión Europea bajo concesión No. 732679. Enero 2017 - Diciembre 2019.
- Proyecto PIXEL Port: "*Port IoT for Environmental Leverage*". Programa de investigación e innovación Horizon 2020 de la Unión Europea bajo concesión No. 769355. Mayo 2018 - Abril 2021.

1.4.5 Software y conjuntos de datos

Los códigos de las aplicaciones y configuraciones realizadas para el desarrollo del estudio de los casos de uso de la arquitectura en los entornos IoT se encuentran en un repositorio en la plataforma GitHub[3]. Además, el conjunto de datos empleados para evaluar la instancia de la arquitectura de Big Data en el dominio de aplicación de IoT del transporte se encuentra publicado en el repositorio IEEE dataport:

- Sarabia-Jácome, D., Palau, C. E. (2020). "Data from Vessels Transportation around the Spain Mediterranean Area". IEEE DataPort. [18]

[3]GitHub: https://github.com/esmaxness/

1.5 Estructura de la memoria

La memoria de tesis está estructurada de la siguiente manera:

- En este capítulo se realiza una breve introducción a IoT y Big Data, así como describe la importancia del manejo del volumen de los datos generados IoT y su impacto económico y social. Adicionalmente, el capítulo describe las motivaciones del problema del volumen de datos generados por IoT y los objetivos que guiaron durante la investigación. Finalmente, el capítulo detalla las principales contribuciones realizadas en la línea de investigación y describe la estructura de la tesis. El capítulo está relacionado con el objetivo **O1**.
- En este capítulo se presentan las metodologías, estrategias y tecnologías habilitadoras relacionadas con Big Data e IoT. Además, el capítulo contiene una revisión de las arquitecturas diseñadas para el manejo de grandes volúmenes de datos. Finalmente, el capítulo analiza el uso de *cloud computing* y *fog computing* como habilitadores tecnológicos en la integración entre IoT y Big Data. El capítulo está relacionado con el objetivo **O2**.
- El tercer capítulo realiza una descripción de la arquitectura de Big Data para el soporte del gran volumen de datos generados en entornos IoT. Inicialmente, el capítulo detalla los requerimientos funcionales y no funcionales para manejar los datos generados en entornos IoT. Además, el capítulo describe desde varios puntos de vista roles, componentes funcionales, procesos y patrones de implementación de la arquitectura de Big Data. El capítulo está relacionado con el objetivo **O3**.
- El cuarto capítulo presenta la aplicación de la arquitectura propuesta para un caso de uso enfocado en el cuidado de la salud y bienestar. En este caso de uso se valida la integración de la arquitectura de Big Data para IoT con un sistema de IoT implementado para la monitorización de la apnea del sueño y cubrir sus requerimientos de gestión de Big Data. La instancia de la arquitectura de Big Data para IoT se integra con el sistema IoT basado en la plataforma FIWARE. El capítulo describe la implementación de los procesos para extraer información y generar servicios de visualización y predicción de los contaminantes para soportar el tratamiento de la apnea del sueño. Finalmente, el capítulo presenta la evaluación del caso de uso. El capítulo está relacionado con el objetivo **O4**.
- El quinto capítulo presenta la aplicación de la arquitectura propuesta para un caso de uso enfocado en la habilitación de entornos inteli-

gentes relacionado con el cuidado de la salud y bienestar. En este caso de uso se valida el uso de *fog computing* como tecnología habilitadora y las ventajas del despliegue de servicios de predicción a nodos *fog* para la detección de caídas en adultos mayores. El capítulo describe la implementación de los procesos para extraer información, generar el servicio de detección de caídas usando algoritmos de *deep learning* y desplegar los modelos de detección de caídas a los nodos *fog*. Finalmente, el capítulo presenta la evaluación de las prestaciones de la implementación de la arquitectura utilizando *fog computing*. El capítulo está relacionado con el objetivo **O5**.

- El sexto capítulo presenta la aplicación de la arquitectura propuesta para un caso de uso enfocado en el transporte marítimo. En este caso de uso se valida la explotación del Big Data en espacios de datos virtuales a través de una instancia de la arquitectura de Big Data propuesta para la monitorización de la flota de buques de una naviera. Además, el capítulo describe los procesos implementados para la explotación de los datos con el fin de generar *Key Performance Indicator* (KPI) de interés para la naviera que son presentados a los usuarios finales a través de un servicio de visualización. Finalmente, el capítulo presenta la evaluación del sistema y sus beneficios tanto económicos como medio ambientales para el transporte marítimo. El capítulo está relacionado con el objetivo **O6**.
- El séptimo capítulo expone las conclusiones finales derivadas de la investigación realizada y las futuras líneas de investigación.

Capítulo 2

Estado del Arte

2.1 Introducción

El concepto de IoT es el presente tecnológico destinado a impulsar el avance industrial y sobre todo a mejorar la calidad de vida de las personas. El grado de penetración de las tecnologías habilitadoras de IoT está llegando a niveles inimaginables gracias a los potenciales beneficios económicos, tecnológicos y sociales derivados de su utilización. Sin embargo, IoT enfrenta desafíos que están frenando la obtención de sus beneficios. Uno de estos desafíos es el relacionado con la extracción de información útil del gran volumen de datos generados por los dispositivos que conforman los ecosistemas IoT.

Por otro lado, el manejo de grandes volúmenes de datos es un problema que ha sido ampliamente estudiado durante las dos últimas décadas. Este problema es conocido en el mundo como Big Data. A lo largo de estas dos últimas décadas, varias metodologías, tecnologías, plataformas, modelos, herramientas y librerías han sido desarrolladas como soluciones para enfrentar el desafío del manejo eficiente del Big Data. Este conjunto de habilitadores tecnológicos de Big Data han sido ampliamente utilizados por buscadores de páginas web en Internet, empresas de marketing y ventas, en redes sociales, entre otros [5]. Sin embargo, el uso de los habilitadores tecnológicos Big Data para resolver el problema del Big Data generado por los ecosistemas IoT es más complicado debido a las particulares características de los datos generados en ecosistemas IoT [7]. De esta manera, resulta nece-

sario un entendimiento de IoT como fuente de datos y de las metodologías y técnicas relacionadas con Big Data para aplicarlas satisfactoriamente en entornos IoT.

Este capítulo proporciona una breve introducción a IoT, describe algunas de las arquitecturas de referencia en IoT y analiza sus dominios de aplicación como punto de partida. Posteriormente, el capítulo se centra en el concepto de Big Data para entender el problema del manejo de grandes volúmenes de datos. Posteriormente, el capítulo describe las metodologías, tecnologías y plataformas involucradas en la gestión del Big Data; así como, analiza las arquitecturas de referencia para el diseño de soluciones de Big Data. Además, el capítulo describe las técnicas de minería de datos y técnicas de IA (*machine learning* y *deep learning*) para brindar un entendimiento sobre su uso en la extracción de conocimiento del Big Data. Además, el capítulo explica las tecnologías computacionales como *cloud computing*, *fog computing*, *edge computing* con el fin de entender los enfoques utilizados para la implementación de soluciones eficientes en el manejo del Big Data generado en ecosistemas IoT.

2.2 Visión de IoT

El concepto de IoT involucra la interconexión global de objetos por medio del uso de las tecnologías de Internet; así como la utilización de otras tecnologías tales como *Radio Frecuency Identification* (RFID), comunicación *Machine to Machine* (M2M), redes inalámbricas de sensores y actuadores (WSAN, por sus siglas en inglés, *Wireless Sensor and Actuator Network*), entre otras [19] [20]. La visión de IoT es permitir la interconexión entre el mundo físico y el virtual con el fin de habilitar una interacción en cualquier momento y en cualquier lugar [1]. Dicha interacción es posible a través del uso de tecnologías habilitadoras que permitan identificar fácilmente a los objetos IoT o dispositivos IoT, y comunicarse e interactuar con ellos. De esta manera, los pilares tecnológicos de la visión de IoT son las tecnologías de identificación, de redes de sensores y de comunicación inalámbrica [19], y se analizan a continuación:

- La tecnología de identificación RFID provee un identificador único a cada objeto a través de un *tag* RFID [19]. Cada *tag* RFID identifica de manera única a cada objeto. El uso de un lector RFID permite descubrir a los objetos que se encuentren en un área cercana e identificarlos [19]. Los sistemas RFID permiten la monitorización de los objetos, habilitando de esta manera la integración entre el mundo virtual y digital.

- Las redes de sensores están compuesta por varios nodos sensores y actuadores que se interconectan entre sí utilizando tecnologías de comunicaciones inalámbricas, formando una red. Estos nodos sensores y actuadores están en la capacidad de percibir su entorno, convertir en datos lo que perciben, transmitir a Internet estos datos y en algunos casos actuar dependiendo de los comandos que reciban desde Internet [20]. Para ello, los nodos sensores están equipados con limitadas capacidades de computación (microprocesadores, microcontroladores, etc.) y tecnologías de comunicaciones inalámbricas.
- Las tecnologías de comunicaciones inalámbricas permiten que los sensores se conecten entre sí. Los sensores utilizan las tecnologías como Bluetooth, ZigBee, *IPv6 over Low power Wireless Personal Area Network* (6LowPAN), *Wireless Fidelity* (WiFi), *Long Term Evolution* (LTE), entre otras, para habilitar una redes de sensores. Los sensores utilizan a un dispositivo conocido como *gateway* IoT que sirve como puente para conectarse a Internet.

Cada día, millones de objetos equipados con la conjunción de estas tecnologías son conectados a Internet [3]. Estos objetos son conocidos como objetos inteligentes o dispositivos IoT. Los dispositivos IoT presentan características limitadas computacionales, de consumo de energía, memoria y uso de red, por lo que requieren de capacidades para el almacenamiento y procesamiento de los datos que recolectan [21]. De esta manera, IoT requiere de otras tecnologías que permitan superar sus limitaciones con el fin de alcanzar los beneficios esperado. Los recursos provistos por una infraestructura de *cloud computing* proveen de las capacidades necesarias de los dispositivos IoT. En la sección 2.4 se analizará a *cloud computing* como habilitador tecnológico para superar las limitaciones de los dispositivos IoT.

2.2.1 Arquitecturas de referencia en IoT

Las arquitecturas IoT son modelos que sirven como guía en el diseño de sistemas de IoT. La Figura 2.1 muestra las arquitecturas de niveles típicas con las que se representa un sistema IoT. La arquitectura más simple tiene tres niveles: percepción, red y aplicación, Figura 2.1a [22].

- **Nivel de percepción:** compuesta por un conjunto de dispositivos electrónicos encargados de percibir los datos del mundo físico. Estos dispositivos pueden ser por ejemplo sensores, Sistema de Posicionamiento Global (GPS), etiquetas de identificación por radiofrecuencia RFID, cámaras, entre otros.

Figura 2.1: Arquitecturas IoT

- **Nivel de red:** encargada de proveer las funcionalidades de comunicación a través de protocolos y tecnologías de comunicación inalámbricas para que los dispositivos electrónicos puedan transmitir sus datos a Internet. Las tecnologías inalámbricas como Bluetooth, ZigBee, 6LowPAN, WiFi, LTE y protocolos de aplicación de bajo consumo de energía como *Message Queue Telemetry Transport* (MQTT) y *Constrained Application Protocol* (CoAP) son utilizadas por los dispositivos de la capa percepción. Típicamente un *gateway* IoT asegura la conectividad de los dispositivos IoT a Internet.
- **Nivel de aplicación:** encargada de gestionar los datos recolectados para presentar la información a los usuarios finales.

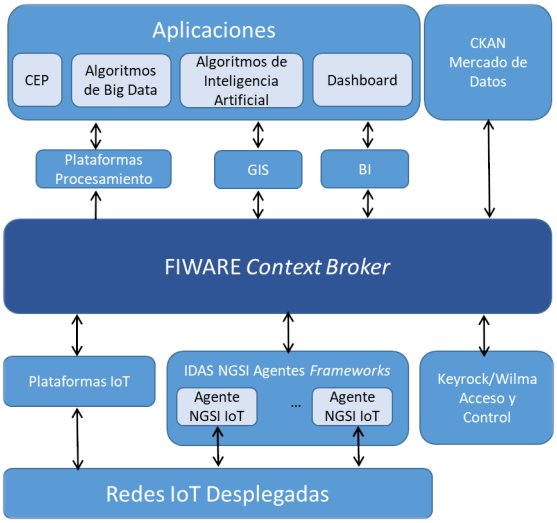
Las arquitecturas de 5 niveles fueron apareciendo con el desarrollo de IoT. Estas arquitecturas están basadas en *middleware* [23] y en servicios [19]. La arquitectura basada en *middleware* básicamente se subdividió el nivel de aplicación del modelo de 3 niveles en aplicación y negocio con el fin de gestionar de mejor manera los dispositivos y los datos de IoT, Figura 2.1b. La arquitectura basada en servicios se enfoca en proveer las funcionalidades necesarias para componer servicios que permitan desarrollar aplicaciones, Figura 2.1c [19].

Adicionalmente a estas arquitecturas de referencia, varias arquitecturas han sido propuestas para ser consideradas como referencias para el diseño de sistemas IoT en sus distintos ámbitos de aplicación. Así por ejemplo tenemos la arquitectura de referencia de FIWARE para entornos IoT inteligentes, la arquitectura de alto nivel propuesto por la AIOTI, la arquitectura de referencia para el IoT industrial (IIRA, por sus siglas en inglés, *Industrial Internet Reference Architecture*) propuesta por la *Industrial Internet Consortium* (IIC) o el modelo arquitectural de referencia de la industria 4.0

(RAMI 4.0, por sus siglas en inglés, *Reference Architectural Model Industrie*).

Arquitectura de referencia FIWARE

La arquitectura de referencia FIWARE está basada principalmente en la suite de aplicaciones, módulos y *Application Programming Interfaces* (APIs) de la plataforma de IoT FIWARE. La arquitectura está compuesta por tres niveles: i) interfaces a dispositivos IoT y otros sistemas, ii) gestión de la información de contexto y iii) el procesamiento, análisis y visualización del contexto, Figura 2.2.



Fuente: FIWARE, 2019 [24]

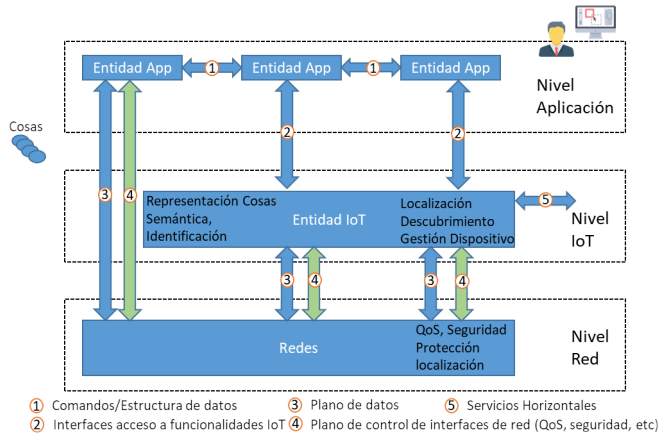
Figura 2.2: Arquitectura de Referencia FIWARE

El primer nivel permite la conexión a los dispositivos IoT a través de sus agentes IoT y a plataformas IoT. El segundo nivel tiene como base fundamental el módulo o *Generic Enabler* (GE) Orion Context Broker, el cual concentra los mensajes enviados por dispositivos IoT y los distribuye a otros GEs del tercer nivel. El tercer nivel corresponde a una serie de GEs modulares dedicados a tareas específicas con la información de contexto, tales como el procesamiento de datos a través del uso de Procesador de Eventos Complejos (CEP), algoritmos de Big Data y algoritmos de Inteligencia artificial, o la visualización de datos *Geographic Information System* (GIS) y la inteligencia de negocios (BI, por sus siglas en inglés *Business Intelligence*). Además, el uso de plataformas de datos abiertos como *Comprehensive Knowledge Archive Network* (CKAN) y mercados de

datos pueden ser añadidos para extraer información del GE Orion Context Broker. La comunicación de mensajes entre niveles es realizada a través de las APIs *Next Generation Service Interface* (NGSI). La flexibilidad de la arquitectura permite que pueda ser empleada en varios ecosistemas de IoT de manera transparente [24].

Arquitectura de referencia AIOTI HLA

Al igual que la arquitectura de FIWARE, la arquitectura de referencia AIOTI HLA está compuesta por 3 niveles desde el punto de vista funcional: i) nivel de red, ii) nivel de IoT y iii) nivel aplicación. La Figura 2.3 muestra el modelo funcional de la arquitectura.



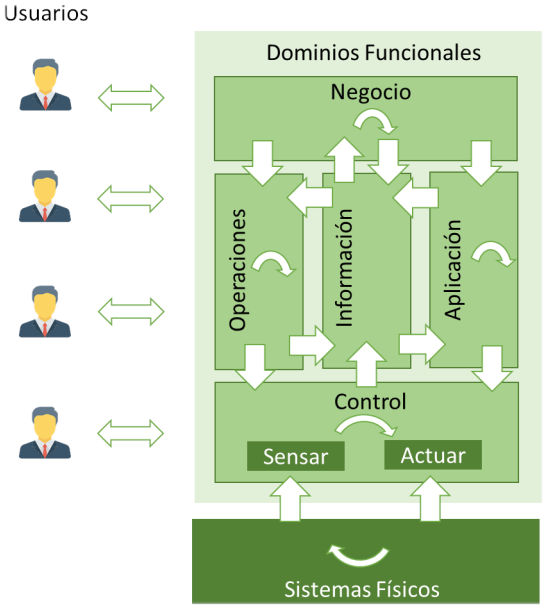
Fuente: AIOTI, 2018 [25]

Figura 2.3: Vista funcional Arquitectura AIOTI HLA

El nivel de red proporciona conectividad de corto y largo alcance y reenvío de datos entre entidades; así como, proporciona servicios de control como localización, QoS, seguridad, protección o determinismo. El nivel de IoT está compuesto por la entidad IoT que expone las funciones IoT a las entidades del nivel de aplicación. Las funcionalidades IoT de la entidad IoT incluyen el almacenamiento de datos, compartición de datos, bróker de comunicaciones, análisis de datos, descubridor semántico, etc. Las entidades IoT utilizan el nivel de red para recibir datos y enviar datos de control a los dispositivos IoT. Finalmente, el nivel de aplicación está compuesto por entidades aplicación o entidades *app* que implementan la lógica de las aplicaciones IoT. La entidad aplicación puede residir en dispositivos, *gateways* IoT o servidores [25].

Arquitectura de referencia IIRA

La arquitectura de referencia IIRA está diseñada para servir de modelo en el desarrollo de sistemas IoT en la industria. La arquitectura fue propuesta en 2015 por el IIC siguiendo la norma ISO/IEC/IEEE 42010:2011. La arquitectura IIRA se describe desde cuatro puntos de vista: i) punto de vista del negocio, ii) punto de vista de uso, iii) punto de vista funcional y iv) punto de vista de implementación. Desde el punto de vista funcional la arquitectura está compuesta por 5 dominios: control, operacional, información, aplicación y negocio. La Figura 2.4 muestra la vista funcional de la arquitectura IIRA.



Fuente: IIRA, 2017 [26]

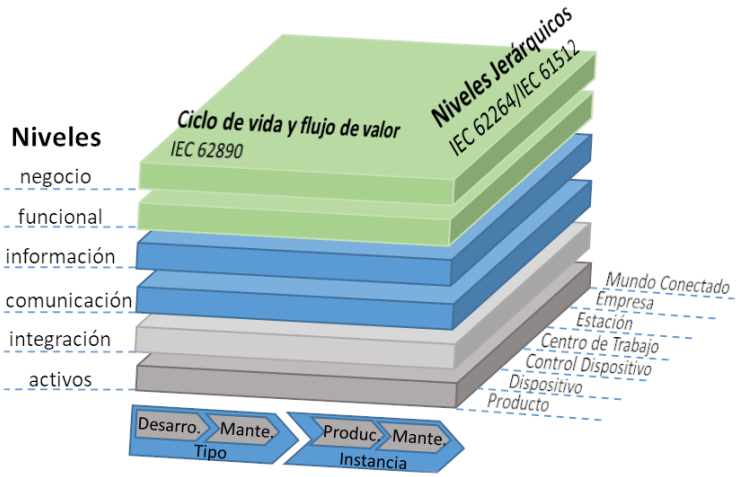
Figura 2.4: Vista funcional arquitectura IIRA

Los flujos de datos y de control son intercambiados entre estos dominios funcionales. El dominio de control está compuesto por los sensores y actuadores distribuidos geográficamente cerca a los dispositivos físicos que forman parte del sistema industrial de control y automatización. El dominio de control se encarga de la comunicación de sensores y actuadores con el resto de los dominios de la arquitectura. El dominio de operación se encarga de la gestión y operaciones del dominio de control. Las funciones principales que provee el dominio de operación están destinadas al monitoreo y diagnóstico, mantenimiento predictivo, gestión de activos, optimización y pronósticos.

El dominio de información se encarga de la gestión y procesamiento de los datos. Las funciones principales son de almacenar persistentemente los datos, pre-procesamiento de datos (limpieza y filtrado), procesamiento en tiempo real y el análisis de los datos recolectados por el dominio de control. El dominio de aplicación implementa la lógica de la aplicación. Desde este dominio se gestionan las reglas de las aplicaciones y las APIs e Interfaces de Usuario IU para interactuar con la aplicación. Finalmente, el dominio de negocio implementa la funcionalidad lógica del negocio. Este dominio permite la integración con sistemas para la planificación de recursos de la empresa (ERP, por sus siglas en inglés, *Enterprise Resource Planning*), gestión del ciclo de vida del producto (PLM, por sus siglas en inglés, *Product Lifecycle Management*), gestión de la relación con clientes (CRM, por sus siglas en inglés, *Customer Relationship Management*), entre otros [26].

Arquitectura de referencia RAMI 4.0

La arquitectura de referencia RAMI 4.0 fue presentada por el proyecto del gobierno alemán *Platform Industrie* 4.0. Esta arquitectura describe aspectos importantes de las tecnologías que envuelven la Industria 4.0 para proveer un entendimiento de los estándares y casos de uso [27]. La arquitectura de referencia RAMI 4.0 presenta una estructura tridimensional compuesta por las dimensiones de jerarquía, ciclo de vida y flujo de valor, y niveles [28]. La Figura 2.5 muestra la estructura tridimensional de la arquitectura RAMI 4.0.



Fuente: Schweichhart K., 2016 [29]

Figura 2.5: Estructura tridimensional de la arquitectura RAMI 4.0

La dimensión de jerarquía representa las funcionalidades de los componentes de un sistema en la industria 4.0. Esta dimensión se basa en la estructura por niveles jerárquicos de la norma IEC 62264/IEC 61512. La dimensión incluye los niveles jerárquicos de producto, dispositivo, control de dispositivo, estación, centros de trabajo, empresa, y mundo conectado [30].

La dimensión ciclo de vida y flujo de valor representa el ciclo de vida de los productos y servicios en tipo e instancia. Esta dimensión está basada en la norma IEC 62890 [27]. La dimensión considera el desarrollo del producto o tipo y el mantenimiento de uso, así como la producción y mantenimiento de la instancia del producto o servicio.

La dimensión de niveles representa las propiedades tecnológicas de los componentes del sistema. Esta dimensión está compuesta por seis niveles: activos, integración, comunicación, información, funcional y de negocio. El nivel de activos contiene los objetos del mundo físico tales como máquinas, sensores y documentos. El nivel de integración es la interfaz entre el mundo real y el digital. El nivel de comunicación gestiona las comunicaciones de los activos, el formato de los datos, protocolos e interconexión de los activos. El nivel de comunicación recomienda el uso del protocolo de comunicación *Open Platform Communications Unified Architecture* (OPC UA) para habilitar el intercambio de datos. El nivel de información gestiona la información generada por los activos. El nivel funcional representa un ambiente de ejecución de servicios y aplicaciones. El nivel de negocio abstrae los modelos de negocio y aspectos legales [30].

La arquitectura RAMI 4.0 requiere una interacción con tecnologías habilitadoras como *cloud computing* y Big Data para gestionar el gran volumen de datos generados por los dispositivos físicos del nivel de activos [31]. La infraestructura *cloud computing* provee los recursos necesarios para almacenar, y procesar lo datos, mientras que las plataformas y herramientas de Big Data proveen un manejo eficiente del procesamiento y almacenamiento de los datos para extraer valor para las empresas. La industria puede beneficiarse de las dos tecnologías para implementar un nivel de información altamente eficiente y seguro para sus aplicaciones [31].

2.2.2 Dominios de aplicación de IoT

IoT presenta una amplia gama de posibles dominios de aplicación, por ejemplo, en salud, energía, industria, transporte y logística, entre otros, como muestra la Figura 2.6. Estos dominios de aplicación forman ecosistemas tecnológicos orientados a recolectar información vital para mejorar la toma de decisiones, Los dominios de IoT pueden cruzarse entre sí formando un dominio más amplio de aplicación como es el caso de las ciudades inteli-

gentes. A continuación describimos brevemente los dominios de aplicación de IoT que están enmarcados en el contexto de esta tesis.

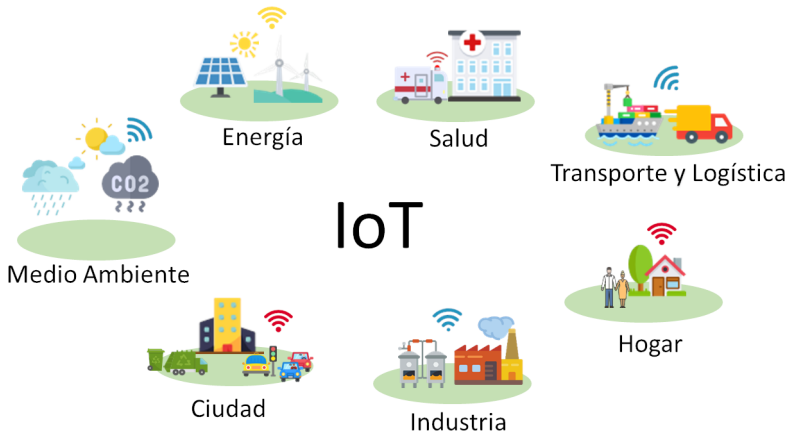


Figura 2.6: Dominios de aplicación de IoT

Dominio de la salud

El dominio de la salud es una de las aplicaciones más prometedoras de IoT y Big Data. La aplicación de IoT en la salud proporciona la visión conocida como salud inteligente (*Smart Health*), la misma que engloba la digitalización de los servicios sanitarios, gestión de récords electrónicos, servicios de asistencia y monitorización remota, y los dispositivos médicos inteligentes [32]. Los dispositivos médicos inteligentes están destinados a cubrir los requerimientos de auto-control de la salud y prevención de enfermedades. Estos dispositivos llevan consigo sensores ligeros y no invasivos que permiten que las personas desarrollen sus actividades normalmente sin descuidar su salud. En los últimos años, los dispositivos electrónicos *wearables* (ropa, relojes inteligentes, muñequeras, entre otros) se han convertido en los dispositivos más populares destinados a la monitorización de la salud [33]. Estos dispositivos están destinados a medir constantes vitales, parámetros ambientales y de bienestar. Cada día los dispositivos médicos inteligentes generan grandes volúmenes de datos que aún no han sido explotados para extraer beneficios [2]. El manejo de este gran volumen de datos a través de las metodologías y técnicas de Big Data es necesario con el fin de extraer beneficios que puedan favorecer al bienestar de las personas y a las entidades y personal involucrado en las prestaciones de servicios de sanidad.

La gestión del gran volumen de datos generados en el caso del cuidado de la salud proporciona varios beneficios. El análisis de los datos de IoT posibilita desplegar una amplia variedad de servicios relacionados a la monitorización

de la salud de pacientes. Los servicios no solo pueden monitorizar el estado de los signos vitales del paciente remotamente, sino también pueden predecir condiciones de salud (ataques cardíacos, infecciones, tipos de cáncer, entre otros) [34]. Además, el análisis del Big Data generado por IoT reduce sustancialmente los costos involucrados en el diagnóstico, tratamiento y monitorización de los pacientes [35] [36]. Por ejemplo, la monitorización remota del estado de los pacientes (latidos del corazón, niveles de glucosa, niveles de oxígeno, presión arterial, etc.) y detección de anormalidades reduce los costos inherentes al uso de la infraestructura de un hospital para estas tareas. Los beneficios del Big Data en IoT no solo están destinadas a mejorar la calidad de vida de las personas, sino también a mejorar las herramientas de diagnóstico, proporcionar mejores tratamientos y control a los pacientes y detectar patrones de enfermedades y trazas de brotes de enfermedades [37] [38].

Dominio del transporte

El transporte es el sector de la industria con mayor impacto económico en el comercio mundial. La aplicación de IoT en el transporte permite el desarrollo de sistemas inteligentes de transporte con el objetivo de incrementar la eficiencia de las operaciones involucradas en los servicios de transporte [39]. A lo largo de las últimas décadas, los sistemas de transporte han ido incorporando tecnologías que le permitan cumplir con este objetivo. Tecnologías y sistemas como GPS, *Automatic Identification System* (AIS), WSAN, entre otras, han sido instaladas en autobuses, camiones, barcos, o bicicletas provocado cambios en los sistemas de transporte [40]; así como el uso de tecnología satelital para el seguimiento de barcos [41].

Los sistemas de transporte inteligente no solo engloban a los medios de transporte sino también involucra a la infraestructura de apoyo para brindar el servicio como por ejemplo puertos marítimos, aeropuertos, estaciones de trenes, vías de trenes, carreteras, etc. Actualmente, los *Cyber Physical Systems* (CPSs) están destinados a la automatización y digitalización de los procesos en las terminales de transporte. Los CPSs conectan los dispositivos físicos con el mundo virtual. Para ello, los CPSs se basan en arquitecturas de referencia como (RAMI 4.0) e (IIRA) [42]. Por otro lado, los sistemas IoT permiten la conexión de cualquier equipo de una terminal de transporte a Internet. Los sistemas IoT están orientados a la monitorización de los equipos involucrados en las operaciones de los sistemas de transporte, como por ejemplo monitorización de grúas portuarias, rieles, contenedores, alumbrado eléctrico, entre otros). Un gran ejemplo de sistemas de transporte inteligentes basados en IoT son los desplegados en la infraestructura por-

tuaria en puertos europeos importantes como Valencia [43], Hamburgo [44], Rotterdam [45] o Las Palmas [46].

Los sistemas de transporte inteligente pueden tomar ventaja de las metodologías y técnicas de Big Data e IA para incrementar la eficiencia en el servicio de transporte. De esta manera, el gran volumen de datos generados por los sistemas de transporte inteligente puede ser gestionado de forma eficiente. Además, el manejo eficiente del gran volumen de datos permitirá extraer información para mejorar la toma de decisiones en la planificación del servicio de transporte [40]. Esto es de vital importancia con el fin de reducir el tráfico en la red de transporte. La IA en apoyo con las metodologías y técnicas de Big Data permitirá predecir patrones en el tráfico provocado por el transporte o los tiempos de llegada y salida de un medio de transporte; así como identificar remotamente actividades anómalas [41]. Esta información pueden ser utilizada para generar un alto beneficio económico e incluso ambiental [39]. Además, la explotación del gran volumen de datos permitirá mejorar los niveles de seguridad en los servicios de transporte. La detección y predicción de un fallo en la infraestructura de operaciones de los servicios de transporte reducirá los tiempos de respuesta en caso de una emergencia, incluso llegar a anticiparse a una situación de emergencia [40]. En la actualidad, proyectos como *Pixel Port* [47] y *Transforming Transport* [39] persiguen hacer efectivo el objetivo de los sistemas de transporte inteligente a través del uso de metodologías y técnicas de Big Data y tecnologías habilitadoras de IoT.

2.3 Big Data

El gran volumen de datos ha afectado al normal funcionamiento de los sistemas tradicionales desde hace un par de décadas. Varias soluciones fueron propuestas destinadas a resolver retos puntuales sobre el problema de manejo del gran volumen de datos con el pasar de los años. Estas soluciones se agrupan en lo que hoy se conoce como las metodologías, técnicas y tecnologías habilitadoras del Big Data. A lo largo de esta sección se presentan las definiciones de Big Data y sus características, se describen las metodologías, técnicas y tecnologías habilitadoras del Big Data; así como, se describen las arquitecturas que sirven de referencia para el diseño de sistemas para el manejo de grandes volúmenes de datos.

2.3.1 Definiciones de Big Data

Big Data es un concepto abstracto, por ello se presentan a continuación algunas definiciones proporcionadas por la academia y la industria que permitirán entender este concepto:

- Unión Internacional de Telecomunicaciones (ITU-T) Y.3600 [48]: "*Un paradigma para habilitar la colección, almacenamiento, gestión, análisis y visualización, potencialmente con limitaciones de tiempo real, de un extenso conjunto de datos con características heterogéneas.*".

- Instituto Nacional de Estándares y Tecnologías (NIST) [49]: "*Big Data consiste de un extenso conjunto de datos principalmente con características de volumen, variedad, velocidad y variabilidad que requieren una arquitectura escalable para un almacenamiento, manipulación y análisis eficiente.*".

- Gartner [50]: "*Big Data es un activo de información de gran volumen, gran velocidad, gran variedad que exige formas rentables e innovadoras para una mejor comprensión y toma de decisiones.*".

- *International Data Corporation* (IDC) [51]: "*Big Data describe una nueva generación de tecnologías y arquitecturas que son diseñadas para económicamente extraer valor de grandes volúmenes de una amplia variedad de datos y que requiere una alta velocidad de captura, descubrimiento y/o análisis.*".

De las diversas definiciones de Big Data que se encuentran en la literatura actual podemos extraer que las características de los datos definen si el conjunto de datos puede ser considerado como Big Data. Además, la forma en como son almacenados, procesados y analizados los datos impacta en la eficiencia de la gestión del Big Data [5]. Dada la importancia de estos puntos para un entendimiento global de lo que es Big Data y las metodologías y técnicas que involucran, las siguientes subsecciones se analizarán las características del Big Data, el ciclo de vida del Big Data, el procesamiento, almacenamiento y análisis de Big Data.

2.3.2 Características del Big Data

Las características que definen el Big Data son Volumen, Velocidad y Variedad. Estas características también son conocidas coloquialmente como las 3Vs. En la actualidad, existen muchas otras características que se han sumado a estas como por ejemplo, valor, veracidad, viabilidad, visualización, entre otras, pero estas son consideradas como características resultantes de un tratamiento previo de los datos [52]. A continuación se describen cada una de las 3Vs:

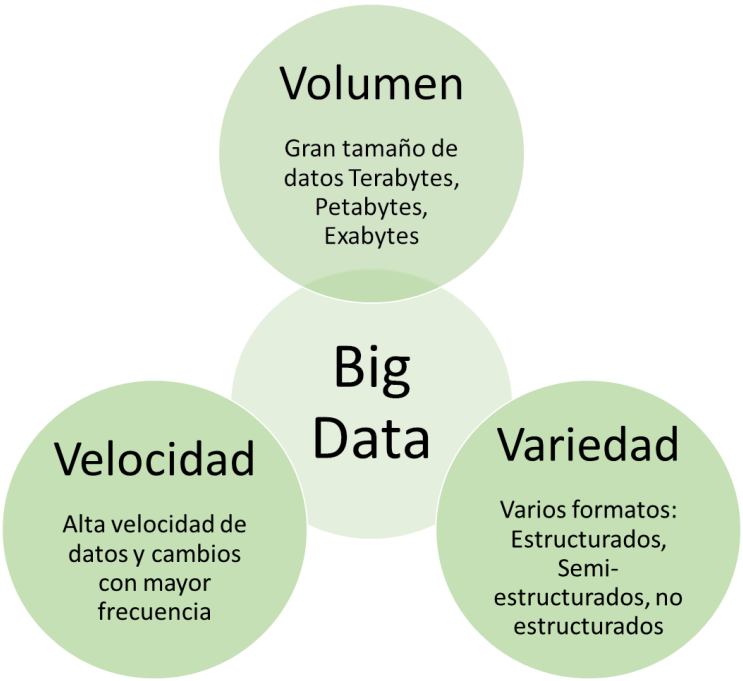


Figura 2.7: 3V's de las características de Big Data

- **Volumen**

  El volumen indica la gran cantidad de datos recolectada de diferentes fuentes de datos, es decir, el volumen se refiere a la magnitud de los datos [6]. Por lo general la magnitud de los conjuntos de datos se acercar a los *Terabyte* (TB), *Petabyte* (PB), incluso a los EB. En 2017 se generaron 3.7 EB de datos por parte de IoT [3]. Estas magnitudes no pueden ser gestionados por sistemas de base de datos tradicionales [53]. Además, la recolección de estas magnitudes de datos puede llegar a ser muy costoso [6].

- **Velocidad**

  La velocidad implica que los datos son generados a con muy alta frecuencia y estos cambian constantemente [6]. Un ejemplo de esta característica la encontramos en los millones de transacciones por hora realizadas durante las compras *online* en las empresas de ventas al por menor como Amazon o Wal-mart [53]. Este nivel de frecuencia en la generación de datos requiere de perspectivas eficientes y escalables para el procesamiento en tiempo real.

- **Variedad**

  La variedad indica los diferentes tipos de datos recolectados, pueden ser texto, imágenes, vídeos, audios, o *logs*; todos ellos en un formato estructurado, semi-estructurado y no estructurado [53]. Esta variedad de tipos de datos y formatos es generado por una amplia variedad de fuentes tales como páginas web, redes sociales, IoT, o transacciones. Los datos en formato estructurado son almacenados en tablas o en sistemas de base de datos relacionales conocidas como *Database Management System* (DBMS) [53]. Los datos en formato semi-estructurado no siguen estrictamente un estándar, un ejemplo de esto son los formatos *Java Script Object Notation* (JSON) y *Extensible Markup Language* (XML) utilizados en el intercambio de información a nivel de Internet (páginas web, aplicaciones, IoT, entre otros.) [53]. Finalmente, los datos no estructurados son aquellos que no siguen un formato específico.

Existen una amplia variedad de fuentes que generan Big Data con estas características pero sin lugar a dudas los datos generados por dispositivos IoT tiene el gran potencial para convertirse en la fuente de principal de Big Data [54]. Esto debido principalmente a que IoT tiene un amplio abanico de dominios de aplicación. Además, IoT es una fuente de datos que ha presentado un enorme crecimiento e impacto en la sociedad en los últimos años [54]. Sin embargo, IoT es compleja al resto de fuentes de datos por su naturaleza heterogénea. Esta heterogeneidad está enmarcada tanto en los dispositivos y plataformas IoT como en sus dominios de aplicación [55]. Por ello, es importante analizar las características particulares del Big Data generado por los entornos IoT.

Características particulares del Big Data generado por IoT

IoT como fuente de Big Data no solo presenta las características de volumen, velocidad y variedad, sino también características propias como la heterogeneidad, correlación espacio-tiempo, alto ruido y sus datos generados a gran escala. A continuación analizamos cada una de estas características:

- **Volumen:** los datos generados por IoT provienen de múltiples sensores y otros dispositivos inteligentes que tiene una alta capacidad para generar grandes cantidades de datos. Tal es así que se estima que los dispositivos IoT generaran 25 EB por mes en el 2022 [3]. El gran volumen de los datos de IoT hace impensable el uso de base de datos relacionales por sus limitaciones de escalabilidad y robustez [56].
- **Velocidad:** las medidas tomadas por los sensores y dispositivos inteligentes son realizadas con una muy alta frecuencia por lo que un gran volumen de datos es generado en periodos muy cortos de tiempo. Los sistemas actuales manejan esta característica de velocidad por medio de CEPs que son capaces de analizar los eventos y encontrar patrones en tiempo real [57]. Sin embargo, los CEPs no son escalables y en sistemas a gran escala no son eficientes [58], por lo que es necesario considera el uso de sistemas de tiempo real distribuidos para el despliegue de aplicaciones en gran escala.
- **Variedad:** los datos de IoT en muchos casos no presentan una estructura definida (datos no estructurados), y en algunos casos presentan una semiestructura (datos semiestructurados). Los datos semiestructurados de IoT no siguen una estructura implícita pero si presenta una organización interna que facilita el procesamiento y análisis [5]. Los sensores suele utilizar el formato JSON como formato semiestructurado, mientras que los dispositivos de vídeo, audio no presentan un formato especifico para estructurar sus datos recolectados.
- **Heterogeneidad:** la heterogeneidad de los dispositivos IoT y plataformas IoT ocasiona una falta de interoperabilidad de sistemas IoT. Principalmente, esto dificulta la integración de datos de distintas plataformas. Además, los datos de IoT están aislados en silos verticales de información con formatos e interfaces especificas. En algunos casos, los datos son almacenados en diferentes almacenes de datos de IoT aislados. La integración de todas estas fuentes diversas de IoT (dispositivos IoT, plataformas IoT, datos abiertos, y almacenes de datos IoT) permite generar mejores y más precisos análisis de datos para una correcta toma de decisiones.

- **Correlación espacio-tiempo:** los datos de IoT son del tipo series de tiempo. Esta característica proporciona información importante cuando es manejada adecuadamente. Así mismo, los datos IoT son del tipo geoespacial, es decir proporcionan información sobre la localización de la medida tomada por los sensores. Estas características interaccionan entre sí brindando una información espacio-temporal que debe ser considerada en el manejo de Big Data, principalmente en la visualización de los datos [59].
- **Alto ruido:** los datos de IoT son altamente redundantes y pueden contener ruido. Especialmente, los datos que son generados por sensores configurados para censar en períodos muy cortos de tiempo. Este exceso de datos puede afectar los análisis de datos, así como afectan directamente a los recursos computacionales disponibles [7].
- **Datos en gran escala:** gran cantidad de dispositivos IoT pueden ser distribuidos en varios lugares geográficamente. Este despliegue masivo de dispositivos IoT dificultan los procesos de adquisición de datos y procesamiento en tiempo real [5].

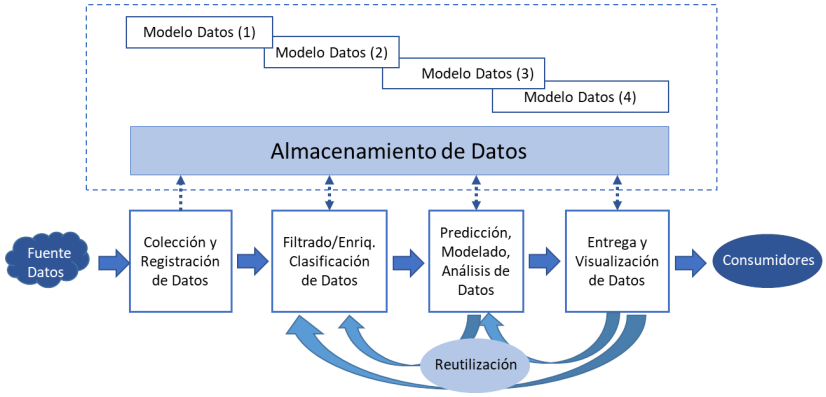
2.3.3 Ciclo de vida del Big Data

La gestión del ciclo de vida del Big Data (BDLM, por sus siglas en inglés, *Big Data Lifecycle Management*) es la base fundamental en el entendimiento de Big Data. Este define las transformaciones que sufren los datos durante el manejo de las características de Big Data (3Vs) con el fin de extraer la mayor cantidad de información útil. Los modelos de ciclo de vida de datos están compuestos por una serie de operadores encargados de realizar las transformaciones de los datos. En la actualidad, el modelo de BDLM propuesto por Demchenko et al. en [52] es uno de los modelos de mayor aceptación para el manejo de las características de Big Data [8]. Este modelo tiene la ventaja de ser genérico, por lo que se puede adaptar fácilmente a las necesidades específicas de las aplicaciones. Otros modelos son más específicos y solo son aprovechados para el escenario particular para el cual fueron diseñados [60].

El modelo BDLM tiene la idea principal de preservar los datos con cada transformación. De esta manera, el modelo permite una reusabilidad de los datos para extraer el máximo provecho de ellos [52]. La Figura 2.8 muestra el ciclo de vida de los datos del modelo BDLM propuesto por Demchenko et al. en [52]. El modelo presenta los operadores de colección y registración de datos, filtrado/enriquecimiento y clasificación de datos, análisis, modelado y predicción de datos, y entrega y visualización de datos.

Durante las transformaciones realizadas por estos operadores, los datos son almacenados siguiendo su respectivo modelo de datos.

Del modelo BDLM se identifica como bloques importantes para la gestión del Big Data a las tareas de almacenamiento, procesamiento y análisis del Big Data. El almacenamiento del Big Data está presente durante todas las etapas del ciclo de vida de los datos en el modelo BDLM por lo que en cada uno de estas etapas se tienen requerimientos específicos. Por otro lado, el procesamiento de los datos permite realizar las tareas de filtrado, enriquecimiento y clasificación de datos con el fin de preparar los datos previo a un análisis. Finalmente, el análisis de datos está relacionado directamente con la extracción de la información a través de modelos que servirán para predecir estados que faciliten la toma de decisiones. En las siguientes subsecciones se analizan los conceptos, metodologías, técnicas y plataformas empleadas para el almacenamiento, procesamiento y análisis del Big Data.



Fuente: Demchenko et al.,2014 [52]

Figura 2.8: Ciclo de vida de los del modelo BDLM

2.3.4 Almacenamiento de Big Data

El almacenamiento de Big Data depende principalmente del formato de los datos. Los datos estructurados son almacenados en sistemas de administración de base de datos relacionales. Estos sistemas presentan un lenguaje de programación utilizado para la gestión y extracción de información llamado *Structured Query Language* (SQL) [6]. Sin embargo, con la proliferación de nuevas fuentes de datos tales como texto, páginas web, videos, audio, entre otros, los DBMS resultaron ineficientes para manejar estos formatos de datos [5].

Los datos semi-estructurados y no estructurado son almacenados en sistemas de base de datos conocidos como NoSQL, llamados así por no soportar el lenguaje de programación SQL. Los sistemas NoSQL pueden ser clasificados por el modelo de almacenamiento que presentan para soportar los formatos de datos semiestructurados y no estructurados. De esta manera, los sistemas NoSQL se clasifican en bases de datos orientados a documentos, orientados a columnas, basados en grafos y clave-valor.

- Las bases de datos orientadas a documentos están diseñadas para almacenar colecciones de documentos que pueden tener formatos complejos de datos. Los estándares populares para almacenamiento son JSON y XML [6] [7].
- Las bases de datos orientadas a columnas permiten almacenar en columnas la información, en lugar de filas como se realizaba en los sistemas tradicionales [6].
- Las bases de datos basadas en grafos utiliza nodos para representar objetos y líneas para representar relaciones, utilizado mayormente en aplicaciones de redes sociales [7].
- Las bases de datos clave-valor están diseñadas para almacenar pares de datos clave-valor. Por su característica de una única clave son deseables para almacenamiento distribuido [5] [6].

Además de estas tecnologías de base de datos, los sistemas distribuidos de archivos nacen como una alternativa escalable y redundante para almacenar datos. El sistema de archivos distribuido de Hadoop (HDFS, por sus siglas en inglés, *Hadoop Distributed File System*) es el mayormente utilizado en Big Data. HDFS está basado en *Google File System* (GFS), propiedad de Google, y permite replicar datos en un clúster de servidores proporcionando redundancia y escalabilidad para el soporte de gran cantidad de archivos [6]. La Tabla 2.1 recopila en resumen los sistemas NoSQL mayormente utilizados en el diseño de sistemas con soporte de Big Data.

Los sistemas de bases de datos NoSQL han demostrado ser los adecuados para manejar el gran volumen de datos generados por IoT. Los sistemas MongoDB, Apache Cassandra, CouchDB, y Apache HBase son los que más han sido empleados para el diseño de sistemas de IoT [61] [62] [63]. Estos sistemas suelen ser considerados como sistemas complementarios en el diseño de *middleware* IoT. Principalmente, estos sistemas son empleados para almacenar los datos IoT semiestructurados, por ejemplo datos en formato JSON (base de datos basados en documentos) o datos que son transformados de JSON a tablas (base de datos basado en columnas). Por otro

Tabla 2.1: Sistemas de base de datos NoSQL

Nombre Sistema	Modelo de datos	Tipo de Almacenamiento
Dynamo	Clave-valor	Memoria RAM
Voldemort	Clave-valor	Memoria RAM
Redis	Clave-valor	Memoria RAM
HBase	Columnas	HDFS
Cassandra	Columnas	Disco duros HDD, SSD
BigTable	Columnas	GFS
MongoDB	Documentos	Disco duros HDD, SSD
SimpleDB	Documentos	Disco duros HDD, SSD
CouchDB	Documentos	Disco duros HDD, SSD

Hard Disk Drive (HDD), *Solid State Drive* (SSD), *Random Access Memory* (RAM).

lado, las pruebas de rendimiento han demostrado ampliamente que Apache Cassandra es adecuado para almacenar los datos en tiempo real, puesto que este ofrece alto rendimiento, baja latencia de lectura y escalabilidad lineal [64]. Además, Apache Cassandra provee el lenguaje de consultas *Cassandra Query Language* (CQL) para interactuar con los datos a través de la línea de comandos.

2.3.5 Procesamiento de Big Data

El procesamiento de Big Data se clasifica en dos tipos: procesamiento tipo *batch* y en tiempo real. El procesamiento *batch* realiza un procesamiento de grandes conjuntos de datos sin la intervención de un usuario a través de la ejecución de trabajos. En este caso, el usuario programa las tareas a realizarse y estas se ejecutaran sin su supervisión durante varias horas. El procesamiento *batch* de datos está destinado para aplicaciones que requieren de análisis estadísticos y sin importancia del tiempo de respuesta [56].

El modelo de programación *MapReduce* es el modelo de facto usado para el procesamiento *batch* en aplicaciones de Big Data [65]. Este modelo se divide en dos partes, *map* y *reduce*, ambos son desarrollados por el programador. La función *map* procesa los datos ingresados y genera pares clave-valor, luego se combina los valores con su misma clave generando una clave con muchos valores. La función *reduce* toma estas claves con sus respectivos valores y los reduce generando un grupo de valores más pequeño para cada clave [6] [5]. La Figura 2.9 muestra como ejemplo la estructura del modelo de programación *MapReduce*.

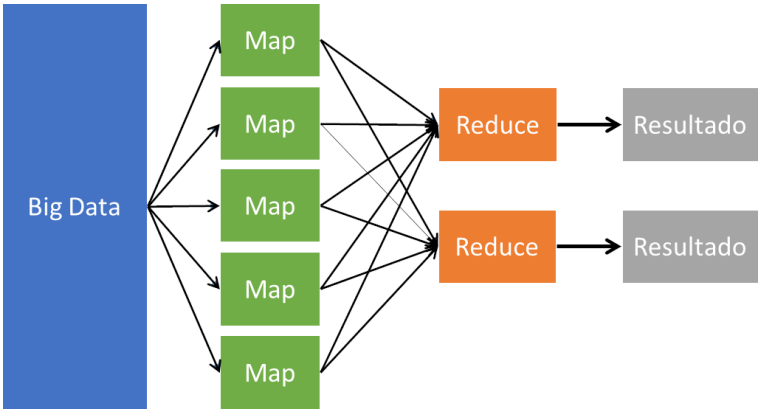


Figura 2.9: Modelo de programación *MapReduce*

Por otro lado, el procesamiento en tiempo real realiza un tratamiento de un flujo continuo de datos con poco retraso [66]. El procesamiento en tiempo real está destinado a aplicaciones donde el tiempo de respuesta es de alta prioridad y donde los datos almacenados cambian constantemente [56]. Este tipo de procesamiento puede realizarse de dos formas: utilizando operadores distribuidos o utilizando procesamiento *micro-batch*.

El procesamiento usando operadores distribuidos redirige el flujo de datos a través de varios operadores distribuidos en distintos *host* [66]. El modelo de programación que sigue este tipo de procesamiento es el modelo grafo acíclico dirigido (DAG, por sus siglas en inglés, *Directed Acyclic Graph*). El grafo acíclico dirigido está compuesto por elementos de procesamiento (PE, por sus siglas en inglés, *Processing Element*) interconectados entre sí para transmitir eventos. Los PEs esperan la llegada de eventos, los procesan, y emiten los resultados a otros PEs [67]. La Figura 2.10 muestra a manera de ejemplo la estructura de un DAG.

El procesamiento usando *micro-batch* también se lo conoce como procesamiento casi en tiempo real debido a que la ejecución de *micro-batch* suele provocar retardos en el tiempo de ejecución. Sin embargo, el procesamiento usando *micro-batch* presenta mejor escalabilidad y tolerancia a fallos [66]. Este tipo de procesamiento realiza una discretización del flujo de datos a través de una ventana temporal para posteriormente realizar la ejecución del procesamiento *batch* sobre la pequeña cantidad de datos de la ventana temporal.

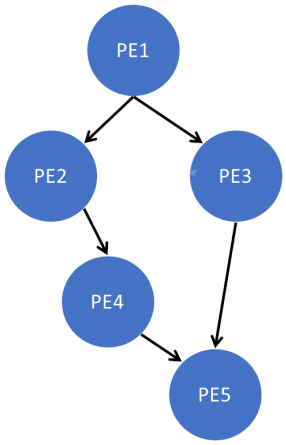


Figura 2.10: Ejemplo del modelo de programación DAG

Varias plataformas han sido desarrolladas para soportar los modelos de programación mencionados facilitando el desarrollo de aplicaciones. Las plataformas más populares para el procesamiento de Big Data se detallan a continuación:

- **Hadoop:** es la más popular de todas las plataformas de Big Data [5]. El ambiente Hadoop está compuesto principalmente por 4 módulos: HDFS, *Yet Another Resource Negotiator* (YARN), *MapReduce* y utilidades comunes. HDFS es el sistema de archivos distribuidos que utiliza Hadoop. YARN es un *framework* utilizado para agendar trabajos y gestionar los recursos del clúster. *MapReduce* es un *framework* que permite la implementación de aplicaciones usando el modelo de programación *MapReduce*. Y las utilidades comunes son módulos complementarios que permiten a desarrolladores crear aplicaciones con más facilidad. Los módulos complementarios proporcionan algunas facilidades como una base de datos distribuida (Apache HBase), un lenguaje de alto nivel para la ejecución de scripts y consultas tipo SQL (Apache Pig Latin, Apache Hive [68]), herramienta de gestión y monitorización del ecosistema Hadoop (Apache Ambari [69]), sincronización y gestión de elementos de un clúster (Apache Zookeeper [70]), minería de datos y algoritmos de *machine learning* (Apache Mahout [71]), entre otros [56]. La flexibilidad y modularidad de Hadoop facilita la integración con otros *frameworks* a través de sus APIs.
- **Apache Spark:** fue creado para solventar la limitaciones de velocidad de Hadoop en el procesamiento por medio del uso de un almacenamiento en memoria durante la ejecución del proceso *MapReduce*. Apa-

che Spark basa su funcionamiento en dos características principales, DAG y conjunto de datos distribuido resistente (RDD, por sus siglas en inglés, *Resilient Distributed Dataset*) [72]. DAG permite crear un flujo de datos acíclico, y se basa en la creación de varios nodos que son etapas para el procesamiento del grafo. Los resultados obtenidos de cada etapa del flujo de grafos es almacenado en memoria, con esto se introdujo la nueva abstracción RDD. Apache Spark puede trabajar sobre el ambiente de Hadoop, es decir puede utilizar HDFS o YARN, incluso podría trabajar por su cuenta. Además, Apache Spark implementa 4 librerías importantes para brindar soporte al desarrollo de aplicaciones, SparkSQL, Spark Streaming, MLlib y GraphX. SparkSQL es un módulo que permite trabajar con datos estructurados al estilo SQL, y tiene la flexibilidad de utilizar lenguajes de programación como Java, Python y R. Spark Streaming es el módulo que facilita la creación de aplicaciones que necesiten un procesamiento en tiempo real utilizando el enfoque de procesamiento *micro-batch*. MLlib es la librería que brinda *machine learning*, e implementa algunos algoritmos de *machine learning* importantes. GraphX es una API para ofrecer el soporte de grafos y el uso de la computación paralela [73] [72].

- **Apache Flink:** es un *framework* para procesamiento distribuido de flujo de datos y sigue el modelo de programación DAG. Flink no solo trabaja con flujo de datos, también permite el procesamiento *batch*. Para ello, Flink proporciona dos APIs, DataStream API para el procesamiento en tiempo real y DataSet API para procesamiento *batch*. Además, Flink brinda alto rendimiento, baja latencia, y escalabilidad. Flink puede ser instalado sobre YARN o Mesos. Además, Flink brinda librerías para el soporte de procesamiento complejo de eventos, *machine learning*, así como de Streaming SQL basado en Apache Calcite [74]. A diferencia de Spark donde se utiliza *micro-batch* para soportar flujo de datos, Flink realiza el procesamiento evento a evento permitiendo mayor flexibilidad en la programación de aplicaciones en tiempo real.

- **Apache Storm:** es un sistema de computación distribuida para aplicaciones en tiempo real, basando su funcionamiento en el modelo DAG. Los diagramas acíclicos o llamados topologías están compuestos por los nodos llamados *spouts* y *bolts*. Los nodos *spouts* son los generadores de los flujos de datos a procesar. Cada flujo de datos está compuesto por tuplas. Los nodos *bolts* son los que procesaran los diferentes flujos de datos generados, y a su vez podría generar como salida un flujo de datos para que sea procesado por otro nodo *bolt*. Además, el clúster de Storm se divide en dos tipos de nodos, un no-

do *master* y nodos *worker*. El nodo *master* implementa un *daemon* llamado Nimbus, encargado de gestionar las diferentes topologías a través del clúster Storm y gestionar los nodos *worker*. El nodo *worker* se implementa un *daemon* supervisor que comunica con Nimbus el estado de un *worker* [75].

Existen otras plataformas para el procesamiento de Big Data propuestas por vendedores de software como por ejemplo SAP, IBM, entre otros. Estas plataformas son propietarias y se integran con la suite de soluciones de cada vendedor. Por ejemplo, SAP Hana para el procesamiento de flujo de datos que se integra con la suite de productos software de SAP. De la misma forma, IBM presenta la plataforma de procesamiento en tiempo real IBM InfoSpere Streams capaz de procesar largos flujos de datos de manera eficiente y escalable [76].

2.3.6 Análisis del Big Data

El análisis del Big Data es el proceso con el cual se extrae la mayor cantidad de valor, información o conocimiento de los grandes volúmenes de datos para lograr conclusiones acertadas y mejorar la toma de decisiones. El proceso de análisis de datos está compuesto por una extracción, limpieza, transformación, modelado y visualización de los datos [77]. Dependiendo del objetivo y profundidad de nuestro análisis de datos, se puede clasificar los tipos de análisis en cuatro niveles: análisis descriptivo, diagnóstico, predictivo y prescriptivo.

Análisis descriptivo

El análisis descriptivo es la forma más básica de análisis de datos, pero con un gran potencial al brindar una amplia vista preliminar de la información que contienen los datos. Este tipo de análisis permite obtener conclusiones sobre el pasado y presente. La obtención de información es realizada a través de un análisis estadístico de los datos utilizando por ejemplo estadísticos de tendencia como mediana, media, máximo, mínimo, desviación estándar, varianza, moda, frecuencia, entre otros [56]. El resultado de estos estadísticos permite responder a preguntas del tipo qué, cuál, cómo, cuándo pasó, y así tener una visión del estado actual [77]. Actualmente, la mayor parte de plataformas de procesamiento de Big Data disponen de librerías y métodos que permiten realizar un análisis descriptivo de los datos [78].

Análisis de diagnóstico

El análisis de diagnóstico es un poco más complejo que el análisis descriptivo y permite responder a preguntas del tipo por qué y dónde. Al igual que el análisis descriptivo, el análisis diagnóstico utiliza la estadística para responder a las cuestiones planteadas. Técnicas estadísticas donde se evalúa la distribución de probabilidad, correlación de Pearson, test de hipótesis, entre otros, son principalmente utilizadas, al igual que las técnicas de *machine learning* como las regresiones lineales, clasificación, y agrupamiento. Algunas librerías de las plataformas permiten realizar este tipo de análisis, pero algunas son muy limitadas. Otras librerías como Scikit-learn y el software estadístico R Studio implementan este tipo de algoritmos y pueden ser utilizadas mediante la extracción de un muestreo aleatorio del gran conjunto de datos para realizar este tipo de análisis [78].

Análisis predictivo

El análisis predictivo permite predecir tendencias en base a probabilidades usando técnicas estadísticas avanzadas y de minería de datos para extraer patrones. Las técnicas de regresión y clasificación son las más utilizadas, pero recientemente las técnicas de *machine learning* y *deep learning* están tomando mayor importancia por el bajo porcentaje de error [77]. Este tipo de análisis permite responde a preguntas sobre el futuro de algunos eventos [77]. Si bien algunas librerías de las plataformas de procesamiento de Big Data soporta las técnicas de *machine learning*, estas aún son limitadas y dificultan implementar modelos de *deep learning*. Por tal motivo, el uso de *frameworks* especializados para implementar *machine learning* y *deep learning* es cada día más utilizado para realizar un análisis predictivo [56].

Análisis prescriptivo

El análisis prescriptivo es el más complejo de todos, puesto que analiza todo el sistema a profundidad para identificar el comportamiento y optimizar su funcionamiento. Este tipo de análisis está enfocado a sugerir varias acciones para una toma de decisiones [77]. El análisis prescriptivo se vale de un análisis predictivo realizado previamente para realizar las recomendaciones de mejora. Así como el análisis predictivo, este tipo de análisis utiliza técnicas de IA para cumplir con su objetivo [56].

Actualmente, las técnicas de minería de datos y de *machine learning* son utilizadas para extraer valor de los datos. Las técnicas de minería de datos, las cuales tiene origen estadístico, son las que mayormente se utilizan en los sistemas de análisis tradicionales [79]. De acuerdo a esto, los métodos cono-

cidos son: la clasificación, regresión, correlación, factor, test A/B, y análisis estadístico [5]. Por otro lado, el Big Data ha potencializado las técnicas de IA debido a que muchos de estas técnicas requieren grandes volúmenes de datos para un óptimo rendimiento. De esta manera, técnicas de *machine learning* y sobre todo *deep learning* han ganado un rol protagónico para el desarrollo de análisis de datos en los últimos años. En los sistemas de análisis actuales son empleados mayormente las técnicas de IA para realizar un análisis predictivo y prescriptivo de los datos [80]. Por ello, en la siguiente sección se analizan las técnicas de *machine learning* y *deep learning* mayormente aplicadas en IoT al igual que los *frameworks* y librerías disponibles.

2.3.7 Big Data y su relación con la inteligencia artificial

La idea de que una máquina pueda pensar y tomar decisiones como un ser humano ha intrigado a varios científicos desde hace algunas décadas. Tal es así que, las técnicas de *machine learning* llevan ya muchos años siendo estudiadas para alcanzar dicho objetivo. Sin embargo, las técnicas de IA han ganado mayor relevancia en los últimos años por ser útiles para extraer información de los datos y sobre todo del Big Data. Las técnicas de *machine learning* son utilizadas para interpretar los patrones, reproducirlos y generar una predicción basados en el gran volumen de datos. Principalmente, las técnicas *deep learning* se han visto beneficiadas con el Big Data porque ha incentivado el desarrollo de innovadoras aplicaciones que han resuelto problemas de difícil resolución con las técnicas de *machine learning*.

Machine learning

El *machine learning* es un conjunto de algoritmos que tienen la capacidad de aprender de manera automática patrones para generar predicciones. Los algoritmos utilizan conjuntos de datos, llamados datos de entrenamiento, cuya composición puede variar dependiendo de la técnica de aprendizaje utilizada. Las técnicas de aprendizaje se dividen en supervisadas y no supervisadas. De esta manera, el conjunto de datos de entrenamiento utilizado por las técnicas supervisadas contendrán los datos de entrada con sus respectivos resultados (etiquetas) que se desean obtener. Mientras que las técnicas no supervisadas utilizarán en su conjunto de datos de entrenamiento únicamente contendrán los datos de entrada. Los algoritmos también se pueden clasificar dependiendo del objetivo en: algoritmos de regresión, clasificación, agrupación y reducción de dimensiones, Figura 2.11 [80] [81].

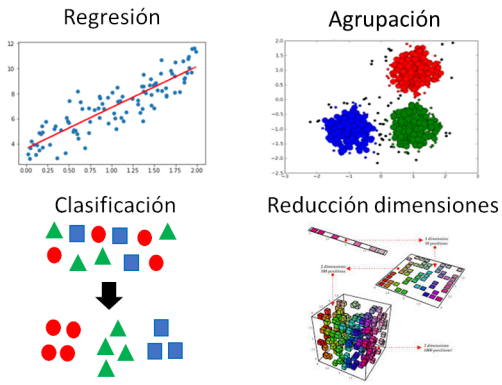


Figura 2.11: Algoritmos de aprendizaje automático

- **Algoritmos de Regresión:** tienen como objetivo predecir un valor determinado conociendo el valor de entrada. Los valores de predicción puede ser un conjunto continuo de valores. Estos utilizan la técnica supervisada para su aprendizaje. Dentro de este grupo se destacan los algoritmos de regresión lineal, máquinas de vectores de soporte de regresión (SVM, por sus siglas en inglés, *Support Vector Machine*) y árbol de regresión.

- **Algoritmos de Clasificación:** tienen como objetivo clasificar los datos de entrada en clases. Las clases son parte de un conjunto discreto de valores. Estos utilizan la técnica supervisada para su aprendizaje. Dentro de este grupo destacan los algoritmos de K vecinos más cercanos (K-NN, por sus siglas en inglés, *K-Nearest Neighbours*), SVM, árbol de decisión y bosque aleatorio.

- **Algoritmos de agrupación:** tiene como objetivo agrupar los datos dependiendo de sus características. Estos utilizan la técnica no supervisada para su aprendizaje. El algoritmo de mayor utilización en IoT es K-means.

- **Algoritmos de reducción de dimensiones:** son algoritmos que buscan reducir la cantidad de variables presentes en el conjunto de datos. El algoritmo de análisis de componentes principales (PCA, por sus siglas en inglés, *Principal Component Analysis*) es el más representativo de todos por ser un algoritmo intuitivo y que facilita su aplicación para reducir la complejidad de IoT.

Los algoritmos indicados en cada clase son los más utilizados en IoT, pero existen más algoritmos que son aplicados en la extracción de valor en otras áreas. En la Tabla 2.2 se enlista los algoritmos de *machine learning* más utilizados en análisis de datos para IoT [80].

Tabla 2.2: Algoritmos de *machine learning* empleados en IoT

Algoritmo de *machine learning*	**Técnica de aprendizaje**
K-NN	Clasificación
Naive Bayes	Clasificación
SVM	Clasificación
Regresión Lineal	Regresión
Regresión SVM	Regresión
Arboles de Decisión	Regresión/Clasificación
Bosque Aleatorio	Regresión/Clasificación
K-medias	Agrupación
PCA	Reducción de Dimensiones
DBSCAN	Agrupación

Density-based Spatial Clustering of Applications with Noise (DBSCAN).

Adicional a estos algoritmos, las redes neuronales artificiales (ANN, por sus siglas en inglés, *Artificial Neural Network*) han sido utilizadas en la extracción de información y detección de patrones en entornos IoT [80]. Las ANNs aparecieron para solventar las limitaciones en la elección de las variables más relevantes para la elaboración de los modelos de regresión y clasificación. Las ANNs tratan de imitar el funcionamiento de las neuronas cerebrales. Cada unidad neurona, también llamado perceptron, recibe una o más variables de entrada (x_i) con su respectivo peso (w_i) y luego de realizar operaciones internas que le ayuden a retener información generarán una salida (y), como se muestra en la Figura 2.12.

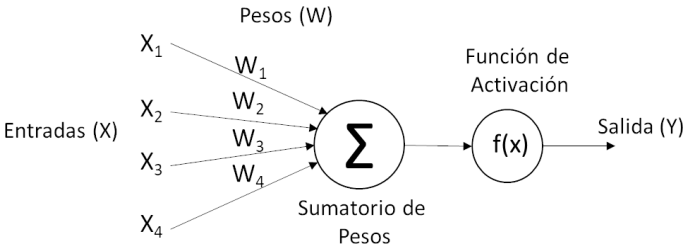


Figura 2.12: Estructura de un Perceptrón

Las neuronas se interconectan entre sí formando una capa de redes neuronales o capa de múltiples perceptrones (MLP, por sus siglas en inglés, *Multilayer Perceptron*). Las ANNs tienen una capa de entrada, una capa escondida y una capa de salida, como se muestra en la Figura 2.13. La capa de entrada es un conjunto de neuronas que reciben las variables del conjunto de datos. La capa de salida es un conjunto de neuronas que muestran los resultados obtenidos, ya sean estos un valor finito (clase) o un valor infinito (regresión). Las capas escondidas están compuesta por una o más capas que son las encargadas de extraer las características de los datos de entrada y detectar los patrones a través de las operaciones que realiza cada perceptron. Las capas escondidas pueden ser tan largas como sean necesarias, esto se conoce como profundidad de la red. Las variaciones en profundidad y diseño de las redes neuronales han permitido desarrollar varios algoritmos que se agrupan dentro del área del *deep learning* [82].

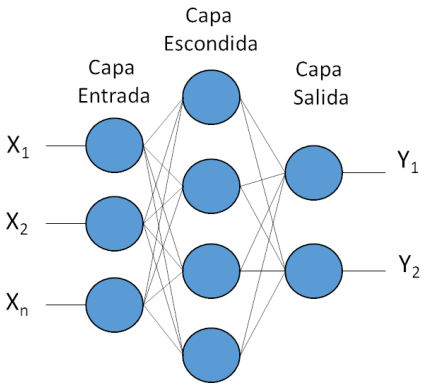


Figura 2.13: Ejemplo de una estructura de una red ANN

Deep learning

El *deep learning* es una rama de *machine learning* que emplea una arquitectura de red neuronal compleja para detectar los patrones de los datos y generar predicciones. Actualmente, existe una amplia variedad de arquitecturas de *deep learning* para extraer información de los datos. Particularmente, las arquitecturas de redes neuronales convolucionales (CNN, por sus siglas en inglés, *Convolutional Neural Network*) y las redes neuronales recurrentes (RNN, por sus siglas en inglés, *Recurrent Neural Network*) han recibido especial atención en los últimos años debido a su amplia aplicación en el campo de la visión artificial y el reconocimiento de voz [82]. Si bien existen otros algoritmos de *deep learning*, los algoritmos CNN y RNN son los mayormente utilizados para detección de patrones y predicción de

estados en el ámbito de IoT [83]. A continuación se presentan una breve introducción a los algoritmos CNN y RNN.

- **CNN:** se caracterizan por realizar una operación convolucional durante el proceso de extracción de información. Este tipo de redes son capaces de procesar datos de una dimensión (secuencia de datos) o de dos dimensiones (imágenes) que siguen una topología cuadriculada o matricial [82]. Las capas de las redes convolucionales pueden realizar tres fases, la convolución, la activación con funciones lineales y la reducción usando una función de agrupación. El proceso de convolución utilizan filtros, también llamados *kernel*, como función convolucional para recorrer la cuadricula de entrada y generar como resultado una cuadricula de menor tamaño que el anterior, llamado mapa de características. Este proceso permite la extracción de características particulares de los datos de entrada; así por ejemplo, la extracción de los bordes de una imagen. El mapa de características pasa por una función de activación no lineal que puede ser la función *sigmoide*, *Rectified Linear Unit* (ReLU), o la función *tanh*. Finalmente, el proceso de reducción disminuye la cantidad de parámetros mediante el uso de estadísticos como el promedio o el valor máximo. El resultado de las capas convolucionales son considerados como datos de entrada para una MLP que es la encargada de predecir los valores o de la clasificación en clase de los datos de entrada. En la Figura 2.14 se puede observar la arquitectura típica de este tipo de redes [82].

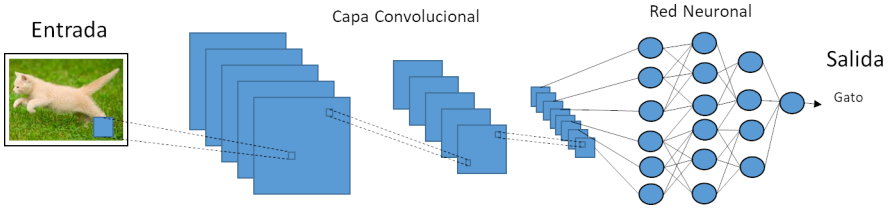


Figura 2.14: Arquitectura típica de las redes neuronales convolucionales

- **RNN:** tienen la habilidad de almacenar en memoria estados anteriores debido a una realimentación de capas previas o de la misma capa [82]. Por ello, estas redes son capaces de resolver problemas del tipo series de tiempo o con datos que siguen una secuencia de tiempo. Las RNN son adecuadas para enfrentar los problemas de IoT, debido a que los datos de IoT son producidos siguiendo una secuencia temporal [83]. Las RNN varían dependiendo del tipo de retroalimentación y de memoria utilizada para retener información. Las más conocidas son las redes con memoria a corto y largo plazo (LSTM, por sus siglas en

inglés, *Long Short-Term Memory*) y las redes de unidad cerrada recurrente (GRU, por sus siglas en inglés, *Gated Recurrent Unit*) [82]. En la Figura 2.15 se puede observar la retroalimentación en cada marca de tiempo realizada en una red RNN.

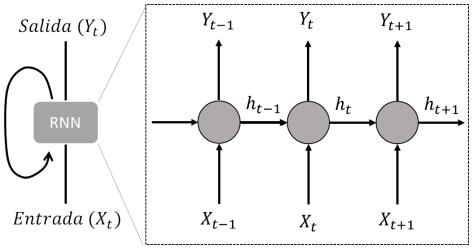


Figura 2.15: Arquitectura de una RNN

Plataformas para machine learning y deep learning

Actualmente existen varios *framework*s y herramientas que permiten implementar modelos de *machine learning* y *deep learning* de manera rápida e intuitiva. Los *framework*s más conocidos para *machine learning* buscan la integración con ambientes de Big Data como Apache Hadoop y Spark para tomar ventaja de la computación distribuida. A continuación se detallan los *framework*s y librerías de mayor utilización en el desarrollo de modelos de predicción basados en los algoritmos de *machine learning* y *deep learning*.

- **Caffe:** es un *framework* de *deep learning* con licencia *Berkeley Software Distribution* (BSD) desarrollado por la Universidad de Berkeley, y permite el despliegue de redes neuronales convolucionales. Caffe está basado en C++ y permite la integración con Python y Matlab. Además, permite el cambio rápido entre el uso de *Central Processing Unit* (CPU) y *Graphics Processing Unit* (GPU) [84]. La conexión con Apache Spark es factible mediante el uso de CaffeOnSpark para el uso en clúster [85].
- **TensorFlow:** es un *software open source* desarrollado por Google para el desarrollo de aplicaciones en *machine learning* y *deep learning*. Presenta una arquitectura flexible para el manejo de CPU o GPUs. Permite una integración con Java, Python y C++. TensorFlow introduce el termino tensores el mismo que hace referencia a la matriz de datos que ingresa a los nodos. En los nodos se programan las operaciones matemáticas que se diseñan en las redes neuronales [86]. Además, TensorFlow tiene una conexión con Apache Spark mediante TensorFlowOnSpark [87].

- **BigML:** es una plataforma de generación de modelos para *machine learning*. BigML utiliza APIs *Representational State Transfer* (REST) para la comunicación con su plataforma web. No es una plataforma *open source* y la misma puede ser implementada de forma privada. En resumen, BigML provee el servicio de *machine learning* vía web [88].
- **Keras:** es un *framework* de alto nivel que permite generar prototipos y entrenar modelos de *deep learning* de manera muy rápida e intuitiva a través de sus APIs. Además, este *framework* es capaz de trabajar conjuntamente con TensorFlow como *backend* para aumentar su flexibilidad. Keras permite una integración con el lenguaje de programación Python [89].
- **Mahout:** es una librería perteneciente al ambiente Hadoop y licenciada bajo Apache. Mahout contiene algoritmos de *machine learning* utilizando *MapReduce*. Mahout es utilizado en un procesamiento *batch* e implementa 4 funcionalidades: filtrado, categorización, clasificación y búsqueda de patrones [90].
- **Weka:** es un software *open source* que implementa algoritmos de *machine learning* y minería de datos. Weka se encuentra implementado con con el lenguaje de programación Java [91].
- **Scikit-Learn:** es una librería que agrupa herramientas simples para el desarrollo de modelos de predicción. Cuenta con una amplia gama de algoritmos de clasificación, regresión, agregación, dimensión de la reducción y de algoritmos que permiten realizar un pre-procesamiento de los datos. La librería se encuentra implementada en Python [92].

2.3.8 Arquitecturas de Big Data

Las arquitecturas para el análisis de Big Data propuestas estructuran una metodología que agrupa las nuevas tecnologías, plataformas, modelos, herramientas, librerías y lenguajes de programación para obtener los beneficios esperados del Big Data. Varias arquitecturas de Big Data han sido propuestas para resolver problemas con el manejo del volumen de datos por entidades y organismos de estandarización, empresas dedicadas a las redes sociales, vendedores de software en la industria, y por la academia. A continuación se presenta algunas de las arquitecturas y su ámbito de aplicación.

Arquitectura de referencia NIST Big Data

La arquitectura de referencia NIST Big Data (NBDRA, por sus siglas en inglés, *NIST Big Data Reference Architecture*) tiene el objetivo de enmarcar las metodologías, técnicas, y modelos que envuelven el Big Data para que exista un entendimiento común con el fin de definir arquitecturas interoperables. Esta arquitectura fue propuesta por el grupo de trabajo *NIST Big Data Working Group* (NBD-PWG). La arquitectura puede ser aplicada en cualquier ámbito de aplicación debido a que es una arquitectura genérica. La arquitecturas es capaz de adaptarse a los requerimiento particulares de cada ámbito de aplicación [93].

La arquitectura NBDRA consiste de un modelo conceptual y dos vistas arquitecturales desarrolladas completamente independientes del vendedor y tecnología de infraestructura [93]. La Figura 2.16 muestra el modelo conceptual de la arquitectura. En ella se observa cinco componentes conectados por interfaces interoperables. Estos componentes son el proveedor de datos, el consumidor de datos, el proveedor de aplicaciones Big Data, el proveedor de *framework* de Big Data y el orquestador del sistema.

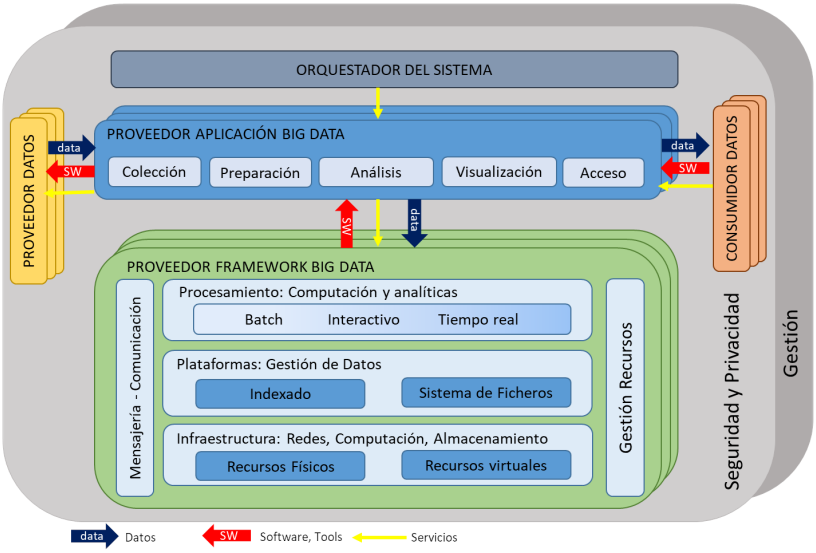


Figura 2.16: Arquitectura de referencia NIST Big Data

El orquestador del sistema está encargado de gestionar y orquestar las operaciones del sistema de Big Data. El proveedor de datos provee nuevos datos al sistema de Big Data a través de APIs basadas en la comunicación publicación/subscripción. Además, el proveedor de datos se encarga de describir

las fuentes de datos (formatos, modelo de datos, políticas de acceso, entre otros) a través de metadatos. El proveedor de la aplicación de Big Data ejecuta las operaciones a lo largo del ciclo de vida del Big Data y dependiendo de las necesidades del orquestador del sistema. El proveedor de aplicaciones sigue las actividades de colección, preparación, análisis, visualización y acceso propuesto en el modelo BDLM de Demchenko et al. [52]. El proveedor de *framework* de Big Data organiza jerárquicamente las instancias de los componentes tecnológicos relacionados con Big Data. El proveedor de *framework* consiste de *frameworks* de infraestructura (recursos de red, computacionales y de almacenamiento), plataformas de gestión de datos (sistema de ficheros, indexados y en memoria) y *frameworks* de procesamiento (procesamiento, *batch*, interactivo, y en tiempo real). El consumidor de datos es el usuario final u otro sistema. Las actividades principales de los consumidores de datos son las de buscar y visualizar resultados, descargar reportes y analizar localmente, o usar los resultados de la aplicación de Big Data para sus propios procesos.

Arquitectura de referencia de Big Data Value (BDV)

La arquitectura de referencia *Big Data Value* (BDV) fue desarrollada por la *Big Data Value Association* (BDVA), la cual está impulsada por la industria europea. La arquitectura tiene el objetivo de servir como marco de referencia para integrar e implementar adecuadamente los habilitadores tecnológicos de Big Data. El modelo de arquitectura está estructurada en niveles horizontales y verticales [94]. La Figura 2.17 muestra el modelo arquitectural de BDV.

Figura 2.17: Modelo Arquitectural BDVA

Los niveles horizontales concentran las funcionalidades específicas para el análisis de datos. Las funcionalidades horizontales son la de gestión de datos, protección de datos, arquitecturas de procesamiento de datos, análisis de datos, y visualización de datos. Además, se incluyen como intereses horizontales a las funcionalidades provistas por el *cloud computing* y por la computación de alto rendimiento (HPC, por sus siglas en inglés, *High Performance Computing*) para el manejo efectivo del Big Data, y a las funcionalidades provistas por *fog computing* y el *edge analytics* para el soporte de los requerimientos de IoT. La arquitectura sugiere el uso de la arquitectura de referencia de IoT propuesta por *Alliance for Internet of Things Innovation* (AIOTI). Esta arquitectura de IoT fue analizada en la subsección 2.2.1

Los niveles verticales concentran las funcionalidades que afectan a todos los niveles horizontales. Las funcionalidades verticales son las de tipos de Big Data y semántica, estándares, comunicación y conectividad, cyberseguridad, mercados de datos, y el desarrollo y operaciones.

Arquitectura lambda

La arquitectura Lambda fue diseñada por Nathan Marz para Twitter. La arquitectura está compuesta por la capa de procesamiento *batch*, la capa de procesamiento en tiempo real y la capa de servicio. La capa de procesamiento *batch* almacena los datos en un repositorio y los procesa utilizando procesamiento *batch*. Los resultados del procesamiento *batch* serán almacenados en forma de vistas de datos en la capa de servicio para su eventual distribución a los usuarios finales. Mientras tanto, la capa de procesamiento en tiempo real compensa la latencia que provoca el procesamiento *batch*, generando de esta manera un procesamiento en períodos cortos para de manera incremental actualizar las vistas generadas por el procesamiento *batch*. En otras palabras, la capa de procesamiento en tiempo real procesa los flujos de datos que llegaron recientemente al sistema [75]. La Figura 2.18 muestra el funcionamiento de la arquitectura Lambda. Si bien el ámbito de aplicación está enmarcado en las redes sociales, esta arquitectura ha servido de base para el diseño de arquitecturas de análisis de Big Data en IoT [57] [8].

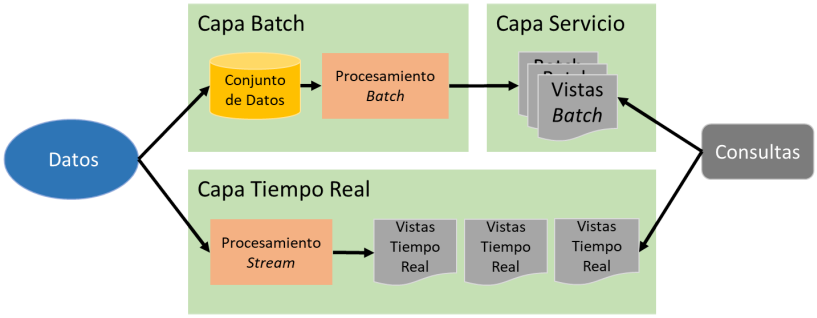


Figura 2.18: Arquitectura Lambda

Arquitectura kappa

La arquitectura de procesamiento Kappa, propuesta por Jay Kreps para LinkedIn, es más simple en funcionamiento que Lambda. Kappa no cuenta con una capa de procesamiento *batch*, considerando únicamente una capa de procesamiento en tiempo real y una capa de servicio. Los datos son almacenados temporalmente en un bróker de comunicaciones del tipo publicación/subscripción para luego ser inyectados en una plataforma de procesamiento en tiempo real. Los datos procesados son almacenados en forma de tablas en una base de datos en la capa de servicio para brindar información a los usuarios. Los datos son reprocesados periódicamente para generar otra tabla de resultados que será almacenado en la capa de servicio. Este proceso se repite la cantidad de veces necesarias. El usuario realizará la consulta a la base datos de la capa servicio y esta podrá agrupar varias tablas de resultados. La principal desventaja es el crecimiento de las tablas en las base datos debido a las tareas de reprocesamiento [95] [96]. La Figura 2.19 muestra el funcionamiento de la arquitectura Kappa.

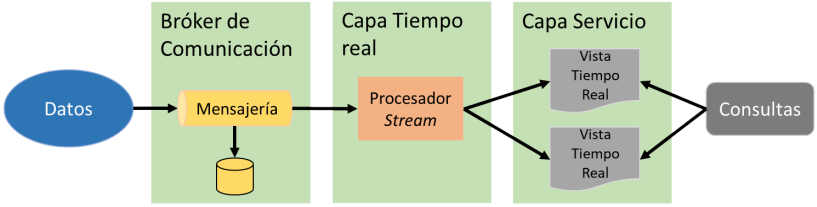


Figura 2.19: Arquitectura Kappa

Arquitecturas referencia de vendedores de software

Varios vendedores de *software* han presentado sus arquitecturas de Big Data para que sirvan como referencia en la implementación de soluciones de Big Data en la industria. A continuación se describe algunas de las arquitecturas de referencia de Big Data de vendedores de software.

- **Arquitectura de referencia de análisis de Big Data de Oracle** Esta arquitectura tiene el objetivo de extender las capacidades de los sistemas de información actuales para brindar un soporte a la gestión de Big Data. Además, la arquitectura ofrece una solución integral con sus productos de base de datos NoSQL y la incorporación de tecnologías de Big Data como Apache Hadoop. El modelo arquitectural de alto nivel está compuesta por seis niveles: nivel de infraestructura, de información, de servicios, de procesos, de interacción, y de multicanal de entrega [97] [98], Figura 2.20.

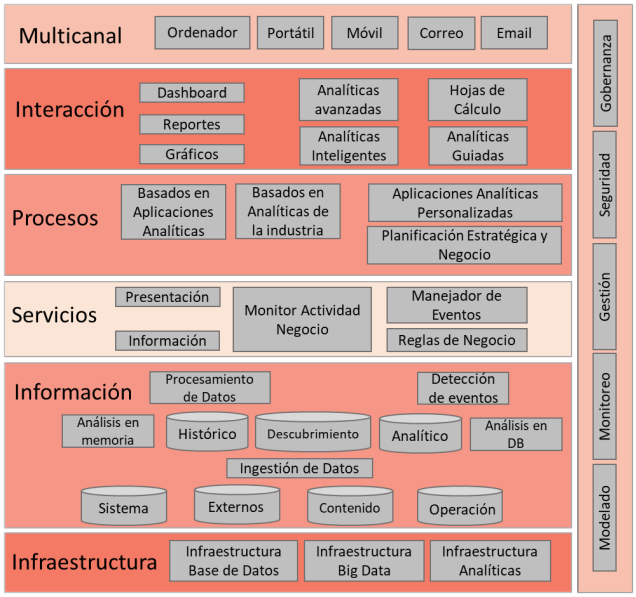


Figura 2.20: Vista lógica arquitectura de referencia Oracle

El nivel de infraestructura incluye *hardware* y plataformas de almacenamiento, procesamiento y análisis de Big Data. El nivel de información incluye las funcionalidades para la ingestión de los datos, procesamiento de datos tanto en tiempo real como *batch*, y almacenes de datos para propósitos específicos. El nivel de servicios incluye

funcionalidad para proveer servicios de presentación de información, gestión de eventos, gestión de reglas de negocio, y monitorización de las actividades de negocio. El nivel de procesos incluye funcionalidades para realizar tareas de procesamiento de alto nivel orientadas a la extracción de valor del Big Data. El nivel de interacción incluye las funcionalidades para soporta la interacción con los usuarios finales. El nivel multicanal engloba los canales por los que se distribuye la información a los usuarios finales tales como teléfonos móviles, correo electrónico, ordenadores, etc. Además, la arquitectura incluye un nivel transversal con funcionalidades para la gestión, seguridad, gobernanza, monitorización y modelado.

- **Arquitectura de Big Data de SAP** La arquitectura está orientada a aplicaciones de negocio. Al igual que la arquitectura anterior, SAP busca incorporar la plataforma Hadoop con su plataforma de procesamiento en tiempo real SAP Hana. La arquitectura está compuesta por un nivel de ingestión y aprovisionamiento, nivel de plataformas de Big Data, nivel de SAP Hana, y nivel de aplicaciones, Figura 2.21.

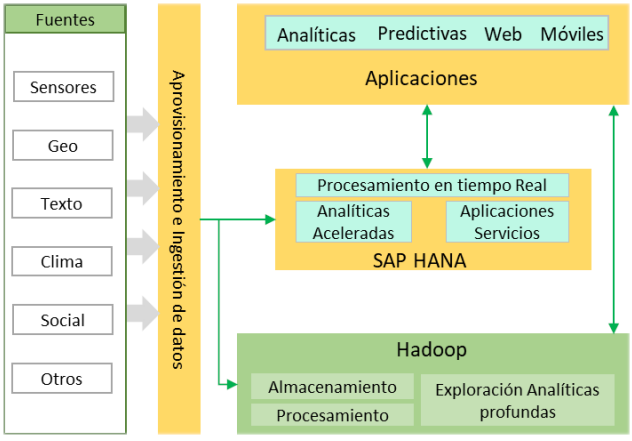


Figura 2.21: Arquitectura de referencia SAP y HortonWorks

El nivel de ingestión carga los datos provenientes de múltiples fuentes a las plataformas de Big Data y la plataforma SAP Hana. El nivel de plataformas de Big Data utiliza Apache Hadoop para almacenar, procesar y realizar un análisis exploratorio de los datos. Por otro lado, el nivel de plataforma SAP Hana se encarga del procesamiento en tiempo real y la exploración interactiva de los datos. Finalmente, el nivel de aplicaciones se encarga de presentar los resultados de las analíticas a los usuarios finales a través de aplicaciones web y móviles.

Otros vendedores de software como IBM [99], Pivotal [97], entre otros, también han presentado una arquitectura de referencia similar a las presentadas por Oracle y SAP. En estas arquitecturas predomina el uso de Hadoop integrado con las plataformas de cada vendedor.

Arquitecturas de Big Data para IoT

El uso de las metodologías, técnicas y plataformas de Big Data para solucionar el problema de la generación de gran volumen de datos en IoT ha sido de interés en la academia en los últimos años. A continuación se describen dos arquitecturas de Big Data propuestas para entornos IoT.

- **Arquitectura para Big Data IoT GSMA**

  La arquitectura de Big Data para IoT propuesta la *Global System for Mobile Communication Association* (GSMA) esta destinada a la participación de los operadores móviles en el despliegue de ecosistemas de IoT y Big Data. La Figura 2.22 muestra un diagrama funcional de la arquitectural. La arquitectura esta compuesta 8 unidades funcionales: nivel de datos de contexto, nivel de servicios IoT, mediador de protocolos y datos, bróker de datos y control, gestor de acceso de APIs a pares, almacenamiento de Big Data IoT, procesamiento de Big Data IoT y el gestor de acceso a APIs de desarrolladores.

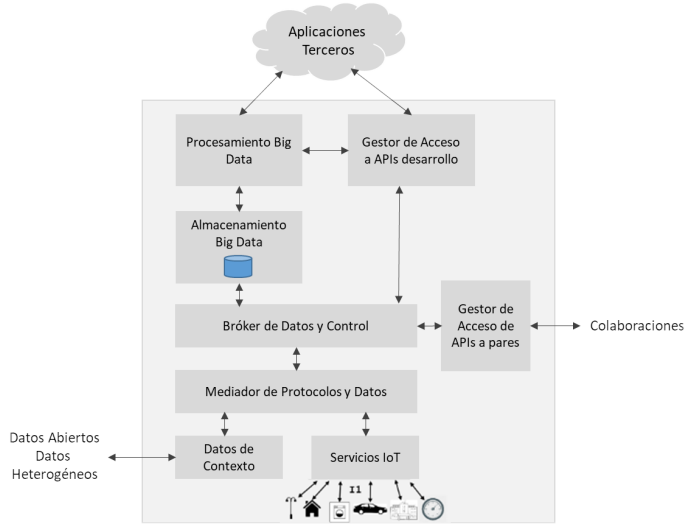


Figura 2.22: Arquitectura general para IoT y Big Data

El nivel de datos de contexto se encarga de integrar otras fuentes de datos no IoT para enriquecer los datos de IoT con contexto. El nivel de servicios IoT se encarga de gestionar las interacciones especificas de los dispositivos IoT a través de una comunicación bidireccional. El mediador de protocolos y datos agrupa las funciones para extraer los datos de los niveles de datos de contexto y los niveles de servicios IoT para cargarlos en bróker de datos y control. El bróker de datos y control tiene la responsabilidad de habilitar el acceso aplicaciones de terceros, así como de gestionar los comandos para los dispositivos IoT enviados desde la aplicación. El gestor de acceso de API a pares controla la comunicación entre bróker de datos y control de otro entorno de IoT para recibir y publicar datos que brinden contexto. El gestor de acceso a APIs de desarrolladores controla el acceso a los datos de contexto de las entidades IoT y controla los servicios para aplicaciones de terceros. El nivel de almacenamiento del Big Data generado por IoT facilita las funciones para almacenar los datos masivos de IoT que serán utilizados para extraer información a través de una base de datos del tipo NoSQL. El nivel de procesamiento de Big Data realiza un tratamiento a los datos almacenados y analiza los datos a través de plataformas de procesamiento como Apache Spark, Elastic Search, Apache Mahout, TensorFlow.

- **Arquitectura BASIS**

  La arquitectura BASIS propuesta por Costa C. y Santos M. [64] está orientada a solucionar los problemas del manejo del Big Data generado en ciudades inteligentes. La descripción de la arquitectura fue realiza desde los puntos de vista conceptual, tecnológico y de infraestructura. Cada vista arquitectural provee información detallada sobre su funcionamiento y los roles involucrados en cada uno de los procesos que intervienen en la gestión del Big Data generado en entornos de ciudades inteligentes.

  La vista conceptual encapsula los componentes que representan un importante papel en la extracción y análisis de Big Data. Los componentes de la vista conceptual son las fuentes de Big Data (principalmente aplicaciones IoT relacionados con la ciudad inteligente), la extracción del Big Data, el almacenamiento del Big Data, el análisis de Big Data y los servicios y aplicaciones.

  La vista tecnológica detalla el uso de las tecnologías principales para la generación de una instancia de la arquitectura BASIS. En esta vista se analizan el uso de plataformas para el almacenamiento como HDFS, base de datos NoSQL como Cassandra, MongoDB y HBase. Además, se analizan las plataformas de procesamiento como Spark, y de análisis como Weka.

La vista de infraestructura describe los niveles físicos de la solución. Esta vista detalla la distribución de las plataformas de Big Data en los servidores que conforman el centro de datos de la ciudad inteligente. Además, la vista proporciona información sobre el soporte de alta disponibilidad y escalabilidad.

La arquitectura Basis no es la única arquitectura propuesta por la academia, pero si es la que más componentes detalla. Así por ejemplo, las arquitecturas propuestas por Rathore M. et al. [100], Ta-Shma P. et al. [101], Sun Y. et al [102], Cheng B. et al. [63] fueron propuestas para solucionar los problemas de ciudades inteligentes; pero a diferencia de la arquitectura BASIS no se presentan en detalle. Por otro lado, otras arquitecturas de interés son las propuestas por Strohbach M. et al. [57] para ciudades inteligentes y Villari M. et al. [103] para edificios inteligentes. Estas arquitecturas adaptan la arquitectura Lambda a los requerimientos de IoT. Otras propuestas se enfocan en solucionar problemas puntuales e individualmente como el almacenamiento, procesamiento, o el análisis de datos y no presentan una arquitectura donde se gestione el ciclo de vida completo del Big Data [8].

2.4 Tecnologías habilitadoras de la gestión del Big Data de IoT

Cloud computing puede considerarse como la tecnología donde confluyen tanto IoT como las metodologías y técnicas de Big Data, principalmente, porque presenta capacidades casi "ilimitadas" para soportar el almacenamiento, procesamiento y análisis del Big Data; así como, para solventar las limitaciones de los dispositivos IoT en el tratamiento de los datos generados. No obstante, *cloud computing* presenta limitaciones para la gestión del Big Data de IoT debido principalmente a las características particulares del Big Data generado por IoT. Los avances en las tecnologías de *fog computing* y *edge computing* buscan superar estas limitaciones del *cloud computing* para de manera conjunta soportar el Big Data generado por IoT. En las siguiente subsecciones se analiza estas tecnologías habilitadoras de la gestión del Big Data generado por IoT.

2.4.1 Cloud computing

Cloud computing o computación en la nube es una tecnología madura que ofrece mayores capacidades para una computación compleja y a gran escala a través de Internet [104]. *Cloud computing* provee un acceso fácil y seguro a recursos computacionales bajo demanda a personas y negocios. Además, esta tecnología provee ventajas técnicas como eficiencia energética, optimización de recursos, elasticidad, o aislamiento de ambientes de ejecución. Todos estos beneficios han sido posible debido a los pilares tecnológicos fundamentales en los que se basa *cloud computing*, la virtualización y la computación distribuida. Como resultado, *cloud computing* presenta características de flexibilidad, escalabilidad, alta disponibilidad y seguridad, convirtiéndolo de esta manera en la tecnología con más interés para manejar el Big Data generador por IoT [104].

Virtualización

La virtualización es una tecnología que permite crear recursos computacionales virtuales o lógicos con base en la abstracción de los recursos físicos [104]. Como resultado, el uso de los recursos computacionales es más eficiente y flexible. Las técnicas de virtualización de mayor utilización son las máquinas virtuales y los contenedores.

- **Máquina Virtual o *Virtual Machine*:** es un sistema operativo (invitado) que es capaz de ser instalado y ejecutado usando recursos computacionales virtuales (memoria, procesador, disco duro, etc.) proporcionados por el sistema operativo (*host*), Figura 2.23 [105]. La virtualización es a nivel de hardware gestionado por un hipervisor. El hipervisor es el encargado de abstraer los recursos físicos, crear un conjunto de recursos computacionales personalizables y presentar estos recursos para que las máquinas virtuales los utilicen. Como resultado, cada máquina virtual se encuentra aislada y es independiente de las demás, por lo que se obtiene cierto nivel de seguridad para instalar cualquier sistema operativo y ejecutar las aplicaciones necesarias para su funcionamiento sin afectar a las otras máquinas virtuales. Algunos ejemplos de hipervisores son Xen, VMware, y *Kernel-based Virtual Machine* (KVM) [106].
- **Contenedor:** es un sistema operativo ligero que es ejecutado usando los recursos computacionales físicos proporcionados por el sistema operativo (*host*), Figura 2.24 [105]. La virtualización es a nivel de sistema operativo gestionado por una capa de virtualización (*container engine*). La capa de virtualización divide los recursos del *host*

para proveer interfaces virtuales de red y espacios independientes para los procesos de los contenedores. De esta manera, cada contenedor es capaz de ejecutar aplicaciones aisladas entre sí. Algunos ejemplos de capa de virtualización son Docker, *Linux Container* (LXC), y OpenVZ [106].

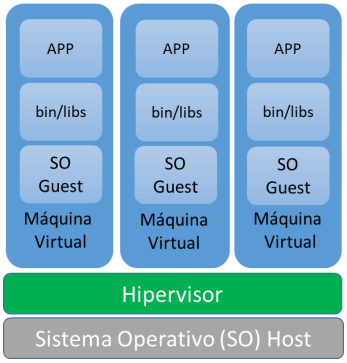


Figura 2.23: Virtualización de máquinas virtuales

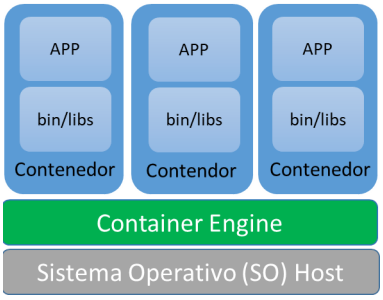


Figura 2.24: Virtualización en contenedores

Modelos de servicio cloud computing

Actualmente, *cloud computing* tiene tres modelos de servicio bajo de manda [104] como se puede ver en la Figura 2.25 y son detallados a continuación:

- *Infrastructure as a Service* (IaaS): provee recursos de infraestructura a través de máquinas virtuales. El usuario puede configurar sistema operativo, software y algunas configuraciones de red y seguridad.
- *Platform as a Service* (PaaS): provee recursos de la capa plataforma dando soporte de sistemas operativos y *software* para desarrollo de

frameworks. Las configuraciones permitidas para el usuario son más restrictivas y se limitan a las configuraciones y despliegue de las aplicaciones.

- *Software as a Service* (SaaS): provee aplicaciones a usuarios finales. El consumidor únicamente es capaz de configurar los parámetros que le permita la aplicación.

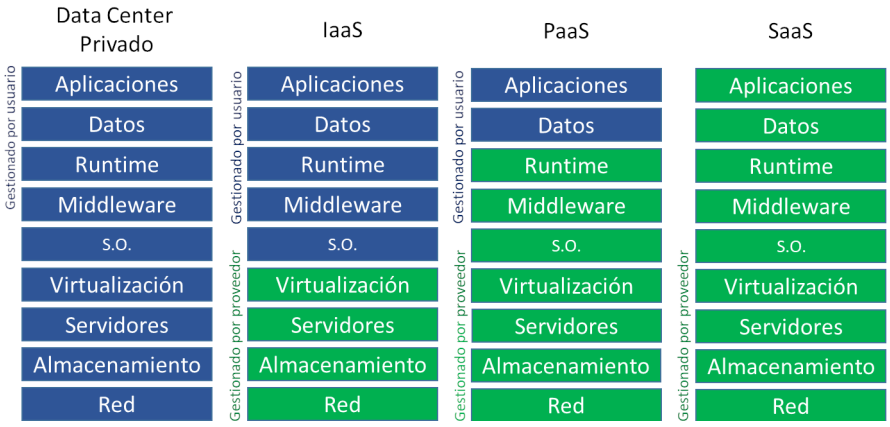


Figura 2.25: Modelos de servicio *cloud computing*

IoT ha encontrado en *cloud computing* una oportunidad para solventar las limitadas características de sus dispositivos para almacenar, procesar y analizar los datos [21]. La integración entre IoT y *cloud computing* era inminente por sus amplias ventajas. Esta integración ha provocado la generación de nuevos modelos de negocio como *Sensing as a Service* (SenaaS) [107].

El modelo de SenaaS propone que los datos recolectados por sensores sean almacenados, vendidos o negociados por el dueño de los datos. Por ejemplo, un sensor de temperatura en una casa puede recolectar datos y almacenar esta información en el *cloud*, y dicha información del sensor de temperatura es publicada para que un fabricante de equipos de termostatos pueda mejorar el rendimiento de sus productos. Además, esta integración dio lugar a la creación del concepto de *Cloud of Things* [108].

Cloud of Things

El concepto del *Cloud of Things* provee mecanismos para llevar los datos de los dispositivos IoT al *cloud*. Esta integración provee escalabilidad, confiabilidad, flexibilidad y seguridad para almacenar los datos de IoT de manera eficiente y económica. Varias propuestas tecnológicas de parte de empresas líderes en el ámbito de *cloud computing*, proyectos *open source* y proyectos de investigación promocionados por la Comisión Europa (programa Horizon 2020) fueron desarrollados con la idea de potenciar esta integración.

Las empresas tecnológicas han buscado explotar la infraestructura que tienen actualmente para soportar la inclusión de IoT en el *cloud*. Empresas como Amazon, Microsoft, Google e IBM promocionan el soporte de dispositivos inteligentes en su infraestructura *cloud computing* por medio de plataformas y diferentes servicios con el fin de satisfacer las necesidades de conectar el mundo físico con el mundo digital. Amazon permite la conexión de dispositivos IoT con su *cloud* usando la plataforma *Amazon Web Services* (AWS) IoT Core, el cual brinda una conexión segura de los dispositivos IoT, enrutamiento y procesamiento, control e interactividad con los dispositivos IoT. Los mensajes que recibe el AWS IoT Core pueden ser enrutados a otras plataformas del portafolio de productos de Amazon como AWS S3, Quicksight, Machine Learning, entre otros. Adicionalmente, el módulo AWS IoT device management es proporcionado para gestionar todos los dispositivos que se conectan al *cloud*, permitiendo una monitorización y configuración de políticas de los dispositivos [109].

Similarmente, Google propone su Google IoT Core para conectar dispositivos a su infraestructura *cloud computing* con funciones parecidas a la de su competidor, gestionando y enrutando mensajes al cloud. Google IoT Core implementa dos funciones básicas, la gestión de los dispositivos inteligentes con opciones de configuración y gestión segura, y convirtiendo los mensaje del protocolo MQTT a *Hypertext Transfer Protocol* (HTTP) y publicándolos en la plataforma de Google Cloud Pub/Sub. Los mensajes que llegan a Cloud Pub/Sub pueden ser consumidos por las distintas plataformas de Google, tales como Cloud BigTable, BigQuery, Cloud ML, Data Studio entre otros [110].

De la misma forma, Microsoft Azure IoT hub es una plataforma de servicio abierta y flexible que permite la conexión y gestión de dispositivos IoT de manera segura, y facilita la integración con otros servicios para procesar, almacenar y analizar los datos de IoT [111]. En resumen, las empresas tecnológicas han desarrollado plataformas para integrar los dispositivos IoT a su infraestructura *cloud computing* a través de APIs y explotar los datos recolectados usando sus amplios portafolios de servicios para Big Data.

Asimismo, una amplia variedad plataformas IoT del tipo *open source* han desarrollado iniciativas de integración con *cloud computing* con el fin de cubrir las limitaciones de IoT. Las más destacadas por su amplia usabilidad son FIWARE, OpenIoT, y Kaa. FIWARE es uno de los *open source* más utilizados en proyectos de ciudades inteligentes en Europa. FIWARE dispone de una amplia gama de GEs para almacenar, procesar y visualizar los datos en el *cloud*. El principal componente GE es Orion Context Broker el cual permite la conexión de sensores o dispositivos inteligentes haciendo uso de los agentes IoT. Orion Context Broker brinda flexibilidad para que se pueda conectar cualquier plataforma de Big Data externa o utilizar sus GEs, tales como WireCloud, Knowage, Kurento, Cosmo, entre otros, para el análisis de Big Data [24]. OpenIoT es un middleware que permite desplegar y proveer servicios en el *cloud* utilizando el concepto de SenaaS. A diferencia de FIWARE, OpenIoT provee mecanismos eficientes para usar y gestionar los sensores físicos y virtuales para ofrecer servicios bajo el modelo *pay-as-you-go*. Además, OpenIoT se enfoca en los aspectos de movilidad para gestionar eficientemente la energía de la recolección y transmisión de los datos en el *cloud* [112].

Similarmente, Kaa IoT platform es un *middleware* que permite construir soluciones IoT *end-to-end*, por medio de herramientas que facilitan el desarrollo de aplicaciones y servicios. Además, el *middleware* provee un entorno para gestionar los esquemas de los datos y permite la integración fácilmente vía API REST con plataformas de análisis de Big Data [113].

La comisión europea ha invertido en varios proyectos para el desarrollo de plataformas IoT que finalizan el 2020 [114]. El programa Horizon2020 busca desarrolla un ecosistema IoT integral, solucionando retos de IoT que retienen el crecimiento de las aplicaciones y servicios IoT. De esta manera, proyectos como symbIoTe, bIoTope, BIG IoT, AGILE, VINICITY, TagItSmart e Inter-IoT están siendo desarrollados para facilitar la interoperabilidad de plataformas IoT, la conexión con el *cloud computing* y la gestión de los dispositivos inteligentes. La heterogeneidad de dispositivos y plataformas IoT afecta el análisis de los datos de IoT porque cada plataforma presenta su propio modelo de datos y formatos para la compartición de archivos. Dicha heterogeneidad dificulta la integración y enriquecimiento de los datos de varias fuentes de IoT por lo que los datos de IoT no son explotados adecuadamente [54]. La interoperabilidad de IoT permite disponer de un solo formato y un solo modelo de datos sin descuidar la semántica de los mismos. Por ello, estos proyectos de integración, gestión y conexión con el *cloud computing* son importantes para el despliegue de servicios innovadores explotando el Big Data generado por los ecosistemas IoT.

Limitaciones de cloud computing

A pesar de las múltiples ventajas que tiene el manejo de los datos en la infraestructura *cloud computing*, esta presenta algunas limitaciones que deben ser afrontadas para el óptimo manejo del Big Data generado por IoT. A continuación describimos las limitaciones del *cloud*:

- **Latencia:** es la más crítica de las limitaciones que debe afrontar el análisis de los datos de Big Data. La latencia de extremo a extremo (tiempo transcurrido desde que el dispositivo IoT envía los datos al *cloud* hasta que recibe un comando para realizar alguna operación) es crítica en aplicaciones sensitivas a retardos, como por ejemplo en el monitorización remota de la salud de pacientes críticos [115]. A pesar de que el *cloud* puede brindar los recursos suficientes para un óptimo rendimiento en el procesamiento en tiempo real, la latencia depende de otros factores como los enlaces de comunicación y los dispositivos intermedios de red.
- **Ocupación de Ancho de Banda:** grandes volúmenes de datos que deben ser transmitidos al *cloud* puede causar una saturación a nivel de red. Los datos del tipo video y audio generados por dispositivos IoT durante la monitorización son principalmente los causantes de esta saturación [115]. Aunque existen mecanismos de compresión para reducir la cantidad de datos del tipo video y audio en IoT [116] [117], este aún se considera un problema.
- **Centralización Geográfica:** los recursos computacionales se encuentran localizados en una locación centralizada que puede ser distinta a la ubicación geográfica de los dispositivos IoT. Como resultado, los dispositivos IoT pueden estar ubicados geográficamente a grandes distancias en relación a la ubicación de los servidores que conforman el *cloud computing* lo cual puede incrementar la latencia y decrecer el QoS [118].
- **Bajo QoS**: debido a la distribución geográfica de los centros de datos que componen el *cloud computing* pueden provocar un bajo QoS y una baja percepción del servicio o calidad de experiencia (QoE, por sus siglas en inglés, *Quality of Experience*). Además, la comunicación entre los dispositivos IoT y el *cloud* depende de una red que por naturaleza no es confiable, el cual puede fallar causando una reducción en el QoS [118].

2.4.2 Fog computing y edge computing

Los conceptos de *fog computing* y *edge computing* emergen como propuestas tecnológicas eficientes para enfrentar a las limitaciones del *cloud computing* y manejar eficientemente el Big Data generado por IoT. La idea principal tras el funcionamiento de ambas es acercar las capacidades de almacenamiento y procesamiento a la fuente que genera los datos (dispositivos IoT, teléfonos móviles, cámaras, etc.). Aunque los dos paradigmas suelen ser vistos como análogos, estos presentan pequeñas características que los diferencian entre sí. A continuación, las características de cada uno de estos paradigmas son detalladas:

Edge computing

También es conocida como *Mobile Edge Computing* (MEC) y tiene su origen en las comunicaciones móviles [119]. MEC aparece bajo la necesidad de usuarios con dispositivos móviles inteligentes (smartphones) de mejorar su acceso a aplicaciones y servicios en la red móvil. En MEC, parte de los recursos computacionales del *cloud computing* son acercados al borde de la red móvil para cubrir estas necesidades a través del uso de servidores, *micro-datacenters* o *cloudlets* instalados en estaciones base [120]. Los *cloudlets* fueron una propuesta previa a MEC y están enmarcados en lo que se conoce como *Mobile Cloud Computing* [121]. Desde el punto de vista de IoT, *edge computing* se refiere a los dispositivos con capacidades de interconexión de red y algunas capacidades computacionales que se encuentran en el borde de la red de IoT. El borde de la red en una arquitectura IoT está compuesto por los *gateways* IoT, *smartphones* y algunos dispositivos embebidos que presentan características limitadas de computación.

Fog computing

A diferencia de *edge computing*, esta tiene su origen en IoT para cubrir las necesidades de almacenamiento y procesamiento de los dispositivos IoT. Por ello, *fog computing* se asocia más a IoT en lugar de *edge computing* asociado más con comunicaciones móviles. *Fog computing* está compuesta por nodos *fog* localizados entre los dispositivos IoT y el *cloud* [122]. Los nodos fog son dispositivos que ofrecen capacidades computaciones, almacenamiento y servicios de red como por ejemplo *gateways* IoT, enrutadores, *smartphones*, o servidores que se encuentra en la misma *Local Area Network* (LAN). La amplia gama de dispositivos para *fog computing* posibilitan la creación de servicios distribuidos geográficamente, por lo que es conocido como una tecnología altamente distribuida y heterogénea [108]. En la literatura actual, *fog computing* es considerado como una extensión de *cloud computing*

principalmente debido que tiene el objetivo de extender las capacidades para el procesamiento, almacenamiento y análisis de datos. Por ello, esta extensión es conocida como *Cloud Continuum* [123] o *IoT-Fog-Cloud Continuum* [124], incluso como Edge-Cloud Continuum [125].

Dado que algunos dispositivos usados como nodos *fog* son los dispositivos usados en el borde de la red, *edge computing* puede ser visto como un caso particular de *fog computing*, Figura 2.26. Para el contexto de la tesis, *fog computing* será utilizado como termino general al referirnos a la computación cercana a la fuente de datos IoT, y solo cuando sea necesario se especificará con el termino *edge computing*.

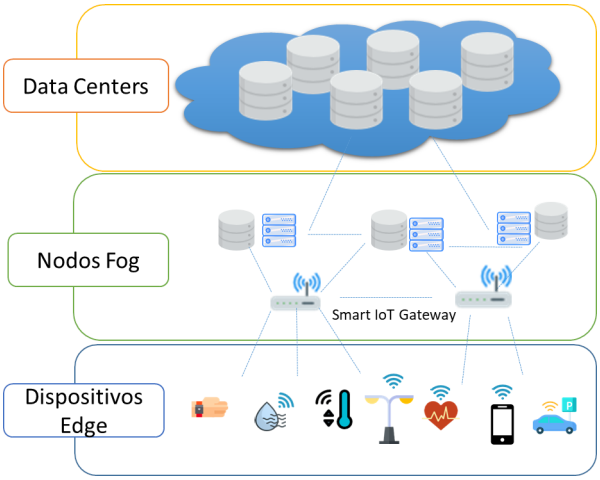


Figura 2.26: Fog Computing y Edge Computing

Fog computing provee importantes beneficios para manejar el Big Data generado por IoT. El procesamiento realizado en el borde de la red, también conocido como *edge analytics*, permite reducir la alta redundancia de datos IoT. El uso de tareas de pre-procesamiento de datos como el filtrado, limpieza de datos, extracción de variables, comprensión y técnicas de reducción de las dimensiones de datos en el borde han comprobado ser efectivas en la reducción del volumen de los datos que son enviados al *cloud* y posteriormente son almacenados [126]. De la misma manera, el uso de técnicas para fusionar varios flujos de datos en uno solo o técnicas para enviar los datos solo cuando se detectada anomalías reducen la frecuencia con la que se generan los datos IoT [127]. Además, el manejo de la interoperabilidad técnica, semántica y sintáctica en nodos fog (*gateway* IoT) reduce la heterogeneidad de IoT [128]. Resumidamente, *fog computing* permite reducir la latencia, el consumo de ancho de banda, espacio para almacenamiento y a su vez provee importantes ventajas como mejorar el QoS, soportar la

movilidad, interoperabilidad, y conocimiento de la ubicación de los dispositivos [122]. La Figura 2.27 muestra una comparativa de las características entre *edge*, *fog* y *cloud computing*.

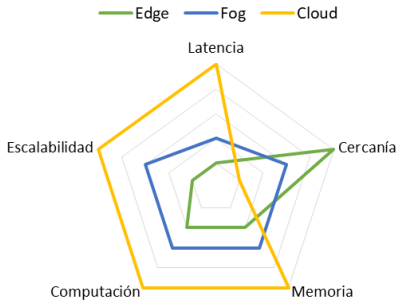


Figura 2.27: Comparativa entre edge, fog y cloud computing

Arquitectura IoT con fog computing

La arquitectura de *fog computing* se presenta como sistema multinivel capaz de extender las capacidades de procesamiento, almacenamiento y red desde el *cloud* al borde de la red. La arquitectura de *fog computing* debe facilitar la interoperabilidad, seguridad, escalabilidad, programabilidad, confiabilidad, disponibilidad, y agilidad [129]. Estas características potencializaran el ecosistema IoT permitiendo el desarrollo de servicios innovadores y rentables para un modelo de negocios. Actualmente, iniciativas como *OpenFog Consortium* buscan presentar una arquitectura abierta de *fog computing* considerando las dichas características. El *OpenFog Consortium* integra a organizaciones de investigación y vendedores de tecnología para promover terminales y servicios basados en estándares. Además de esta iniciativa, varias arquitecturas han sido propuestas por diferentes grupos de investigación. Por lo general, las arquitecturas de IoT que utilizan *fog computing* propuestas están compuestas por 3 niveles, el nivel de dispositivos inteligentes, *fog computing*, y el *cloud computing*. El *Internet of Things World Forum* (IoTWF) de 2014 propuso una ampliación de la arquitectura de referencia de IoT de 3 capas en una arquitectura de 7 capas, en la que se incluye *fog computing*, Figura 2.28 [130].

Los nodos *fog* presentan recursos computacionales limitados y con una capacidad que varía entre ellos, por lo que es un reto enorme desarrollar una plataforma y un modelo de servicio en esta capa. Actualmente, las tecnologías ligeras de virtualización están siendo utilizadas por su eficiencia en el uso de los recursos disponibles. Principalmente, la tecnología basada en contenedores de servicios emerge como la mejor opción para ser adaptada

Fuente: IoTWF, 2014 [130]

Figura 2.28: Arquitectura IoT con *Fog Computing*

en los nodos *fog*. Esta tecnología provee virtualización y aislamiento de las aplicaciones o servicios a nivel de sistema operativo evitando la virtualización de *hardware* y *drivers* [131]. Un claro ejemplo de uso de la tecnología de contenedores es el presentado en [132] [106] donde se implementa un *gateway* IoT utilizando micro-servicios basados en la virtualización de contenedores Docker para optimizar los recursos. Conjuntamente con esta visión del uso de tecnologías de virtualización ligera, plataformas como Apache Edgent o ParaDrop han sido desarrollados para proveer procesamiento de datos en el borde [133] [134]. Apache Edgent es una plataforma para realizar procesamiento de datos en tiempo real en dispositivos ubicados en el borde, mayormente utilizado para integrarse con la plataforma de IBM Watson IoT en el Cloud [135]. Sin embargo, aún se espera el desarrollo de más plataformas adaptadas para el uso en *fog computing* [136].

2.5 Conclusiones

Las secciones previas detallan las tecnologías involucradas para el manejo de Big Data, al igual que provee una concisa descripción de la visión de IoT y sus características generales. Sin embargo, la integración de los dos conceptos tecnológicos requiere de nuevas estrategias que aprovechen al máximo las características de cada una de ellas. Estas estrategias son plasmadas en el desarrollo de arquitecturas de Big Data. Las arquitecturas propuestas estructuran una metodología que agrupa las nuevas tecnologías,

plataformas, modelos, herramientas, librerías y lenguajes de programación para obtener los beneficios esperados del Big Data.

Las arquitecturas de Big Data consideradas como referencia carecen de un análisis particular de los datos que se generan en entornos IoT. Por ello, estas arquitecturas no se adaptan fácilmente a los entornos IoT. En la literatura actual, existen pocas arquitecturas de Big Data especificas para algunos entornos IoT por lo que la adopción de los conocimientos para integrar IoT con Big Data deriva en pérdidas de tiempo y recursos.

Por otro lado, *cloud computing* ha sido empleado como principal tecnología habilitadora para la integración de Big Data con IoT. Si bien el uso de los modelos de servicios de *cloud computing* han sido utilizados ampliamente para soportar la gestión del Big Data, estos han demostrado limitaciones en el manejo del Big Data generador por IoT. Las tecnologías de *fog computing* presentan un gran potencial para cubrir las limitaciones del *cloud computing*. Sin embargo, *fog computing* aún es una tecnología en desarrollo. Sin lugar a dudas las tecnologías de *fog computing* serán un aliado importante para el manejo del Big Data generado por IoT en un futuro inmediato.

Capítulo 3

Especificación de la Arquitectura

3.1 Introducción

La especificación de la arquitectura de Big Data para IoT responde a la falta de una arquitectura de referencia de Big Data que considere las características particulares de los datos generados por IoT, y que hoy por hoy limita la adopción de las metodologías de Big Data por parte de las partes interesadas. En tal virtud, este capítulo describe una arquitectura para la gestión del gran volumen de datos generados por IoT en sus diversos dominios de aplicación.

La arquitectura tiene un carácter genérico para que se pueda adaptar a los requerimientos particulares de los distintos dominios de aplicación. De esta manera, la arquitectura puede ser usada como modelo para diseñar la arquitectura de los sistemas para el soporte del Big Data de IoT con sus respectivos requisitos particulares. La arquitectura va dirigida a los administradores de sistemas IoT y arquitectos de sistemas IoT; así como, al público en general que quiere entender la integración de las metodologías y técnicas de Big Data con los ecosistemas de IoT.

Además, la arquitectura no define los proveedores de plataformas, tecnologías o protocolos de comunicación para implementar las funcionalidades

específicas de la arquitectura, por lo que esta se puede adaptar a futuros entornos y plataformas. Además, la arquitectura guarda estrecha relación con las diferentes estrategias para la gestión del Big Data, arquitecturas de referencia de IoT, y tecnologías computacionales actuales; las mismas que fueron analizadas en el capítulo de Estado del Arte.

En la primera sección del capítulo se analizan los requerimientos genéricos de los sistemas de IoT para la gestión del Big Data de IoT, los mismos que son divididos en requerimientos funcionales y no funcionales. Posteriormente, se presenta una visión general de la arquitectura donde se describe la perspectiva y consideraciones de diseño empleadas para el desarrollo de la arquitectura. Las siguientes secciones describen la arquitectura desde los puntos de vista conceptual, funcional, de procesos y de implementación para abordar todos los requerimientos e intereses de las partes interesadas.

3.2 Requerimientos

IoT presenta requerimientos funcionales para el manejo de Big Data que pueden variar entre los distintos dominios de aplicación. Sin embargo, existen algunos requerimientos comunes que pueden ser utilizados para diseñar una arquitectura genérica y adaptable a los requerimientos particulares de cada dominio de aplicación. Los requerimientos comunes para el diseño de sistemas de IoT y los requerimientos de IoT para soportar datos masivos son enumerados por la ITU-T en sus recomendaciones Y.2066 [137] y Y.4114 [138], respectivamente. Estas recomendaciones son la base en el proceso de estandarización de las funcionalidades, API y protocolos relacionados con Big Data e IoT que aún se encuentran en estudio por parte de las entidades ITU-T y NIST. A continuación los requerimientos funcionales son enumerados, los cuales están basados en las recomendaciones de la ITU-T en su mayoría y alineados con los objetivos de la tesis.

R1. Soporte de conexión con las múltiples fuentes de IoT

La arquitectura debe ser capaz de conectarse con las distintas fuentes de datos IoT. Las fuentes de IoT en este caso pueden ser dispositivos IoT (sensores), *gateways* IoT, plataformas IoT, otros sistemas de aplicaciones IoT, datos abiertos y espacio de datos virtuales IoT. En la mayoría de los casos, las fuentes de IoT exponen API que facilitan la extracción de los datos. Por lo que la arquitectura debe gestionar estas API para la extracción de datos de las múltiples fuentes. Otras fuentes de datos como sistemas propietarios también deben ser considerados porque estos sistemas aportan información de contexto que puede enriquecer aún más los datos de las fuentes de IoT.

R2. Soporte para la conversión de formatos de datos de IoT

La arquitectura debe ser capaz de realizar una normalización u homogenización del formato de los datos para que los distintos componentes funcionales de la arquitectura sean interoperables entre sí. Dado que los datos pueden llegar de múltiples fuentes, es necesario considerar un mecanismo que permita convertir los formatos típicamente utilizados por las fuentes IoT (XML, JSON, o *Comma Separated Values* (CSV)) a formatos que permita el almacenamiento y procesamiento del Big Data de IoT de manera eficiente.

R3. Procesamiento de datos de IoT en tiempo real

La arquitectura debe ser capaz de soporta el tratamiento de los datos en tiempo real debido principalmente a que IoT presenta una alta frecuencia de generación de datos. El tratamiento de datos debe estar enfocado en la selección de variables, agregación, compresión, y limpieza de datos para reducir la frecuencia y la redundancia de los datos generados por IoT.

R4. Procesamiento de datos de IoT tipo *batch*

La arquitectura debe ser capaz de realizar el tratamiento de los datos históricos recolectados para extraer información útil que ayude a describir la situación actual y que sirvan para generar modelos predictivos. El procesamiento tipo *batch* es ampliamente utilizado en los sistemas de Big Data para cumplir con los objetivos de generar análisis descriptivos y preparar los datos para el análisis predictivo.

R5. Soporte de almacenamiento

La arquitectura debe ser capaz de almacenar los datos de IoT durante todo el ciclo de vida del Big Data de manera temporal o permanentemente para garantizar una reutilización eficaz de los datos. La reutilización de los datos permitirá generar mejores análisis de datos y posibilitar un enriquecimiento de datos con múltiples fuentes para mejorar la toma de decisiones. Además, la arquitectura debe considerar el almacenamiento localmente o remotamente dependiendo de las capacidades de la tecnología empleada para almacenar los datos.

R6. Análisis de datos

La arquitectura debe ser capaz de extraer información del Big Data de IoT soportando al menos los tipos de análisis descriptivo, predictivo y de diagnóstico. La ejecución de estos tipos de análisis debe poder ser realizada a través de modelos estadísticos y algoritmos de IA. Principalmente, el uso de algoritmos de IA permitirá el desarrollo de innovadores servicios que aporten mayores beneficios a los usuarios de los sistemas IoT.

R7. Soporte en la generación y distribución de servicios

La arquitectura debe ser capaz de soportar la generación de servicios basados en la información útil extraída mediante el análisis del Big Data. Estos servicios deben poder ser fácilmente accesibles a través de la exposición de API para que usuarios finales, desarrolladores de software, y otros sistemas se beneficien de ellos.

R8. Visualización de resultados de los análisis de datos

La arquitectura debe ser capaz de mostrar los resultados de los análisis de datos en una forma que resulte de fácil interpretación para la toma de decisiones. Una de las formas de facilitar la interpretación de resultados es a través de representaciones visuales de los datos.

R9. Gestión de los sistemas IoT

La arquitectura debe ser capaz de permitir la gestión de las funcionalidades para garantizar una monitorización de las operaciones de los sistemas IoT. La arquitectura debe considerar la monitorización de sus componentes funcionales y procesos para garantizar los acuerdos de niveles de servicio en todos los dominios de la arquitectura.

R10. Soporte múltiples tecnologías computacionales

La arquitectura debe ser capaz de poder implementarse empleando las tecnologías computacionales y sus modelos de servicio actuales, como por ejemplo el uso de *cloud computing*, *fog computing*, y *edge computing*. Principalmente, este requerimiento guarda relación con la extracción de valor de los datos de IoT en etapas tempranas de generación con el fin de mejorar los tiempos de respuesta de los sistemas IoT.

R11. Seguridad

La arquitectura debe ser capaz de proteger el Big Data generado por los entornos IoT durante su tiempo de vida. Para ello, la arquitectura debe considerar mecanismos para garantizar la privacidad, confidencialidad, e integridad de los datos. La arquitectura debe soportar mecanismos de autenticación basados en usuario y contraseña o a través de certificados digitales. Además, la arquitectura debe considerar el uso de mecanismos de anonimización de datos para mantener bajo privacidad la identidad de los usuarios, el cual puede ser útil en aplicaciones de IoT críticas con alta sensibilidad a los datos, como por ejemplo en el ámbito de la salud y bienestar.

R12. Soberanía de Datos

La arquitectura debe considerar que los propietarios de los datos requieren controlar en todo momento sus datos. Esto quiere decir que los propietarios de los datos necesitan conocer cómo se usan sus datos y para qué se usan sus datos. Este requerimiento está acorde con las nuevas necesidades de privacidad y seguridad de la información cuando los datos son gestionados por terceros.

Por otro lado, los requerimientos no funcionales permiten apreciar la calidad de la arquitectura diseñada. Estos requerimientos están relacionados con la implementación y operación de la arquitectura. Los requerimientos no funcionales también son considerados en las recomendaciones de la ITU-T Y.2066 [137] para el correcto funcionamiento de IoT. Adicionalmente, el estándar de calidad de software ISO/IEC 25010 [139] es empleado para extraer los atributos de calidad que serán considerados en el desarrollo de esta arquitectura. A continuación, los requerimientos no funcionales son enlistados basados en los dos estándares mencionados.

R13. Escalabilidad

La arquitectura debe ser capaz de manejar el alto volumen de datos que se espera de los dominios de aplicación de IoT, sin que esto pueda interferir en su normal funcionamiento. Además, los costos y eficiencia del sistema no deben sufrir impacto alguno con el incremento de las fuentes de IoT y el consiguiente aumento del volumen de los datos generados.

R14. Eficiencia

La arquitectura debe ser capaz de asegurar su desempeño en cualquier momento durante sus operaciones. El procesamiento del Big Data no debe ser afectado con el incremento del volumen de datos, es decir se debe garantizar el procesamiento eficiente de un gran volumen de datos.

R15. Alta Disponibilidad

La arquitectura debe ser capaz de mantener disponible sus funcionalidades operaciones y servicios cuando se lo requiera. En caso de una operación anómala, se debe garantizar el funcionamiento de los sistemas ya sea a través de componentes redundantes o con la replicación de los datos con el fin de que estén disponibles cuando se los requiera.

R16. Adaptabilidad

La arquitectura debe ser capaz de adaptarse a los entornos determinados de tecnologías y uso sin que esto implique degradación en su eficiencia. La arquitectura debe ser lo suficientemente flexible para que se pueda adaptar a futuras metodologías, técnicas y tecnologías de Big Data e IoT; por lo que es necesario considerar la modularidad de sus funcionalidades.

3.3 Visión general de la arquitectura

La arquitectura está basada en las recomendaciones recientes propuesta por el grupo de trabajo del NBD-PWG para el diseño de una arquitectura de Big Data interoperable [93]. Esta arquitectura es empleada como referencia para establecer los componentes y roles que intervienen en la gestión del Big Data de IoT. De esta manera, la arquitectura garantiza que será interoperable con futuras tecnologías y *frameworks* para la gestión del Big Data.

Además, el diseño de la arquitectura sigue el modelo BDLM del ciclo de vida del Big Data principalmente porque permite una reusabilidad de los datos mediante el almacenamiento de los datos en todas las etapas del ciclo de vida [52]. El modelo BDLM es considerado para establecer las principales funcionalidades que requiere ser soportada por la arquitectura durante la transformación de los datos y el correspondiente proceso de extracción de información útil.

Asimismo, la arquitectura utiliza las directrices planteadas en el estándar ISO/IEC/IEEE 42010:2011 para la descripción de la arquitectura [140]. De esta manera, la conceptualización de la arquitectura puede ser fácilmente entendible por los arquitectos de sistemas de IoT. Además, este estándar ha sido empleado para describir arquitecturas de referencia en el ámbito de IoT como la IIRA y la AIOTI HLA, lo que permitirá un mejor entendimiento a arquitectos que están familiarizados con los dominios IoT. Por otra parte, el uso del estándar reduce la complejidad para expresar los conceptos, principios y procedimientos con el fin de que el público en general comprenda y utilice la arquitectura como guía durante la implementación de sus sistemas IoT para el soporte de Big Data.

De acuerdo al estándar ISO/IEC/IEEE 42010:2011, la arquitectura está dividida en vistas las cuales son definidas como *"un producto de trabajo que expresa la arquitectura de un sistema desde la perspectiva de interés específicos del sistema."* [140]. En este caso las vistas resuelven uno o más requerimientos de las partes interesadas. Cada vista se rige por un punto de vista, el cual se define como *"producto de trabajo que establece las conven-*

ciones para la construcción, interpretación y uso de vistas de arquitectura para enmarcar intereses específicos del sistema." [140]. De esta manera, la arquitectura es descrita a través de los puntos de vista conceptual, funcional, de procesos, y de implementación, los cuales fueron seleccionados para satisfacer los requerimientos planteados en la sección 3.2.

El punto de vista conceptual se enfoca en la descripción de los roles principales y la relación entre ellos para tener un mejor entendimiento de la arquitectura. Este punto de vista enmarca los requerimientos (R3 – R7 y R12). El punto de vista funcional se enfoca en las funcionalidades de los componentes de la arquitectura, la comunicación entre ellos y las interfaces necesarias para cada componente. Este punto de vista enmarca los requerimientos funcionales (R1 – R12). El punto de vista de procesos se enfoca en describir los procesos principales de la arquitectura para soportar el Big Data generado por IoT. Este punto de vista enmarca los requerimientos funcionales (R1 – R9). El punto de vista de implementación se enfoca en las tecnologías computacionales empleadas para implementar las funcionalidades de la arquitectura presentadas en la vista funcional. Este punto de vista enmarca los requerimientos relacionados con el uso de las tecnologías computacionales actuales (R10) y los requerimientos no funcionales (R13 – R16).

Cada punto de vista contiene su respectiva vista arquitectónica desde la cual se expresa la arquitectura por medio del uso de patrones arquitectónicos para un mejor entendimiento. En las siguientes secciones se detallan estas vistas arquitectónicas. La Figura 3.1 muestra las vistas de la arquitectura y su relación con los dominios de aplicación de IoT.

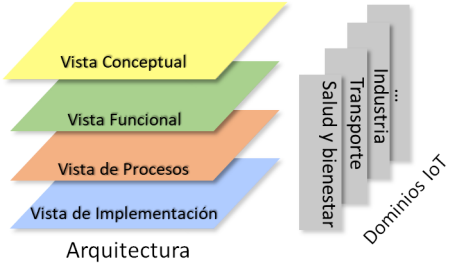


Figura 3.1: Puntos de vista de la arquitectura y su relación con los dominios de aplicación de IoT

3.4 Vista conceptual

La vista conceptual describe y categoriza los diferentes roles que intervienen en la gestión del Big Data de IoT. Además, la vista conceptual especifica las actividades principales e interacciones entre los diferentes roles. Cada rol abstrae las tareas principales que debe soportar un sistema de IoT para gestionar el Big Data. Esta vista está basada en las especificaciones de la ITU-T Y.4114 [138], donde se especifican los roles que debe soportar una arquitectura IoT. Además, esta vista sirve como guía para un mejor entendimiento del resto de vistas de la arquitectura.

Los roles necesarios para soportar las funcionalidades requeridas son: propietario de datos, proveedor de datos, proveedor de habilitadores tecnológicos de Big Data, proveedor de servicios, y el consumidor de servicios. La Figura 3.2 muestra los roles y la interacción que se produce entre ellos.

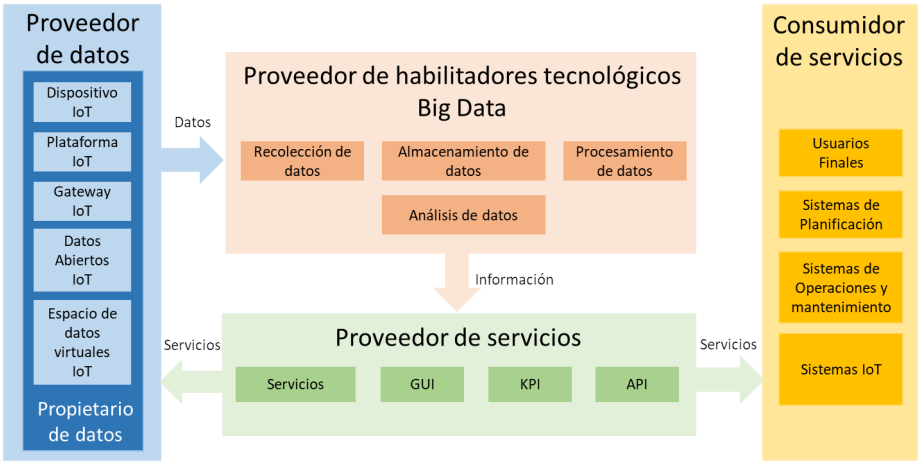


Figura 3.2: Diagrama de la vista conceptual de la arquitectura

3.4.1 Propietario de datos

El propietario de datos es la persona o sistema que tenga los derechos legales de los datos. En muchos de los casos, el propietario de datos también asumirá el rol de proveedor de datos. En otros, el propietario de datos autorizará el uso de sus datos bajo una política de utilización de datos, como por ejemplo utilizando la regulación general de protección de datos (GDPR, por sus siglas en inglés, *General Data Protection Regulation*). En algunos modelos de negocio de IoT, los servicios relacionados con el Big Data generado por IoT tiene como beneficiario final el propietario de datos,

por ejemplo en SenaaS, donde el dueño de los datos autoriza la utilización de datos a cambio de un servicio relacionado con la explotación de los mismos.

3.4.2 Proveedor de datos

El proveedor de datos comparte los datos IoT con el proveedor de los habilitadores tecnológicos de Big Data. El proveedor de datos, que a la vez puede ser propietario de datos, expone API para habilitar una conexión con el proveedor de datos y transferir los datos de IoT. En la mayoría de casos, el proveedor de datos es una fuente de generación de datos IoT, en este caso, los dispositivos IoT (sensores, actuadores), *gateways* IoT, plataformas IoT, datos abiertos de IoT y espacio de datos virtuales IoT. Además, el proveedor de datos podrá usar los servicios que se generen como parte del análisis de datos para mejorar la eficiencia de sus sistemas.

3.4.3 Proveedor de habilitadores tecnológicos de Big Data

El proveedor de habilitadores tecnológicos de Big Data recibe los datos masivos del proveedor de datos para almacenarlos, procesarlos y analizarlos de manera eficiente. Para ello, el proveedor de habilitadores tecnológicos de Big Data implementa parte de las funcionalidades de la arquitectura para la gestión del Big Data generados por IoT. El proveedor no solo se preocupa de las funcionalidades tecnológicas y la exposición de los habilitadores tecnológicos sino también de proveer un ambiente eficiente, escalable, tolerante a fallos, redundante, y seguro para gestionar el Big Data generado por IoT. Además, este rol implementa los procesos relacionados con la recolección de datos, almacenamiento de datos, procesamiento de datos en tiempo real y tipo *batch*, y el análisis de datos a través de modelos estadísticos y uso de IA para la generación de información útil. La información útil en forma de modelos estadísticos, predictivos, y analíticas descriptivas son enviadas al proveedor de servicios para que gestione su uso y distribución.

3.4.4 Proveedor de servicios

El proveedor de servicios recibe los resultados del análisis de Big Data, transforma estos resultados en forma de servicios y los distribuye a los consumidores de servicios. Estos servicios contienen información útil y de fácil interpretación para que los consumidores de servicios puedan tomar decisiones más acertadas. Además, el proveedor de servicios se encarga de la gestión de KPI y API, y el desarrollo de interfaces gráficas de usuario (GUI,

del inglés *Graphical User Interface*), aplicaciones móviles y aplicaciones web.

3.4.5 Consumidores de servicios

El consumidor de servicios recibe los resultados del análisis de Big Data en forma de servicios para soportar su toma de decisiones. Los consumidores de servicios pueden ser usuarios finales, otros sistemas IoT, sistemas de operaciones y mantenimiento, y sistemas de planificación en el ámbito industrial; incluso los proveedores de datos pueden llegar a ser consumidores de servicios en el caso en que los servicios puedan ayudar a mejorar la eficiencia de sus sistemas.

3.5 Vista funcional

La vista funcional especifica las funcionalidades y características que se implementarán para la gestión del Big Data de IoT. Las funcionalidades y características están agrupadas en dominios funcionales. La arquitectura presenta cuatro dominios funcionales: dominio de aplicaciones, dominio operacional, dominio de información y dominio IoT. Estos dominios funcionales guardan estrecha relación con los requerimientos funcionales descritos en la sección 3.2. Cada dominio funcional está compuesto por un conjunto de funciones comunes. La implementación de los componentes de cada dominio funcional dependerá de la complejidad de los sistemas, siendo en algunos casos necesario omitir algunos componentes. La Figura 3.3 muestra los dominios funcionales de la arquitectura y sus componentes específicos. En las siguientes subsecciones se describen los dominios funcionales y sus respectivos componentes.

3.5.1 Dominio de IoT

El dominio de IoT corresponde a todas las fuentes de datos cuyo origen son dispositivos IoT. Las principales fuentes de IoT son los dispositivos IoT (sensores, actuadores), *gateways* IoT, plataformas IoT, datos abiertos IoT y espacio de datos virtuales IoT.

Los dispositivos IoT son todos aquellos dispositivos capaces de recolectar la información que se genera en su entorno, transformar esta información en señales digitales y transmitirlas a Internet en forma de datos. Los dispositivos IoT principalmente están equipados con sensores, actuadores y microcontroladores para poder realizar las funcionalidades descritas. Un ejemplo de los dispositivos IoT son los llamados *wearables* (ropa, relojes inteligen-

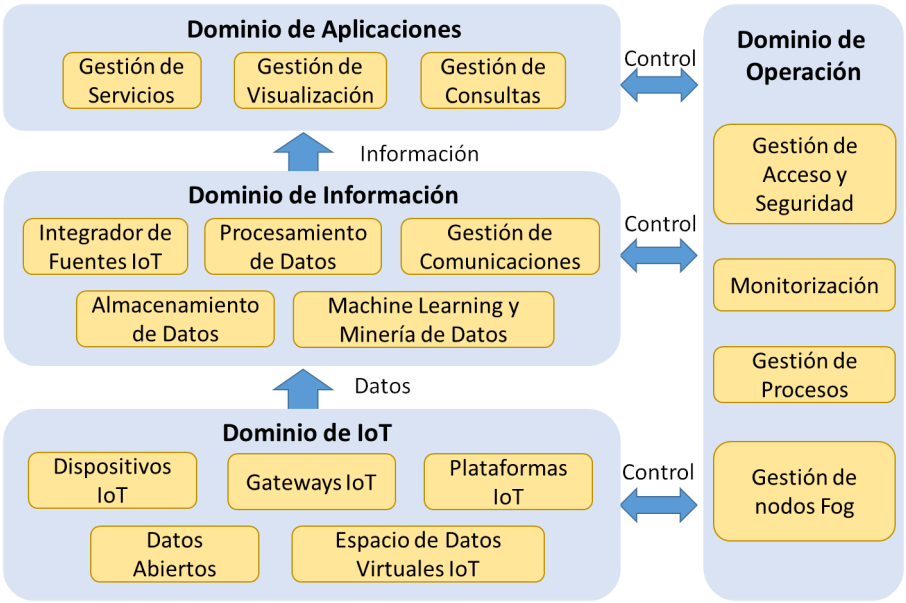


Figura 3.3: Dominios funcionales y sus componente para el manejo de Big Data generado por IoT

tes, muñequeras, entre otros) que se han convertido en los dispositivos más populares destinados al monitorización de la salud [33]. Los dispositivos IoT utilizan protocolos y tecnologías de comunicación inalámbricas para el envío de los datos a Internet. Algunas de las tecnologías inalámbricas utilizadas por los dispositivos IoT son Bluetooth, ZigBee, WiFi, 6LowPAN, LoRA y tecnologías de telefonía móvil como LTE. Además, los dispositivos IoT emplean protocolos de aplicación ligeros para el envío de datos. Algunos de los protocolos utilizados por los dispositivos IoT pueden ser MQTT y CoAP.

Los *gateways* IoT son dispositivos intermedios con capacidades de red para re-direccionar los paquetes enviados por los dispositivos IoT a Internet. Algunos de estos *gateways* IoT no solo brindan las funcionalidades de red, sino también presentan funcionalidades para procesar, almacenar, e incluso analizar los datos recolectados por los dispositivos IoT. Sin embargo, las capacidades computacionales de los *gateways* IoT son muy limitadas para el soporte de grandes volúmenes de datos. En los últimos años, la proliferación de teléfonos móviles inteligentes, los cuales están equipados con tecnologías inalámbricas como Bluetooth y WiFi, ha permitido que estos sean utilizados como *gateways* IoT. Los dispositivos IoT *wearables* son los que mayormente utilizan a los teléfonos móviles como *gateways* IoT para el envío de datos

a Internet. Además, los teléfonos móviles inteligentes incorporan sensores que pueden ser empleados para recolectar datos y enviarlos a Internet.

Las plataformas IoT proveen las funcionalidades y capacidades para gestionar los dispositivos IoT, permitir una comunicación bidireccional entre los dispositivos IoT y la aplicación IoT, y sobre todo recolectar los datos generados por los dispositivos IoT. Las plataformas IoT también son consideradas como *middleware* IoT debido a que funcionan como un intermediario entre los dispositivos y las aplicaciones. Los dispositivos IoT comparten sus datos generados a través de varias API de comunicación facilitadas por plataformas IoT. Además, las plataformas IoT proveen funcionalidades para gestionar la seguridad de los dispositivos IoT y los usuarios de las aplicaciones IoT. Algunos ejemplos de plataformas IoT son: FIWARE, UniversAAL, SOFIA2, OpenIoT, entre otras.

Los datos abiertos de IoT son conjuntos de datos libremente disponibles para que cualquier persona pueda reutilizarlos. Los datos proceden de dispositivos IoT pertenecientes a entidades gubernamentales y organizaciones internacionales como por ejemplo los ayuntamientos de ciudades, agencias estatales de meteorología, etc. Los datos abiertos son expuestos a la comunidad a través de varias API de manera gratuita pero con protección o restricciones para evitar la comercialización de estos conjuntos de datos.

Los espacios de datos virtuales IoT son almacenes privados que contienen conjuntos de datos de IoT. Principalmente, estos espacios de datos recopilan datos masivos de IoT producidos a nivel industrial por varias empresas interesadas y que conforman el espacio de datos virtuales. Unos de los ejemplos de los espacios de datos virtuales es el que se forma al emplear la arquitectura de referencia *Industrial Data Space* (IDS) para la compartición de datos en un entorno seguro en la industria. A diferencia de los datos abiertos de IoT, estos datos presentan mayores restricciones para explotar los datos debido a que las empresas involucradas en la industria valoran a los datos como un activo más, por lo que no quieren perder control sobre el uso que se dé a sus datos. El acceso a estos datos puede darse a través de varias API o en el caso de IDS, empleando un conector seguro para integrarse al espacio de datos virtuales.

3.5.2 Dominio de información

El dominio de información tiene como objetivo gestionar el ciclo de vida del Big Data generado por las múltiples fuentes de IoT. Sus funciones principales son las de gestionar la integración con el dominio IoT para la extracción de datos, almacenar y procesar los datos masivos generados por IoT y analizar el Big Data para extraer información útil. El dominio de información presenta cinco componentes: integrador de fuentes IoT, almacenamiento de datos, procesamiento de datos, gestión de comunicaciones, y *machine learning* y minería de datos.

Integrador de fuentes IoT

El componente para la integración de fuentes realiza dos funciones específicas: la gestión de conexiones a las fuentes IoT y la conversión de los formatos de datos. La gestión de conexiones a las fuentes es realizada para facilitar la conexión con las fuentes de datos de IoT (sensores, *gateways* IoT, plataformas IoT, sistemas de aplicaciones). Generalmente, el mecanismo publicación/suscripción es el empleado por las fuentes de datos de IoT. Por ello, el módulo gestiona el proceso de suscripción al servidor de mensajería de la fuente de datos. Otro mecanismo empleado son las API bien definidas por parte de las fuentes de datos. El módulo gestiona las funcionalidades necesarias para explotar las API de la fuente, que en su mayoría suelen ser del tipo API REST. Por otro lado, la conversión de los formatos de datos es realizada con el fin de estandarizar los datos para mantener la compatibilidad entre bloques. Esta conversión de formato busca realizar una estandarización para mantener la compatibilidad entre capas. Además, el componente gestiona los mecanismos para realizar la validación de datos a través de tareas de pre-procesamiento enfocadas en reducir ruido, imperfecciones (valores nulos o incompletos) y valores atípicos (valores inconsistentes). Adicionalmente, el componente provee mecanismos de compresión de datos para la optimización en el almacenamiento de los datos.

Procesamiento de datos

El componente de procesamiento de datos maneja el volumen y la velocidad de los datos de IoT empleando los recursos computacionales para tratar los datos de manera eficiente, escalable y confiable. Para ello, el componente hace uso de la computación paralela y distribuida. Principalmente, el componente proporciona las plataformas que soportan el modelo de programación *MapReduce* y DAG para procesar los datos *batch* (volumen) y en tiempo real (velocidad), respectivamente.

Además, el componente provee funcionalidades para la limpieza de datos, optimización de datos y enriquecimiento de datos. La limpieza de datos es realizada con el fin de reducir el ruido, imperfecciones (valores nulos o incompletos) y valores atípicos (valores inconsistentes). La optimización de datos es realizada para reducir la cantidad de datos que serán procesados a través de los mecanismos de compresión de datos y reducción de redundancia de datos. El enriquecimiento de datos emplea mecanismos de fusión de los conjuntos de datos de dos o varias fuentes para agregar contexto a los datos recolectados IoT. Este enriquecimiento permite generar análisis de datos más precisos y ricos en contexto.

Almacenamiento de datos

El componente de almacenamiento de datos proporciona los mecanismos para mantener los datos durante el ciclo de vida del Big Data. Los mecanismos utilizados son los sistemas de ficheros distribuidos y sistemas de base de datos no relacionales NoSQL. Estos mecanismos garantizan la disponibilidad, tolerancia a fallos, escalabilidad y seguridad; así como, explotan eficientemente los recursos para almacenamiento. El almacenamiento de datos estará destinado a recopilar los datos de las fuentes de IoT en bruto, los datos procesados, las analíticas resultantes del procesamiento, los modelos de predicción y los resultados de predicción. Además, el almacenamiento puede ser de dos tipos: almacenamiento a largo plazo y almacenamiento a corto plazo. El almacenamiento a largo plazo permitirá una reusabilidad del Big Data para cumplir con el ciclo de vida del Big Data propuesto en el modelo BDLM. El almacenamiento a corto plazo hace referencia a un almacenamiento temporal y sujeto a políticas de gestión del dominio de operaciones y de acuerdo con la aplicación IoT.

Machine learning y minería de Datos

El componente de *machine learning* y minería de datos proporciona las funcionalidades para desarrollar modelos estadísticos y modelos de *machine learning* y *deep learning*. Para ello, el modulo provee plataformas y herramientas que ayudan a implementar los modelos empleando los datos procesados y almacenados de IoT. Además, el componente provee un entorno de desarrollo integral. Este entorno presenta una interfaz gráfica de usuario para que los desarrolladores o analistas de datos puedan realizar las tareas relacionadas con la explotación de datos de manera interactiva.

Gestión de comunicaciones

El componente gestión de comunicaciones facilita la comunicación entre los módulos de la arquitectura a través de un mecanismo de mensajería publicación/suscripción. Este mecanismo permite distribuir los datos entre las plataformas de Big Data empleadas por los módulos de procesamiento, almacenamiento, y *machine learning* y minería de datos. Además, este componente gestiona las comunicaciones para el envío de datos entre el dominio de información y el dominio de operación.

3.5.3 Dominio de operación

El dominio de operación agrupa las funcionalidades dedicadas a gestionar y controlar las operaciones del dominio de información, dominio de aplicación y del dominio de IoT. Sus funciones principales son las de gestionar, monitorizar y controlar los procesos relacionados con los dominios funcionales. El dominio de operación presenta cuatro componentes: gestión de acceso y seguridad, monitorización, gestión de procesos y gestión de nodos *fog*.

Gestión de acceso y seguridad

El componente de gestión de acceso y seguridad provee funcionalidades para controlar el acceso a las diversas funcionalidades de los dominios de información, de aplicación y de IoT. El control de acceso a las diferentes funcionalidades de la arquitectura se realizará través de la autenticación basada en usuario y contraseña o a través del uso de certificados digitales. Además, el componente provee los mecanismos para mantener la privacidad y confidencialidad de los datos gestionados en el dominio de información. La privacidad de los datos se garantiza a través del cifrado de los datos que son enviados entre el proveedor de datos y el proveedor de los habilitadores tecnológicos de Big Data; así como, para la comunicación entre el proveedor de los habilitadores tecnológicos de Big Data y el proveedor de servicios.

Monitorización

El componente de monitorización provee funcionalidades y mecanismos para facilitar la monitorización de las tecnologías y plataformas empleadas por los dominios de información y de aplicación. El principal objetivo es garantizar que las funcionalidades provistas por el resto de dominios estén disponibles y que no ocurran fallos. La detección de fallos y la posterior notificación al personal de operaciones y mantenimiento del sistema facilitará su solución con el fin de mantener los acuerdos de niveles de servicio.

Gestión de procesos

El componente de gestión de procesos provee funcionalidades para controlar la ejecución y correcto funcionamiento de los procesos que componen la arquitectura. El componente facilita la gestión de los *logs* generados durante la ejecución de los procesos. El registro de *logs* permite la detección de posibles fallos en los procesos y su posterior solución. La gestión de procesos está alineado con el mantenimiento de los acuerdo de niveles de servicio entre diferentes roles de la arquitectura.

Gestión de nodos fog

La gestión de nodos *fog* provee funcionalidades para la gestión y monitorización remota de los dispositivos utilizados como nodos *fog* (IoT *gateways*, y servidores). El objetivo principal es controlar la ejecución de los procesos remotos en los casos particulares de implementación donde se utiliza *fog computing* como tecnología computacional. El componente facilitará un entorno web para la gestión de los nodos *fog*. La gestión remota será posible únicamente para los casos en que los nodos *fog* siguen las especificaciones del estándar 1934 IEEE, basado en la arquitectura de referencia Open Fog [129]. De acuerdo con el estándar, los nodos *fog* utilizan virtualización ligera de contenedores como tecnología para el despliegue y desarrollo de aplicaciones y servicios.

3.5.4 Dominio de aplicaciones

El dominio de aplicaciones tiene el objetivo de utilizar la información extraída por el dominio de información y presentarlo a los usuarios en forma de aplicaciones y servicios. Para ello, este dominio presenta tres componentes: gestión de servicios, gestión de visualización y gestión de consultas.

Gestión de servicios

El componente provee funcionalidades para gestionar los servicios generados a través del uso de la información extraída por el dominio de información. Las funcionalidades principales se relacionan con el aprovisionamiento de recursos para el soporte de servicios, distribución de servicios a usuarios finales y la disposición de un catálogo de servicios para que los consumidores de servicios puedan acceder fácilmente a ellos.

Gestión de visualización

El componente está encargado de propiciar los mecanismos necesarios para generar y mantener una GUI para la visualización de los resultados. El servicio web es empleado como mecanismo de presentación de la información al usuario final en forma de gráficos. Este tipo de servicio es útil para presentar KPI en tiempo real con el fin de mejorar la toma de decisiones. El acceso a la GUI será controlada desde el dominio operacional a través de sus funcionalidades de seguridad.

Gestión de consultas

El componente gestiona la generación de consultas a los datos procesados para extraer información puntual de los datos. Esto es realizado a través de la explotación de los lenguajes de alto nivel y API provistos por los componentes de almacenamiento y procesamiento del dominio de información. De esta manera, los desarrolladores y administradores de los sistemas de planificación, sistemas de operación y mantenimiento u otros sistemas IoT puedan acceder a la información y beneficiarse de los resultados de la explotación del Big Data generador por el dominio de IoT.

3.6 Vista de procesos

La vista de procesos describe las interacciones entre los componentes de la arquitectura de Big Data para IoT y las actividades que se realizan para gestionar el Big Data durante su ciclo de vida. A continuación se describen los procesos que es capaz de soportar la arquitectura.

3.6.1 Recolección de datos

El proceso recolección de datos tiene la tarea de establecer conexión con el proveedor de datos IoT, extraer los datos, convertir el formato y cargar los datos en las plataformas de Big Data provistas por el proveedor de habilitadores tecnológicos de Big Data. La Figura 3.4 muestra el diagrama de flujo del proceso.

El subproceso de conexión con fuente IoT gestiona los conectores y sus configuraciones para establecer la conexión con las fuentes de IoT. Cada fuente IoT deberán exponer las configuraciones necesarias como por ejemplo: *Uniform Resource Locator* (URL), o dirección IP y puerto, parámetros de seguridad (usuario y contraseña o el uso de certificados digitales para

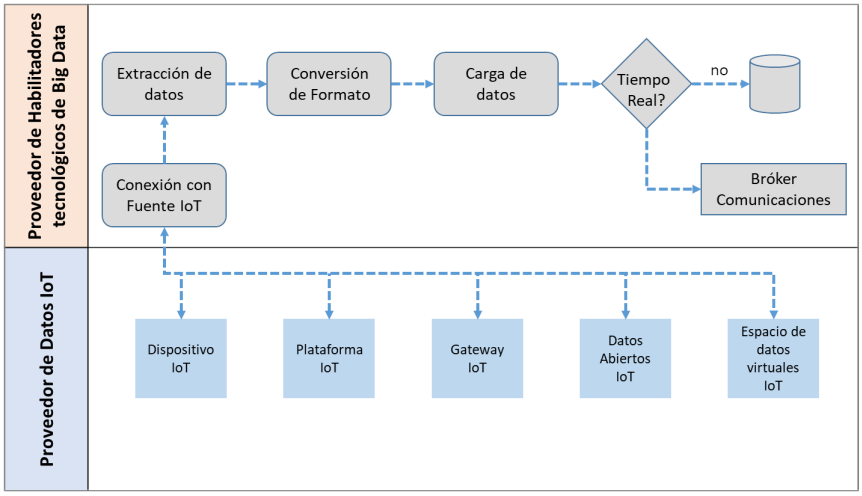


Figura 3.4: Diagrama de flujo del proceso de recolección de datos

conexiones *Secure Sockets Layer* (SSL)/ *Transport Layer Security* (TLS)) y el esquema de los datos a través de metadatos.

El subproceso de extracción de datos emplea los conectores para explotar las API de las fuentes de datos de IoT. Este subproceso podrá utilizar temporizadores para extraer los datos cada cierto tiempo para los casos de uso en los que se requiera.

El subproceso de conversión de formato utiliza el esquema de los datos especificado en los metadatos para convertir los datos a un formato específico. En caso de que la fuente no adjunte los esquemas de los datos a través de metadatos, el subproceso podrá inferir el esquema de los datos para ser empleado en la conversión de formato. Los esquemas deben ser almacenados conjuntamente con los datos para un posterior uso.

El subproceso de carga de datos cargará los datos a las plataformas para el almacenamiento y procesamiento de Big Data. Si los datos son en tiempo real, los datos se envían al sistema publicación/suscripción provistos por el componente de comunicación del dominio de información de la arquitectura para que la plataforma de procesamiento en tiempo real pueda acceder a estos datos. En caso de que no sean en tiempo real, los datos se enviarán directamente a la plataforma de almacenamiento.

Finalmente, cada subproceso registrará el éxito o no de cada uno de ellos en *logs* con el fin de que se pueda dar seguimiento a sus operaciones desde el dominio de operación de la arquitectura.

3.6.2 Extracción de Información

El proceso de extracción de información puede realizarse con datos en tiempo real y con datos históricos. Sin embargo, este proceso se enfoca en la extracción de la información de datos históricos, dejando la explicación del proceso de extracción de información en tiempo real como un proceso único y que se analiza en la subsección 3.6.3. El proceso de extracción de información es realizado por el proveedor de habilitadores tecnológicos de Big Data. El proceso está compuesto por las tareas de carga de datos, pre-procesamiento, y el análisis descriptivo y predictivo. La Figura 3.5 muestra el diagrama de flujo del subproceso.

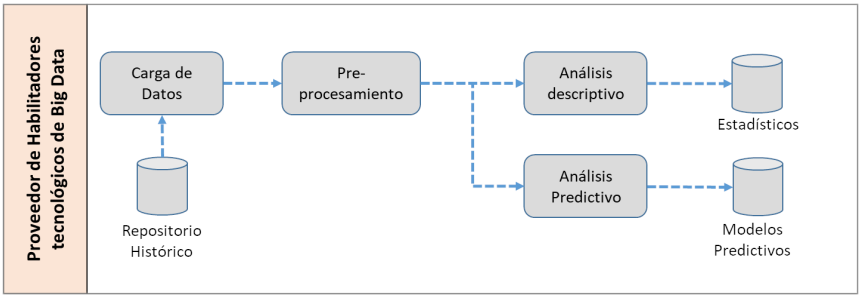


Figura 3.5: Diagrama de flujo del proceso de extracción de información

El subproceso de carga de datos está encargado de leer los datos del repositorio histórico donde se almacenaron los datos luego de cumplir el proceso de recolección de datos. Este subproceso requiere conocer el esquema de los datos. Para ello, el subproceso utiliza el esquema provisto por los metadatos o el esquema inferido en el proceso de recolección de datos.

El subproceso de pre-procesamiento de datos tiene las tareas de filtrar, limpiar, y agregar los datos. En algunos casos de uso, este subproceso se encarga de juntar los datos de varias fuentes para cumplir con las necesidades de la aplicación IoT. Para cumplir con estas tareas, el subproceso emplea el modelo de programación *MapReduce* y las capacidades computacionales provistas por el componente de procesamiento del dominio de información. Los datos resultantes de este procesamiento son enviados a un repositorio de resultados *batch*. Este nuevo repositorio con datos ya pre-procesados servirá para la extracción de información con los siguientes subprocesos. El modelo de datos y la tecnología a utilizarse para almacenar estos resultados del pre-procesamiento dependerá de la aplicación IoT.

El subproceso de análisis de datos realiza la extracción de información a través del uso de modelos estadísticos o utilizando algoritmos de *machine learning* o *deep learning*. El análisis de datos empleando modelos estadísti-

cos permite a la aplicación de IoT obtener información sobre el pasado y presente. Los estadísticos de tendencia como la mediana, media, máximo, mínimo, desviación estándar, varianza, moda, frecuencia, entre otros, serán usados en el desarrollo de varios KPI para brindar una visión de la situación actual de la aplicación de IoT. Además, el uso de técnicas estadísticas como la distribución de probabilidad, correlación de *Pearson*, test de hipótesis, entre otros, permite diagnosticar situaciones fuera de lo normal en la aplicación de IoT. Por otro lado, el uso de algoritmos de *machine learning* y *deep learning* permite predecir tendencias y detectar patrones. Este tipo de análisis facilitará la generación de innovadoras aplicaciones y servicios. Los resultados (estadísticos o modelos de predicción) son almacenados en un repositorio para que el proveedor de servicios pueda acceder a ellos y distribuirlos a los consumidores de servicios.

Finalmente, cada subproceso registrará el éxito o no de su tarea en *logs* con el fin de que se pueda dar seguimiento a sus operaciones desde el dominio de operación de la arquitectura.

3.6.3 Predicción en tiempo real

El proceso de detección y predicción emplea los modelos de predicción generados durante el proceso de extracción de información para detectar patrones o predecir estados en tiempo real. El proceso está compuesto por el bróker de comunicaciones, pre-procesamiento de datos en tiempo real, la detección y predicción de patrones y el almacenamiento de las predicciones. La Figura 3.6 muestra el diagrama de flujo del proceso.

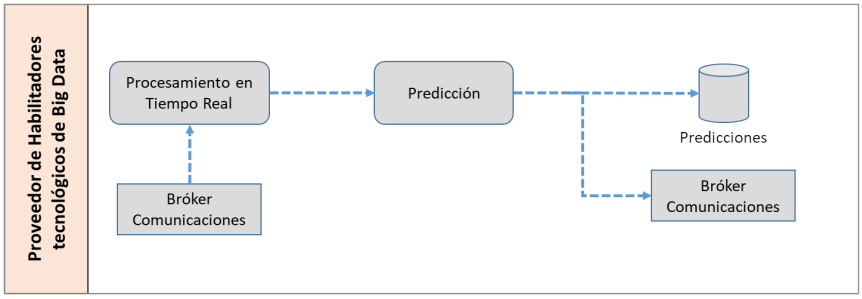


Figura 3.6: Diagrama de flujo del proceso de predicción en tiempo real

El subproceso de pre-procesamiento de datos en tiempo real realiza tareas de limpieza de datos, agregación de datos, y selección de variables. El subproceso utiliza los datos publicados en el bróker de comunicaciones de la arquitectura, el cual es provisto por el componente de comunicaciones del dominio de información de la arquitectura, para cargar los datos en tiem-

po real a la plataforma de procesamiento. La plataforma de procesamiento utiliza una ventana temporal que se desliza en el tiempo para realizar operaciones sobre los datos. Las operaciones están destinadas a limpiar los datos, reducir la frecuencia de datos y extraer variables. Una vez cumplidas estas tareas, las variables de los datos son seleccionadas para ser enviadas al subproceso de predicción para su evaluación.

El subproceso de predicción evalúa los datos enviados y retorna una predicción. Este subproceso utiliza el modelo generado por el proceso de extracción de información para evaluar los datos en tiempo real. El subproceso utiliza una API REST para recibir las solicitudes del tipo POST con un archivo adjunto (datos seleccionados). El modelo realiza la predicción y envía este resultado al bróker de comunicación y aun repositorio de predicciones. El proveedor de servicios se suscribe en el bróker de comunicación para recibir en tiempo real las predicciones. En algunos casos, el proveedor de servicios realizará una consulta de las predicciones a través de la explotación de las API o a través del lenguaje de alto nivel del sistema de almacenamiento empleado como repositorio de predicciones.

Finalmente, cada subproceso registrará el éxito o no de su tarea en *logs* con el fin de que se pueda dar seguimiento a sus operaciones desde el dominio de operación de la arquitectura.

3.6.4 Distribución de modelos predictivos como servicios

El proceso de distribución de modelos predictivos como servicios toma los modelos almacenados por el proceso de extracción de información en el repositorio de modelos y los transforma en servicios. Este proceso está compuesto por el repositorio de modelos, el subproceso de desarrollo del servicio, el registro del servicio, el repositorio del servicio y el consumidor del servicio. La Figura 3.7 muestra el diagrama de flujo del proceso.

El subproceso de desarrollo del servicio realiza una consulta al repositorio de modelos de predicción para determinar la versión más reciente del modelo de predicción. Una vez localizado el modelo, este se introduce en una aplicación web que habilita una API REST para recibir peticiones del tipo POST. La aplicación internamente recibe las peticiones con los datos como archivo adjunto, evalúa empleando el modelo predictivo (basado en *machine learning* o *deep learning*), y envía una respuesta en formato JSON indicando la predicción realizada. Posteriormente, la aplicación web es aislada en contenedores virtuales para su fácil despliegue. El contenedor virtual que contiene la aplicación web es registrado en el catálogo de servicios y almacenado en un repositorio de servicios para ser desplegada en la infraestructura del sistema cuando se la requiera.

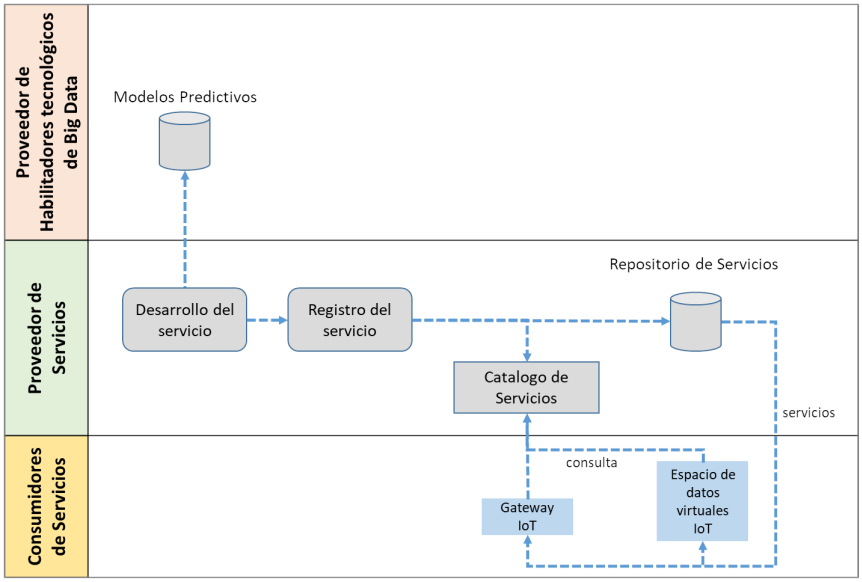


Figura 3.7: Diagrama de flujo del proceso de distribución de modelos como servicios

El subproceso de registro del servicio tiene la tarea de gestionar las aplicaciones web que contienen las modelos predictivos a través del uso de las funcionalidades provistas por el componente de gestión de servicios del dominio de aplicaciones. Este subproceso registra la aplicación web en el catálogo de servicios. El subproceso utiliza como parámetros de registro el nombre del servicio, versión del servicio y ubicación en el repositorio de servicios. De esta manera, los consumidores de servicios pueden localizar el servicio requerido fácilmente.

El despliegue del servicio es realizado a entornos tecnológicos que soportan la virtualización ligera de contenedores. Así, los consumidores de servicios deben habilitar este tipo de entornos para la ejecución del servicio. Los posibles consumidores de servicio para este proceso pueden ser *gateways* IoT y los espacios de datos virtuales que soporten la virtualización de contenedores; e incluso el proveedor de habilitadores tecnológicos de Big Data puede utilizar el servicio para realizar predicciones en tiempo real, como vimos en el proceso de la sección 3.6.3. El consumidor de servicios consulta el servicio requerido en el catálogo de servicios, y solicita el despliegue del servicio. El servicio se despliega y ejecuta en la plataforma de virtualización de contenedores del consumidor de servicios. Todos los subprocesos descritos registrarán *logs* de la operación del proceso.

3.6.5 Visualización de analíticas y KPI

El proceso de visualización de analíticas y KPI tiene el objetivo de presentar los resultados de los procesos de extracción y predicción en tiempo real a los consumidores de servicios. El proceso está compuesto por los repositorios de resultados de análisis y predicción, el bróker de comunicaciones, la aplicación y el usuario final. La Figura 3.8 muestra el diagrama de flujo del proceso.

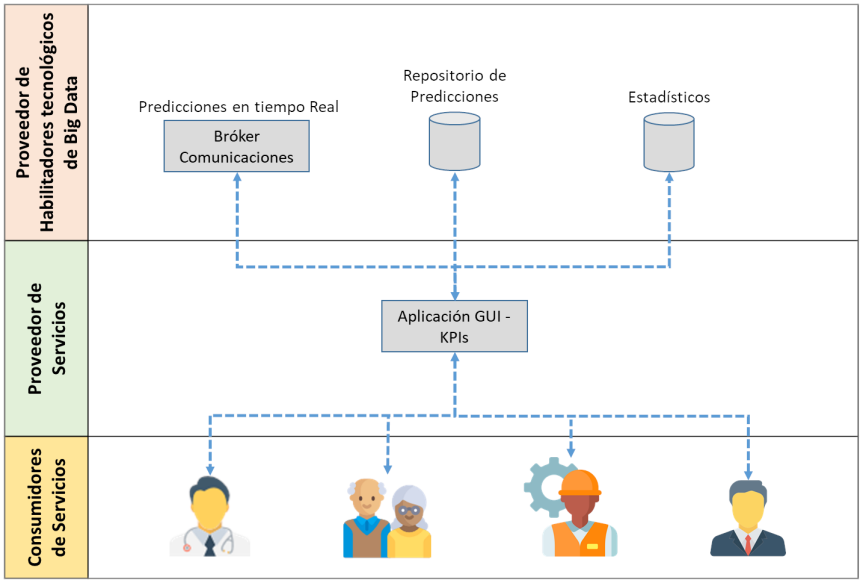


Figura 3.8: Diagrama de flujo del proceso de visualización de analíticas y KPIs

La aplicación GUI realiza consultas puntuales sobre los repositorios de resultados *batch*, tiempo real y de predicción. Los datos consultados son mostrados en forma de gráficos y valores resumidos para una mejor interpretación de los resultados. Además, la aplicación GUI utiliza el bróker de comunicación para recibir los resultados en tiempo real realizados por el proceso de predicción en tiempo real. La visualización de la información en forma de KPI tiene como ventaja la rápida interpretación de los resultados y con ello una mejor toma de decisiones. El acceso de la aplicación a los datos será registrado en forma de *logs* para registrar las operaciones realizadas por el proceso.

Los usuarios finales accederán a la aplicación web GUI usando el método de autenticación usuario y contraseña. Las actividades realizadas dentro de la interfaz GUI por los usuarios finales serán registradas en forma de *logs*.

3.7 Vista de implementación

La vista de implementación describe técnicamente la distribución de los componentes en un sistema IoT para el soporte de Big Data. La implementación de la arquitectura se la puede realizar usando tecnologías como *cloud computing* y *fog computing*. Estas tecnologías se las puede combinar entre sí formando patrones tecnológicos. La arquitectura se puede implementar siguiendo el patrón tecnológico basado en *cloud computing*, el patrón basado en la combinación de *cloud-fog computing*; y en algunos casos siguiendo el patrón basado en *cloud-cloud computing*. A continuación se describen estos patrones de implementación y la correspondiente distribución de las funcionalidades y procesos de la arquitectura.

3.7.1 Patron basado en cloud computing

El patrón basado en *cloud computing* utiliza los recursos computacionales provistos por *cloud computing* para la implementación de los componentes funcionales y procesos de la arquitectura. La implementación de este patrón puede darse para los casos en que el proveedor de datos, el proveedor de habilitadores tecnológicos de Big Data y el proveedor de servicios son implementados en una misma infraestructura *cloud computing*. La Figura 3.9 muestra el patrón basado en *cloud computing*.

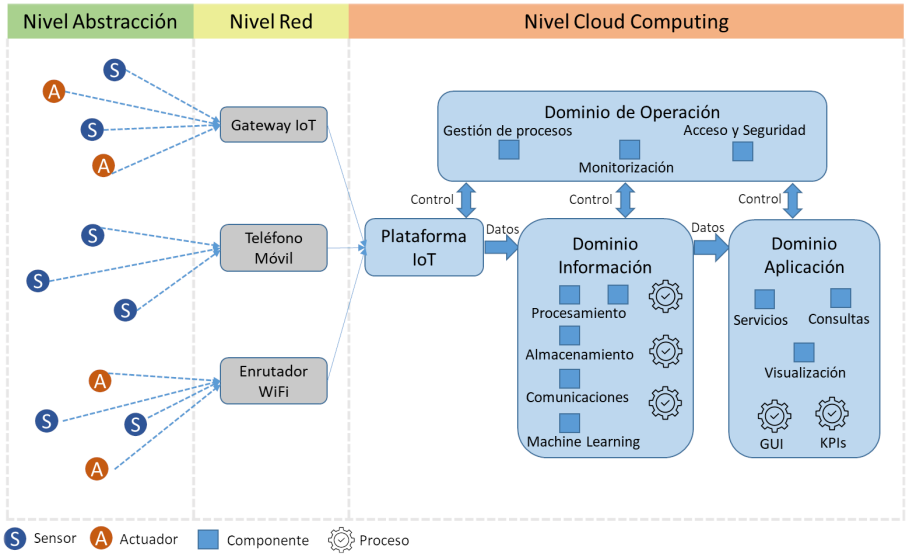


Figura 3.9: Implementación de la arquitectura basado en *cloud computing*

Este patrón responde a las necesidades de soporte de Big Data de sistemas IoT implementados bajo el concepto de *Cloud of Things*. En este caso, la arquitectura típica de IoT presenta tres niveles: el nivel de percepción, el nivel de red y el nivel de aplicación. El nivel de percepción está compuesto por los dispositivos IoT (sensores y actuadores) que se encargan de percibir el mundo físico. El nivel de red puede ser cualquier dispositivo que soporte los protocolos y tecnologías de comunicación inalámbricas (Bluetooth, ZigBee, 6LowPAN, WiFi, etc.) para que los dispositivos IoT envíen los datos al *cloud*. Estos dispositivos con capacidades de red pueden ser *gateways* IoT, teléfonos móviles, enrutadores inalámbricos, etc. Los datos recolectados por los dispositivos IoT son enviados a los dispositivos de red empleando protocolos de bajo consumo de energía como MQTT y CoAP. En el *cloud*, una plataforma IoT o *middleware* IoT se encarga de recibe los datos y controlar los dispositivos IoT.

La arquitectura de Big Data es implementada en la misma infraestructura *cloud* donde se encuentra la plataforma o *middleware* IoT. El *cloud computing* brinda las capacidades computacionales para implementar las funcionalidades de los diferentes dominios y los procesos para el soporte de Big Data. Además, *cloud computing* provee características intrínsecas de alta disponibilidad, escalabilidad, eficiencia y adaptabilidad que requiere la arquitectura. Dado que comparten la misma infraestructura tecnológica, algunos de los componentes funcionales del dominio de operación pueden ya estar implementados como parte de la plataforma IoT. De esta manera, la arquitectura aprovecha parte de los recursos y funcionalidades de la plataforma IoT destinados por ejemplo a la monitorización o al acceso y seguridad de la plataforma IoT.

3.7.2 Patrón basado en fog-cloud computing

A diferencia del enfoque *cloud computing*, el enfoque combinado emplea los recursos y capacidades computacionales provistos por *fog* y *cloud computing* para implementar las funcionalidades y los procesos de la arquitectura. Como resultado, algunas funciones y procesos son distribuidos tanto en la infraestructura *fog computing* como en el *cloud computing*. La Figura 3.10 muestra el patrón de implementación *fog-cloud computing* y la distribución de procesos y funcionalidades.

Los sistemas IoT que implementan este tipo de patrón presentan una arquitectura de tres niveles: el nivel dispositivos IoT, nivel *fog computing*, y el nivel *cloud computing*. El nivel de dispositivos IoT está compuesto por sensores capaces de percibir el entorno físico y por actuadores capaces de ejecutar comandos para realizar alguna acción en el entorno físico. Los

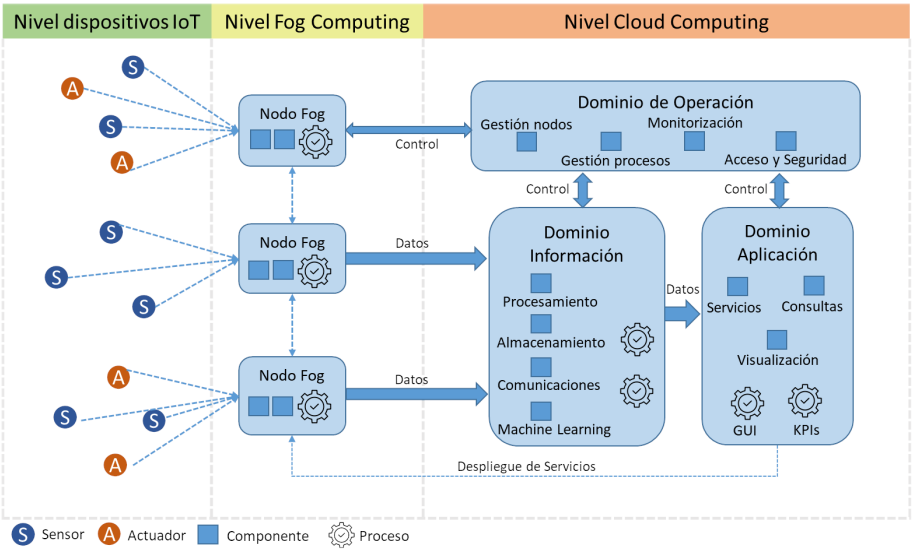


Figura 3.10: Implementación de la arquitectura basado en *fog-cloud computing*

dispositivos IoT están equipados con tecnologías de comunicaciones inalámbricas (por ejemplo, Bluetooth, ZigBee, 6LowPAN, WiFi, etc.) y soporta protocolos ligeros en consumo de energía (por ejemplo, MQTT y CoAP) para el envío de datos a Internet.

Por otro lado, el nivel *fog computing* está compuesto por nodos *fog* que presenta capacidades computacionales y de comunicación. Los nodos *fog* estarán equipados con las tecnologías de comunicaciones inalámbricas y protocolos para la comunicación bidireccional con los dispositivos IoT y con recursos computacionales para el almacenamiento y procesamiento de datos. A este grupo de nodos pertenecen los *gateways* IoT. En algunos casos, un nodo *fog* podrá solo tener capacidades de comunicación, por lo que requerirá de otro nodo *fog*que se encuentre en su mismo entorno para cubrir las tareas de almacenamiento y procesamiento de datos. Este último caso puede darse, por ejemplo, en entornos donde se utiliza enrutadores inalámbricos WiFi para conectarse con dispositivos IoT y utilizan otro nodo *fog* conectado en la misma LAN con capacidades computaciones para brindar los servicios de almacenamiento y procesamiento de datos.

El nivel *fog computing* implementa algunas funcionalidades del dominio de información y el proceso de predicción en tiempo real descrito en el dominio de procesos. Las funcionalidades de procesamiento de datos, almacenamiento de datos y gestión de comunicación son implementadas en los nodos *fog* que provean capacidades computacionales. Considerando que los nodos *fog*

tienen limitados recursos computacionales, el procesamiento en tiempo real deberá ser implementado utilizando librerías y plataformas ligeras para el procesamiento de flujos de datos, dejando el procesamiento tipo *batch* para que sea realizado en el nivel de *cloud computing*. Esta es una de las funcionalidades más críticas de la arquitectura usando este patrón de implementación, por esta razón el análisis de las ventajas y desventajas será analizado en el capítulo 5 con un caso de uso.

Por otro lado, las capacidades de comunicación estarán destinadas a gestionar las comunicaciones de los dispositivos IoT ya sea a través de un bróker de comunicación MQTT o un servidor CoAP. Además, los modelos de predicción serán desplegados como servicios desde el nivel de *cloud computing* para realizar predicciones en tiempo real y reducir los tiempos de predicción y detección de patrones. La detección de patrones y predicción en etapas tempranas de generación del Big Data reducirá la cantidad de datos que se envían al nivel *cloud computing* cubriendo de esta manera el requerimiento de eficiencia, y adaptabilidad. Además, el nodo *fog* podrá emitir una alerta notificando la detección de patrones críticos. Esto es útil para aplicaciones de IoT donde el tiempo de respuesta es crítico, por ejemplo aplicaciones de salud y bienestar. Finalmente, los datos recolectados por todos los nodos *fog* y las predicciones son enviados al nivel *cloud computing* para más procesamiento y análisis con el fin de extraer información útil.

El nivel *cloud computing* proporciona las capacidades computacionales para implementar las funcionalidades y procesos requeridos para la gestión del Big Data generado por IoT. Dado que algunas funcionalidades han sido distribuidas a nivel *fog computing*, el nivel *cloud computing* implementará las funcionalidades del dominio de información de procesamiento tipo *batch*, almacenamiento de datos, gestión de comunicación, y *machine learning* y minería de datos. Por otro lado, el proceso de recolección de datos en el que intervienen las funcionalidades del dominio de información será ejecutado utilizando la opción de temporalización del subproceso de extracción de datos. De esta manera, el subproceso de extracción de datos de los nodos *fog* extrae los datos pre-procesados a nivel *fog computing* en horarios de baja congestión de red. Además, el subproceso de extracción de datos podrá extraer datos puntuales de los nodos *fog* a través del uso de los lenguajes de consulta provistos por la funcionalidad de almacenamiento de los nodos *fog*. Siendo estrictamente necesario la extracción de todos los datos en los casos en que se requiera un análisis más preciso. De esta manera, el volumen de datos enviados al nivel *cloud computing* se reduce considerablemente.

Por otro lado, todas las funcionalidades del dominio de aplicación son implementadas, con especial énfasis en las funcionalidades requeridas para ejecutar el proceso de distribución de modelos predictivos como servicios.

De la misma forma, todas las funcionalidades del dominio de operación son implementadas, considerando la funcionalidad de gestión de nodos *fog* como la principal funcionalidad de este dominio a implementarse para habilitar la gestión remota de los nodos *fog* y controlar el buen funcionamiento y aprovisionamiento de recursos de los nodos *fog*.

3.7.3 Patrón basado en cloud-cloud Computing

El patrón basado en *cloud-cloud computing* puede darse en casos en que el proveedor de datos está implementado en una infraestructura de *cloud computing* distinta a la infraestructura de *cloud computing* (privada o pública) del proveedor de habilitadores tecnológicos de Big Data y del proveedor de servicios. El patrón puede darse con fuentes de IoT como plataformas IoT, datos abiertos y los espacios de datos virtuales de IoT. La Figura 3.11 muestra el patrón basado en *cloud-cloud computing*. A continuación analizamos los casos de implementación siguiendo este patrón dependiendo de la fuente de datos de IoT.

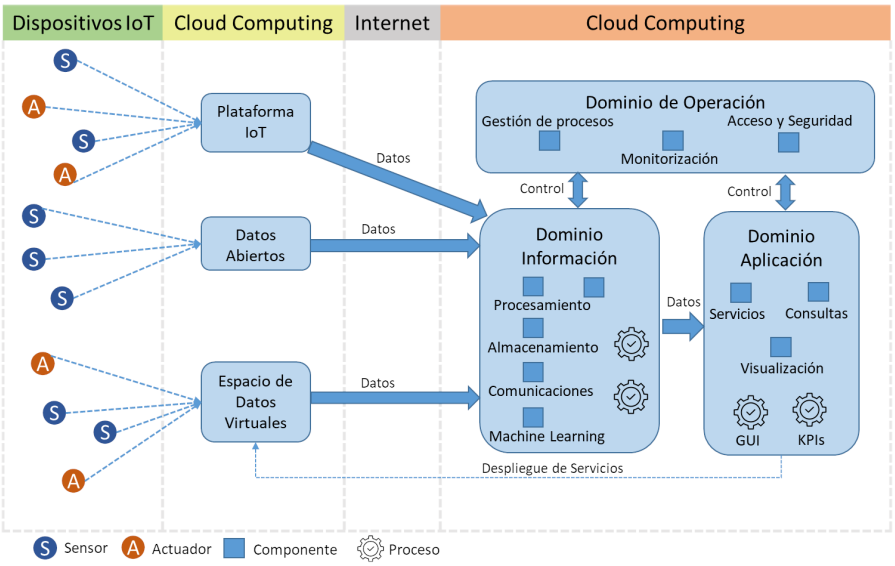


Figura 3.11: Implementación de la arquitectura basado en *Cloud-Cloud Computing*

El primer caso a analizar es el relacionado con la implementación de la arquitectura en una infraestructura de *cloud computing* distinta al de la plataforma IoT. Si bien este es un caso atípico es importante analizar el posible funcionamiento de la arquitectura en caso de llegar a darse. En este caso, la arquitectura IoT de la fuente de datos sigue la arquitectura

de referencia de *Cloud of Things* constituida por el nivel de abstracción, nivel de red y nivel de *cloud computing*. El nivel de abstracción dedicada a percibir el entorno físico a través de sensores y ejecutar comandos para realizar alguna acción a través de actuadores. El nivel de red encargado de transportar los datos de sensores al nivel *cloud computing*. Y el nivel *cloud computing* encargado de concentrar estos datos generados en el nivel de abstracción y presentarlos en la aplicación IoT. Por otro lado, la arquitectura de Big Data para el manejo de IoT es implementada en una infraestructura distinta de *cloud computing*, ya sea privada o pública. Es decir, todas las funcionalidades del dominio de información, dominio de operación y dominio de aplicación son implementadas en esta nueva infraestructura de *cloud computing*. El intercambio de datos entre ellas es realizado a través de Internet.

El segundo caso a analizar es el relacionado con datos abiertos como fuente de datos de IoT. Los datos abiertos son de libre acceso y están disponibles a través de plataformas que exponen sus API en Internet. Los datos abiertos de IoT son datos que tienen como fuente de origen dispositivos IoT y a los que cualquier persona puede acceder. La gran mayoría de datos abiertos de IoT viene de entidades gubernamentales que exponen sus datos para que sean reutilizados por la comunidad. Ejemplos de datos abiertos pueden ser los expuestos por ciudades inteligentes y empresas estatales para la monitorización de medio ambiente. El acceso a estos datos es controlado por una plataforma web para el almacenamiento y distribución de los datos, como por ejemplo la plataforma de gestión de datos CKAN. En este caso, la arquitectura propuesta se implementa en una infraestructura *cloud computing* distinta al de la fuente de datos. La implementación de las funcionalidades y procesos dependerá de los requerimientos de la aplicación y de los datos que es capaz de compartir CKAN. Por ejemplo, la mayoría de datos abiertos no son en tiempo real, por lo que no será necesario implementar el proceso de detección en tiempo real, ni proveer funcionalidades para el procesamiento en tiempo real. Este caso será más analizado como parte de un caso de uso en el capítulo 4.

El último caso a analizar es el relacionado con los espacios de datos virtuales de IoT. Este caso se presenta principalmente en entornos industriales donde existen varias partes interesadas y requieren compartir sus datos en entornos seguros. La principal motivación de los espacios de datos virtuales de IoT es mantener el control de los datos en todo momento, es decir conocer cómo se explotan los datos y para qué son usados estos datos. Esta motivación está ligada al requerimiento de soberanía de datos, donde el propietario de los datos mantiene bajo control el uso que se dé a sus datos de IoT. Los espacios de datos virtuales de IoT definen conectores con las funcionalidades de necesarias para compartir datos y mantener la privacidad y

seguridad sobre los mismos. Por ejemplo, en la arquitectura IDS las partes interesadas que requieran formar parte del entorno seguro IDS deben implementar el conector IDS. La arquitectura propuesta puede ser implementada por cualquiera de las partes interesadas para extraer el conocimiento de los datos compartidos en los entornos seguros. La implementación de las funcionalidades y procesos, e incluso el conector del entorno seguro, es realizada en una infraestructura de *cloud computing* distinta al resto de las partes interesadas. Además, los modelos predictivos pueden ser desplegados como un servicio para ejecutar el proceso de predicción en tiempo real en la infraestructura de *cloud computing* del propietario de datos. Esto es útil para los casos donde el propietario de los datos requiere controlar el proceso de extracción de información con el fin de controlar el uso que se dé a sus datos. El despliegue de los modelos predictivos como un servicio seguirá el proceso detallado en la sección 3.6.4 sobre la distribución de modelos predictivos. Este despliegue puede ser realizado a conectores del espacio de datos virtuales, como por ejemplo al conector IDS.

3.8 Conclusiones

En este capítulo se ha presentado una arquitectura integral para la gestión del Big Data generado por los entornos IoT. La arquitectura ha sido diseñada considerando el volumen de los datos, la velocidad de los datos y la variedad de fuentes de datos de IoT. Además, la arquitectura consideró los requerimientos planteados en las recomendaciones técnicas propuestas por los organismos internacionales de estandarización ITU-T y el NIST para la interoperabilidad de *frameworks* de Big Data y la implementación de ecosistemas IoT. Además, la arquitectura considera como un requerimiento fundamental la soberanía de datos para garantizar la privacidad de los datos durante el tiempo de vida del Big Data sin afectar a los intereses de los propietarios de los datos. El uso de estas consideraciones generales permitió el planteamiento de requerimientos funcionales y no funcionales para el soporte de Big Data en aplicaciones IoT. Como resultado, la arquitectura integral ha sido diseñada de manera genérica para que pueda adaptarse a los requerimientos particulares de cada aplicación IoT.

La arquitectura ha sido descrita utilizando la normativa ISO/IEC/IEEE 42010:2011 para la descripción de arquitecturas de *software*. De esta manera, la arquitectura descrita en este capítulo podrá ser entendida y aplicada por los arquitectos de aplicaciones IoT y por el público en general. La descripción de la arquitectura ha sido realizada desde el punto de vista conceptual, funcional, de procesos y de implementación. La descripción de la

arquitectura desde estos puntos de vista permite abordar de mejor manera los requerimientos funcionales y no funcionales.

Desde el punto de vista conceptual, la arquitectura muestra los roles que intervienen en durante el ciclo de vida del Big Data y sus interacciones. La descripción de estos roles permitirá identificar fácilmente las partes interesadas en el manejo de Big Data generado por IoT y compartir su visión empresarial. Como resultado, las partes interesadas pueden entender el funcionamiento de la arquitectura y adoptarla para la resolución de problemas con el manejo de Big Data en sus aplicaciones IoT.

Desde el punto de vista funcional, la arquitectura presenta sus componentes y funcionalidades para la gestión del ciclo de vida del Big Data y los agrupa en los dominios funcionales de IoT, de información, de operación y de aplicación. Las funcionalidades están diseñadas para gestionar el volumen, velocidad y variedad de datos generados por los dispositivos IoT con el fin de extraer valor de los datos.

Desde el punto de vista de procesos, la arquitectura introduce la novedad del despliegue de los modelos de *machine learning* y *deep learning* para la predicción y detección de patrones lo más cercano posible a las fuentes de datos IoT. Esta especial característica permitirá reducir los tiempos de generación de análisis. De esta forma, la arquitectura se alinea con las nuevas tendencias en la técnica del procesamiento de borde o *edge analytics*.

Desde el punto de vista de implementación, la arquitectura define los patrones de implementación utilizando tecnologías computacionales como *fog computing* y *cloud computing*. La especificación de estos patrones facilitará adopción de la arquitectura en entornos de aplicación de IoT basados en el estándar 1934 IEEE para *fog computing*. De esta manera, la arquitectura promueve el uso de *fog computing* para reducir el volumen de datos enviados al *cloud* con el fin de reducir problemas de congestión, retados y mejorar la calidad de experiencia de los usuarios finales.

Capítulo 4

Sistema IoT para el control y monitorización de la Apnea

4.1 Introducción

Las enfermedades respiratorias crónicas son consideradas como enfermedades críticas en las personas y en especial en adultos mayores. Estas enfermedades han sido causantes de cerca de 3 millones de muertes en el 2015 [141]. Las enfermedades respiratorias crónicas en su mayoría son producto de una larga exposición al humo de tabaco y contaminación ambiental tanto dentro como fuera de casa [141]. Uno de los trastornos más comunes y peligrosos es la apnea obstructiva del sueño. La apnea del sueño es una enfermedad respiratoria crónica que afecta a gran parte de la población mundial. Esta es una enfermedad que puede ocasionar muertes, pobre calidad de vida y altos costos sanitarios [142]. Por ello, la apnea del sueño requiere de innovadoras estrategias dirigidas a reducir sus problemas relacionados.

El uso de un sistema de IoT tiene el potencial de monitorizar el avance de la enfermedad. Varios dispositivos IoT entre los que se encuentran sensores de ambiente, temperatura o movimiento están destinados a recolectar información relacionada con el ambiente del sueño. Los enfoques actuales de

esta clase de sistemas utilizan plataformas IoT para concentrar los datos generados por los dispositivos IoT y controlar estos dispositivos. De esta manera, se habilita un entorno inteligente capaz de controlar los factores que afectan a la apnea del sueño.

Sin embargo, los sistemas IoT presentan limitaciones para manejar el gran volumen de datos generados durante el proceso de monitorización y con poca calidad de resultados de análisis de datos para la toma de decisiones. En este caso de uso se verifica la capacidad de la arquitectura de Big Data propuesta en el capítulo 3 para integrarse con un sistema de apnea basado en IoT con el fin de generar mejores prestaciones y servicios para controlar y monitorizar la apnea del sueño. Además, el caso de uso integra datos abiertos de una ciudad inteligente como fuente adicional de IoT para facilitar un enriquecimiento de los datos recolectados por el sistema IoT de la apnea.

En este capítulo se inicia con la motivación y los trabajos relacionados. A continuación, se describe la relación de la arquitectura del sistema con la arquitectura global. Posteriormente, se describe la instanciación de la arquitectura de Big Data para ser integrada con el sistema de IoT. Posteriormente, se describe el diseño de servicios y desarrollo de varios KPI basados en las analíticas generadas tras la explotación de los datos masivos del sistema IoT y otras fuentes de datos. Finalmente, se realiza la evaluación de los resultados de las analíticas con varios conjuntos de datos y la evaluación del rendimiento de la instancia de la arquitectura durante la explotación de los datos.

4.2 Motivación y trabajos relacionados

Actualmente, la Polisomnografía (PSG) es el método convencional para evaluar la apnea. Esta prueba es realizada en un hospital o clínica en donde el paciente debe permanecer toda la noche con sensores atados a su cuerpo [143]. Un sistema de monitoreo basado en PSG resulta impráctico por su alto costo y por ser intrusivo [143]. Por estas razones, varios sistemas han sido propuestos para monitorizar la apnea del sueño remotamente basado en IoT. Los sistemas pueden ser agrupados por las funciones que cumplen en: sistemas para monitorizar la apnea del sueño y sistemas para la detección de la apnea.

Los sistemas de monitorización tienen el objetivo de evaluar la evolución de la apnea a través de sensores desplegados en los pacientes. Los sistemas están basados principalmente en el uso de sensores para la generación de un Electrocardiograma (ECG) con tecnologías inalámbricas como Bluetooth

[144] [145]. Este tipo de sistemas emplean un teléfono inteligente como *gateway* IoT para habilitar la conexión de los sensores ECG a servidores en Internet. Los servidores son los encargados de pre-procesar los datos, detectar los eventos de apnea y emitir alertas en caso de un riesgo crítico (paciente con ataques cardíacos durante sus etapas de sueño). La detección de los eventos de apnea son realizados a través de reglas [145] o empleando algoritmos de *machine learning* [144].

Para enriquecer con contexto, algunos sistemas están basados en sensores multimodales capaces de monitorizar varios parámetros fisiológicos. La recolección de datos sobre la respiración, ritmo cardíaco, postura del cuerpo cuando la persona duerme, y sonidos de respiración han sido empleados por los sistemas para agregar precisión en la detección y proporcionar más información durante la monitorización del paciente [146] [147] [148]. Los nuevos sensores añadidos a los sistemas de monitorización de la apnea no solo utilizan Bluetooth sino también tecnologías inalámbricas como ZigBee y Z-wave formando una red inalámbrica de sensores. Dado que las tecnologías ZigBee y Z-wave no son soportadas de manera transparente por un teléfono móvil, los sistemas utilizaron un computador portátil como dispositivo *gateway* IoT capaz de realizar una traslación de protocolos y facilitar la conexión a Internet de los sensores inalámbricos [147] [149].

Una mejora a este tipo de sistemas apareció con la agregación de sensores con capacidades de monitorización de parámetros ambientales [150]. Los parámetros ambientales (temperatura, humedad, contaminación interna) están directamente relacionados con la apnea del sueño. El sistema llamado "*Sleep Mon*" recolecta los datos de los sensores inalámbricos, teléfonos móviles y relojes inteligentes a través de WiFi. Los datos recolectados son enviados al servidor de "*Sleep Mon*" para realizar un análisis y procesamiento. Una de las características que integran este nuevo sistema es el envío de recomendaciones a los pacientes sobre cómo pueden mejorar su salud (por ejemplo, recordatorios para realizar ejercicios).

Los sistemas actuales de la apnea del sueño basados en IoT no contemplan el manejo del gran volumen de datos en despliegues a gran escala. Esto ocasiona problemas para afrontar los retos relacionados con el manejo de Big Data generado por IoT y una pérdida de oportunidades para extraer información útil que ayude a gestionar mejor el tratamiento de la apnea del sueño. Además, los sistemas actuales de la apnea del sueño basados en IoT no sacan ventaja de una posible integración con otras fuentes de datos como puede ser los datos abiertos de una ciudad inteligente para proveer mejores prestaciones en sus sistemas. Estas necesidades actuales son la principal motivación para validar la arquitectura de Big Data propuesta en el capítulo 3 en este caso de uso. Como solución a las limitaciones actuales

se presenta la integración de la arquitectura de Big Data para la gestión del gran volumen de datos generados por el sistema IoT de la apnea del sueño. A diferencia de los trabajos relacionados, este trabajo no solo incluye la arquitectura de Big Data para IoT y su integración con el sistemas IoT sino también considera como fuente de IoT los datos abiertos compartidos por la ciudad inteligente.

4.3 Relación de la arquitectura del sistema con la arquitectura global

El principal objetivo de este capítulo es demostrar la utilidad de la arquitectura general presentada en el capítulo 3 por medio de la definición de una instancia de la arquitectura para la integración con un sistema IoT en producción con necesidades de soporte de Big Data.

Las necesidades de soporte de Big Data vienen definidas en forma de requerimientos. En la Tabla 4.1 se estructuran los requerimientos funcionales de este caso de uso y su correspondiente relación con los requerimientos generales definidos en la sección 3.2 del capítulo 3. Los requerimientos de gestión de los sistemas IoT (R9), Soporte de múltiples tecnologías computacionales (R10) y la soberanía de datos (R12) no son considerados en este caso de uso debido a que no son parte de las necesidades del sistema propuesto.

Tabla 4.1: Requerimientos del sistema Apnea para el soporte de Big Data

Requerimiento	**Relación Requerimientos Generales**
El sistema debe ser capaz de soportar las conexiones con la plataforma IoT del sistema basada en FIWARE y del sistema CKAN de la ciudad inteligente a través de sus API.	**R1.** Soporte de conexión con las múltiples fuentes de IoT.
El sistema debe ser capaz de gestionar y normalizar el formato JSON utilizado por la plataforma IoT y los formatos CSV y GEOJSON del sistema CKAN de datos abiertos de la ciudad inteligente.	**R2.** Soporte para la conversión de formatos de datos de IoT.
El sistema debe ser capaz de procesar en tiempo real los datos de la plataforma IoT.	**R3.** Procesamiento de datos de IoT en tiempo real.
El sistema debe ser capaz de soportar el preprocesamiento masivo de datos para generar conjuntos de datos que permitan generar análisis descriptivos.	**R4.** Procesamiento de datos de IoT tipo *batch*.

El sistema debe ser capaz de soportar el almacenamiento de datos enviados en tiempo real desde la plataforma IoT y los datos de CKAN.	**R5.** Soporte de almacenamiento.
El sistema debe ser capaz de realizar un análisis descriptivo y predictivo de los datos con el fin de proveer servicios de visualización y servicios de predicción que ayuden a tomar decisiones.	**R6.** Análisis de datos
El sistema debe ser capaz de distribuir los servicios a los usuarios finales.	**R7.** Soporte en la generación y distribución de servicios.
El sistema debe ser capaz de resumir los resultados de las analíticas descriptivas en forma de gráficos para mejorar su interpretación.	**R8.** Visualización de resultados de los análisis de datos.
El sistema debe ser capaz de proteger los datos enviados por la plataforma IoT y los datos abiertos.	**R11.** Seguridad.

Desde el punto de vista conceptual de la arquitectura, la instancia de la arquitectura de Big Data cumple el rol de proveedor de tecnologías habilitadoras de Big Data y de proveedor de servicios. Por otro lado, los proveedores de datos IoT son la plataforma IoT del actual sistema de apnea y la plataforma CKAN de la ciudad inteligente. El rol de consumidor de servicios es protagonizado por los especialistas médicos que siguen el tratamiento de la apnea en sus pacientes y por los pacientes diagnosticados con apnea.

La instancia de la arquitectura se implementa siguiendo las recomendaciones provistas por la descripción de la arquitectura desde el punto de vista de implementación. El patrón de implementación basado en *cloud computing* definido en la sección 3.8.1 se adecua para este caso en particular debido a que la plataforma IoT se encuentra en producción en un centro de datos (infraestructura *cloud computing*). De esta manera, la instancia de la arquitectura de Big Data toma ventaja de la infraestructura computacional provista por el centro de datos donde está instalada la plataforma IoT.

4.4 Sistema de monitorización de la apnea del sueño con soporte de Big Data

El sistema de monitorización de la Apnea del sueño está orientado a gestionar la salud de pacientes adultos mayores. De esta manera, el sistema se alinea con los objetivos que persigue el proyecto ACTIVAGE para soportar un envejecimiento activo y saludable en esta población vulnerable. En esta sección se describe la instanciación de la arquitectura de Big Data para solucionar los problemas de Big Data en el sistema de monitorización de la apnea de sueño y generar servicios para guiar el tratamiento y controlar la apnea del sueño. Además, se proporcionan detalles sobre la implementación de los procesos para recolección de datos de las fuentes de IoT, extracción de información, predicción en tiempo real y visualización de datos.

4.4.1 Instanciación de la Arquitectura de Big Data

La arquitectura del sistema de monitorización de la apnea del sueño con soporte de Big Data es descrita siguiendo el punto de vista funcional de la arquitectura propuesta en el capítulo 3. La Figura 4.1 muestra las plataformas y herramientas utilizadas en esta instancia de la arquitectura conforme el patrón arquitectural de la vista funcional de la arquitectura (sección 3.5).

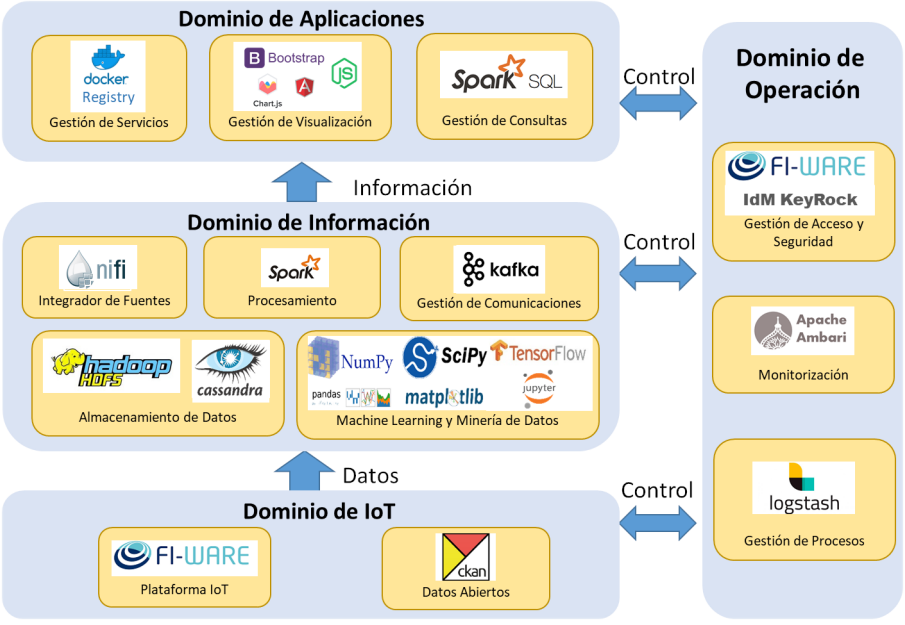


Figura 4.1: Instancia arquitectura de Big Data para IoT

Dominio IoT

El dominio IoT en este caso de uso está compuesto por el sistema IoT de la apnea basado en la plataforma IoT FIWARE que se encuentra en producción y los datos abiertos de Valencia Ciudad Inteligente (VLCi).

La plataforma IoT FIWARE proporciona un conjunto de funcionalidades generales a través de sus GE. *Orion Context Broker* es el GE principal en la plataforma puesto que concentra toda la información contextual de los sensores. Para ello, *Orion Context Broker* proporciona las interfaces NGSI 9 y 10 basadas en REST API para el registro y actualización de información de contexto de los sensores. *Orion Context Broker* basa su funcionamiento en un mecanismo de comunicación publicación/suscripción para concentrar los datos de IoT y distribuirlos a los demás GE. *Orion Context Broker* está en capacidad de almacenar los últimos datos de los sensores por lo que requiere de un mecanismo para almacenar estos datos de manera persistente.

Empleando la plataforma IoT FIWARE, el sistema IoT para la apnea del sueño es capaz de monitorizar las actividades físicas, el ambiente del sueño y el estado del sueño de los pacientes. Las actividades físicas son monitorizadas empleando una muñequera inteligente capaz de detectar la cantidad de pasos y el ritmo cardíaco del paciente durante el día. Por otro lado, el ambiente del sueño monitoriza la temperatura y humedad de la habitación del paciente durante todo el día a través de sensores distribuidos en la casa del paciente. Mientras tanto, el estado del sueño monitoriza las fases del sueño (sueño ligero y profundo) empleando una muñequera inteligente; así como, la severidad de la apnea (eventos de apnea durante la etapa del sueño) empleando un sensor de sonido. Estos parámetros de monitorización están configurados como entidades dentro de la plataforma IoT FIWARE. La Figura 4.2 muestra como ejemplo el modelo de datos de la entidad ambiente del sueño estructurado en formato JSON-LD (*keyValue*).

```
{
    "id":"urn:ngsi-ld:AmbienteSueño:001",
    "refPaciente": "urn:ngsi-ld:Paciente:2",
    "type": "AmbienteSueñoObservado",
    "temperatura": 29.3,
    "humedad": 65.4,
    "dateObservation": "2018-11-24T08:34:04.00Z"
}
```

Figura 4.2: Ejemplo JSON normalizado modelo de datos de la entidad ambiente del sueño

La entidad *AmbienteSueñoObservado* representa a una habitación con sus características individuales. La característica de identificación *id* corresponde a un identificador de recursos uniforme (URI, del inglés *Uniform Resource Identifier* basado en el modelo de información de FIWARE NGSI v2 (línea código 2). La característica *refPaciente* es la referencia al paciente al que corresponde el ambiente del sueño (línea código 3). La característica de temperatura y humedad corresponde a las medidas del sensor de temperatura ubicado en la habitación del paciente (línea código 5-6). La característica de *dateObservation* corresponde a la marca de tiempo en la que fue tomada la medida (línea código 7).

Por otro lado, la plataforma VLCi provee servicios relacionados con la monitorización de los contaminantes en el ambiente y del clima y que son considerados en este caso de uso para enriquecer los análisis y mejorar la toma de decisiones. La monitorización de contaminantes como Monóxido de Carbono (CO), Dióxido de Carbono (CO_2), Ozono (O_3), Dióxido de Azufre (SO_2), Monóxido de Nitrógeno (NO), Dióxido de Nitrógeno (NO_2), y partículas en suspensión (PM_{10}, PM_{25}) es realizada a través de 6 estaciones distribuidas en la ciudad de Valencia. Algunas de estas estaciones están en la capacidad de realizar la monitorización de las variables climáticas como la temperatura, humedad relativa, velocidad del viento, dirección del viento, presión atmosférica y precipitaciones.

La plataforma VLCi emplea la plataforma IoT FIWARE para la gestión de los datos de sensores distribuidos en la ciudad y la plataforma CKAN para la gestión y distribución de datos históricos. La plataforma CKAN provee una API REST para buscar el conjunto de datos en su catálogo. Los conjuntos de datos indicados para este caso se encuentran en formato CSV en el CKAN. Los datos históricos son descargados una única vez. Por otro lado, los datos en tiempo real de la plataforma IoT FIWARE proporcionaran de manera constante los datos actualizados de los contaminantes de manera horaria.

Dominio información

- **Integrador de Fuentes:** las funcionalidades del componente integrador de fuentes son implementadas empleando el sistema de automatización de flujos Apache NiFi. Esta herramienta es altamente escalable, configurable y segura, principalmente utilizada en arquitecturas de Big Data para el diseño de flujos de datos automáticos. Además, Apache NiFi proporciona un entorno gráfico a través de su interfaz web de usuario que facilita el diseño e implementación de flujos de datos. Esta herramienta es útil para soportar las tareas involucradas en el proceso de recolección de datos de las fuentes de IoT. Además,

Apache NiFi permite la conversión de los formatos de los datos (CSV y JSON) a formato Parquet. La utilización del formato Parquet se debe principalmente a que es considerado como un formato de alto rendimiento por lo que optimizará los recursos de almacenamiento [151].

- **Almacenamiento:** las funcionalidades de almacenamiento están divididas en dos tipos: almacenamiento histórico de datos y el almacenamiento de analíticas. El almacenamiento histórico emplea una instancia del sistema de ficheros distribuido HDFS para proveer alta disponibilidad, confiabilidad y escalabilidad. HDFS emplea 3 máquinas virtuales para formar un clúster Hadoop. Las máquinas virtuales serán configuradas como *Nodo Name*, *Nodo Data* 1 y *Nodo Data* 2. Los *Nodos Data* están destinados a almacenar bloques de datos de 64 *Megabyte* (MB), y el *Nodo Name* gestiona los *Nodos Data* y tiene conocimiento de la ubicación y replicación de los datos en los *Nodos Data* a través de los metadatos [152]. Un programa cliente es empleado para la lectura y escritura de los datos en HDFS. La Figura 4.3 muestra el clúster de HDFS de la arquitectura.

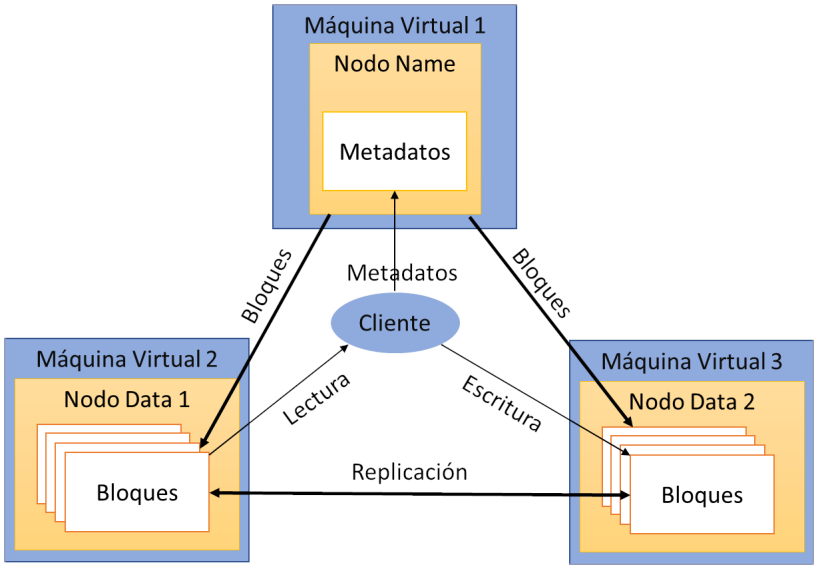


Figura 4.3: Arquitectura HDFS para el almacenamiento histórico de datos

El almacenamiento de analíticas será soportado por una instancia del sistema de base de datos NoSQL Apache Cassandra. Este sistema de base de datos es más robusto que otros durante los procesos de lectura y escritura de datos. Además, Apache Cassandra provee el lenguaje CQL para realizar consultas, similar a SQL [153].

- **Procesamiento:** las funcionalidades de procesamiento son proporcionadas por Apache Spark. El *framework* Apache Spark soporta la ejecución de aplicaciones que requieren un procesamiento *batch* y en tiempo real. Además, Apache Spark presenta mejores prestaciones que Hadoop en la ejecución de trabajos basados en *MapReduce* debido a la utilización de la memoria para su computación (RDD) [72]. Igualmente, Apache Spark provee el API Streaming para el soporte de aplicaciones en tiempo real, ampliamente utilizado en la actualidad. El gestor de recursos YARN provisto por el ecosistema Apache Hadoop será empleado para gestionar los recursos del clúster Hadoop y para la ejecución de las aplicaciones Apache Spark dentro del clúster Hadoop. Apache Spark está estructurado por un nodo *master* y varios nodos *worker*. El nodo *master* se encarga de la gestión de los nodos *worker*, mientras que los nodos *worker* ejecutan las tareas y emplean una memoria cache para almacenar resultados intermedios. En la Figura 4.4 se muestra la arquitectura de Spark dentro del clúster HDFS con YARN.

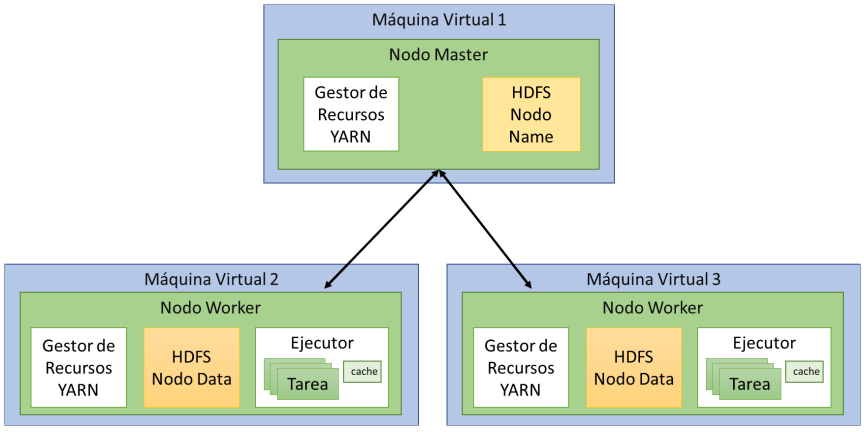


Figura 4.4: Arquitectura Spark dentro del clúster HDFS con YARN

- **Comunicación:** las funcionalidades de comunicación son implementadas usando Apache Kafka como plataforma de comunicación entre módulos debido a su escalabilidad, tolerancia a fallos y robustez; así como, por su amplio soporte con las plataformas y herramientas empleadas para integrar, procesar y almacenar los datos. Además, el componente emplea Apache Zookeper para mantener el estatus de los nodos Kafka así como de sus *topics* y particiones.

- **Machine Learning y Minería de Datos:** las funcionalidades de *machine learning* y minería de datos son implementadas empleando las librerías Scikit-Learn, TensorFlow y Keras para la explotación de los datos. En este caso, TensorFlow es empleado como *backend* de Keras para que ambas plataformas trabajen conjuntamente. La funcionalidad de desarrollo interactivo proporcionado por este componente funcional es implementado con la aplicación web Jupyter Notebook. La interfaz web provista por Jupyter Notebook permite la creación y compartición de documentos capaces de contener líneas de código interactivas que facilitan el modelado y explotación de los datos.

Dominio de aplicaciones

- **Gestión de servicios:** las funcionalidades de gestión de servicios son implementadas usando la plataforma Docker. Docker provee un entorno de virtualización de contenedores conjuntamente con un repositorio de contenedores para gestionar su utilización (Docker Registry). Empleando la virtualización de contenedores, los modelos desarrollados en el bloque de análisis de datos son aislados en programas aplicaciones. Estas aplicaciones implementan interfaces basadas en API REST para brindar sus servicios internamente. Los servicios de predicción son aplicados para realizar predicciones de los datos en tiempo real y brindar un servicio de recomendación para guiar en el tratamiento de la apnea siguiendo el proceso de predicción en tiempo real definido por la arquitectura.
- **Gestión de visualización:** el gestor de visualización provee una GUI web. Esta interfaz organiza los resultados de los análisis en forma de gráficas visuales para que el usuario pueda interpretar fácilmente los resultados. Para ello, el componente utiliza Nodejs (*backend*) para implementar el servidor web y las herramientas Bootstrap, Chartjs, Angular y Socket.io para el diseño la interfaz gráfica (*frontend*).
- **Gestión de consultas:** las funcionalidades del módulo gestor de consultas son implementadas empleando los lenguajes de alto nivel proporcionados por los sistemas de almacenamiento (Apache Cassandra) y sistemas de procesamiento (Apache Spark). El lenguaje CQL es empleado para generar consultas sobre los resultados de procesamiento, y SparkSQL (Structured Streaming) para generar consultas sobre los datos en tiempo real.

Dominio de operaciones

- **Gestión de acceso y seguridad:** las funcionalidades de la gestión de acceso y seguridad son provistas por el GE de FIWARE *Identity Manager* (IdM) KeyRock. En este caso se toma ventaja de la ya existente herramienta provista por la plataforma IoT y expandir su funcionamiento para controlar el acceso a las plataformas de la arquitectura de Big Data. IdM KeyRock utiliza el protocolo de autenticación OAuth2 el cual está basado en *tokens* de acceso para autorizar el acceso a un recurso protegido.
- **Monitorización:** Las funcionalidades de monitorización de las plataformas de Big Data es realizada a través de Apache Ambari. Esta herramienta permite una gestión y monitorización simple de entornos Hadoop a través de un *dashboard*. Además, las propias plataformas proveen sus interfaces para la gestión y monitorización de la ejecución de trabajos en el clúster de Apache Spark y Hadoop.
- **Gestión de procesos:** las funcionalidades de gestión de procesos se implementan usando la aplicación LogStash. Esta aplicación permite transportar los *logs* generados durante la ejecución de los procesos definidos en la arquitectura a un repositorio y analizarlos para encontrar rápidamente fallos.

4.4.2 Integración de la arquitectura de Big Data con las fuentes de datos

La integración de la arquitectura de Big Data con las fuentes de datos se la realiza utilizando el proceso de recolección de datos definido en la sección 3.6. En este caso se han creado dos instancias del proceso de recolección de datos definidos por la arquitectura para recolectar los datos del sistema IoT de la apnea y de los datos abiertos de la ciudad inteligente VLCi.

Las dos instancias del proceso fueron implementadas empleando la plataforma Apache NiFi provista por el dominio de información. Los procesos siguen un flujo de datos definidos en la plataforma Apache NiFi. Estos flujos de datos están compuestos por procesadores, los cuales proveen funcionalidades específicas destinadas a cumplir con los subprocesos de extracción de datos, conversión de formatos y la carga de los datos a las tecnologías y sistemas de destino, tal y como se describió en la subsección 3.6.1. Los flujos diseñados están compuestos por cuatro procesadores: ListenHTTP, PutParquet, PublishKafka y LogAttribute. La Figura 4.5 muestra el flujo de datos configurado en Apache NiFi para la integración con la plataforma FIWARE IoT del sistema de apnea.

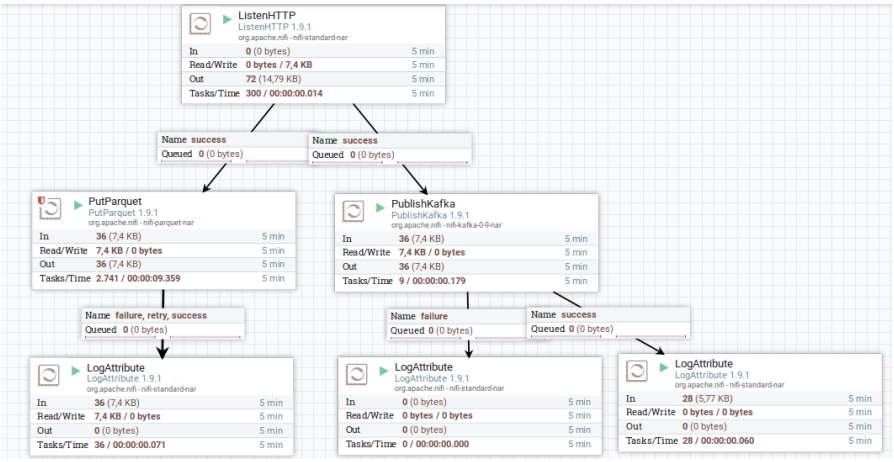


Figura 4.5: Flujo de datos configurado en NiFi para el proceso de recolección de datos

El primer flujo está compuesto por los procesadores ListenHTTP, PutParquet y LogAttribute. El procesador ListenHTTP es empleado para recibir los datos enviados desde la plataforma FIWARE (Sistema IoT y VLCi). Básicamente, el procesador implementa un servidor HTTP que se encuentra en la escucha el puerto 8080 para recibir peticiones POST que contienen las notificaciones de actualización de los datos de IoT. El flujo de datos entre el *Orion Context Broker* y este primer procesador ListenHTTP se muestra en la Figura 4.6.

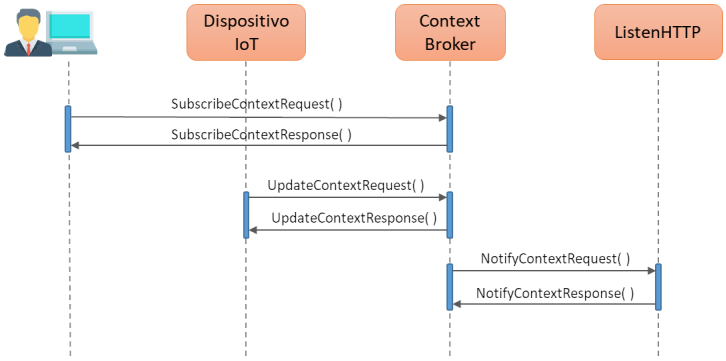


Figura 4.6: Flujo de datos entre *Orion Context Broker* y el procesador ListenHTTP

El flujo de datos aprovecha las interfaces NGSI-9 y NGSI-10 provistas por el GE *Orion Context Broker* de la plataforma IoT FIWARE. La interfaz NGSI-10 es utilizada para el intercambio de información acerca de las

entidades y sus atributos [24]. La interfaz NGSI-9 es usada para obtener información de disponibilidad sobre entidades y sus atributos [24]. Para este flujo, el administrador del sistema de IoT envía una petición de suscripción al GE *Orion Context Broker* para recibir notificaciones de contexto de todas la entidades (*SubscribeContextRequest()*). La petición puede ser enviada desde la línea de comandos del servidor donde reside Apache NiFi. Esta petición contiene el URL del servidor HTTP activado por el procesador ListenHTTP, las entidades de las que se requiere recibir notificaciones de actualización de datos, y los formatos de los atributos de cada entidad. El GE *Orion Context Broker* enviará las notificaciones de contexto a la dirección URL indicada en la petición de suscripción, que en este caso es la dirección IP del servidor donde se aloja Apache NiFi y el puerto 8080 del procesador ListenHTTP. El GE *Orion Context Broker* enviará notificaciones de contexto (*NotifyContextRequest()*) cada vez que las entidades se actualicen. Esto se realiza con cada petición de actualización de contexto (*UpdateContextRequest()*) enviada por los dispositivos IoT. Las notificaciones de contexto que recibe el procesador ListenHTTP se encuentran en formato JSON y requieren ser transformados a un formato más eficiente.

El procesador PutParquet es empleado para realizar la conversión de formatos de JSON al formato basado en columnas Parquet. Este formato es empleado por sus altas prestaciones en la compresión y acceso a los datos, brindando eficiencia en el almacenamiento de los datos en HDFS. Este procesador realiza dos tareas, convertir el formato de los datos y cargar los datos en HDFS. Para ello, el procesador requiere conocer a priori el esquema de los datos (esquema Avro), la ubicación de los archivos de configuración del clúster HDFS y la carpeta de destino en HDFS. El esquema Avro es un formato JSON que describe los tipos de datos y protocolos empleados en la definición del modelo de datos. La Figura 4.7 muestra como ejemplo el esquema Avro empleado para el estado del sueño monitorizado por la plataforma IoT. Por otro lado, los archivos de configuración (core-site.xml y hdfs-site.xml) proporcionan información acerca del Nodo Name del clúster HDFS y la configuración de los bloques de replicación. El procesador carga los datos en la carpeta de destino en HDFS utilizando la información de los archivos de configuración de HDFS.

El procesador de LogAttribute es utilizado para registrar el estado (*logs*) de cada transacción que ocurre durante el flujo de datos. Este procesador permite gestionar los errores que puedan presentarse durante la carga de los datos en HDFS.

El segundo flujo está compuesto por los procesadores ListenHTTP, PublishKafka y LogAttribute. Este flujo comparte el procesador ListenHTTP para recibir los datos enviados desde la plataforma FIWARE (Sistema IoT

```
{
 "name": "AmbienteSueño",
 "type": "record",
 "namespace": "apnea.avro",
 "fields": [
  {
   "name": "id",
   "type": "string"
  },
  {
   "name": "refPaciente",
   "type": "string"
  },
  {
   "name": "type",
   "type": "string"
  },
  {
   "name": "temperatura",
   "type": "float"
  },
  {
   "name": "humedad",
   "type": "float"
  },
  {
   "name": "dateObservation",
   "type": "int",
   "logicalType": "date"
  }
 ]
}
```

Figura 4.7: Ejemplo esquema Avro de la entidad ambiente del sueño

y VLCi). El procesador PublishKafka es utilizado para publicar los datos que llegan de las fuentes en tiempo real al sistema de comunicación de la arquitectura basado en Apache Kafka. Para ello, la URL del sistema de comunicación y el *topic* configurado para publicar los datos de la aplicación de la apnea son configurados en el procesador PublishKafka. Finalmente, el procesador LogAttribute registra el estado (*logs*) de cada transacción de este segundo flujo de datos.

4.4.3 Servicios para la monitorización y control de la apnea

La explotación del Big Data de IoT permite el desarrollo de servicios que faciliten la toma de decisiones y guíen el tratamiento para el empoderamiento de los adultos mayores. De esta manera, el uso de Big Data contribuyó a la generación de dos servicios para la apnea del sueño. Los servicios fueron desarrollados a través de una instanciación de los procesos definidos en la arquitectura de Big Data para IoT para la extracción de la información, predicción en tiempo real, y visualización de analíticas y KPI.

Servicio de visualización de los parámetros presentes en la apnea del sueño

El gran volumen de datos generados por el sistema de IoT para la monitorización de la apnea presenta varias oportunidades para extraer información útil que describa la situación actual de los adultos mayores y ayude a conocer la evolución de la apnea en este grupo vulnerable. El servicio de visualización tiene como objetivo representar gráficamente el resultado del análisis descriptivo de los datos para que los especialistas médicos puedan interpretar de mejor manera los resultados y sacar conclusiones que conduzcan a una mejor toma de decisiones. El servicio de visualización de los resultados está estructurado por la instancia de dos procesos definidos por la arquitectura: el proceso de extracción de información enfocado en un análisis descriptivo y el de visualización de analíticas y KPI. La Figura 4.8 muestra el diagrama de flujo de los datos durante los procesos de extracción de información y la visualización de analíticas.

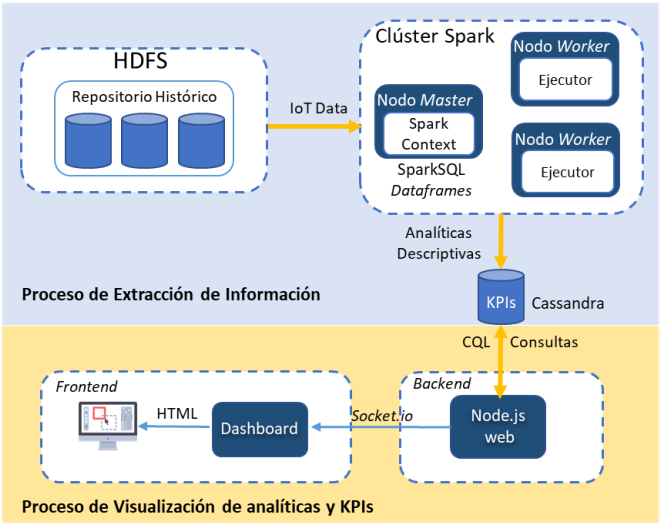


Figura 4.8: Flujo de datos del servicio de visualización

El proceso de extracción de información está compuesto por los subprocesos de carga de datos, pre-procesamiento y análisis descriptivo. El subproceso de carga de datos lee los datos del repositorio HDFS en formato Parquet utilizando el API SparkSQL de Apache Spark. Los datos son cargados de forma tabular en un *Dataframe*. El subproceso de pre-procesamiento realiza una limpieza de los datos. SparkSQL provee funcionalidades para realizar este tipo de manipulaciones con los datos contenidos en el *Dataframe*. Las principales funcionalidades utilizadas en este caso son la de verificación de

valores no nulos en el *Dataframe* y corrección de valores anómalos (valores atípicos propios de las mediciones con sensores).

El subproceso de análisis descriptivo se realiza ejecutando un análisis estadístico que ayuden a describir las características de los conjuntos de datos. El subproceso de análisis descriptivo de los datos se realiza empleando los datos recolectados por el sistema IoT sobre las actividades físicas de los pacientes (contador de pasos), estados del sueño (monitorización del sueño y severidad de la apnea), monitorización de constantes vitales (ritmo cardíaco), ambiente del sueño (temperatura y humedad). La estadística descriptiva es empleada para extraer los siguientes KPI:

- La cantidad de pasos en promedio, máxima, y mínima que realizan los pacientes.
- La cantidad de horas que duermen en promedio, máximo, y mínimo los pacientes.
- La distribución de la severidad de la apnea.
- El ritmo cardíaco en promedio, máximo y mínimo durante la etapa de sueño.
- La temperatura y humedad promedio de los ambientes del sueño de los pacientes.
- La contaminación interna en promedio, máximo y mínimo del ambiente del sueño de los pacientes.

Cada KPI es un programa aplicación que son ejecutados por Apache Spark. Los programas aplicación están basados en *MapReduce* y utilizando el API SparkSQL y sus funcionalidades para estructurar el conjunto de datos de forma tabular (*Dataframes*). Estos programas aplicación son trabajos que se ejecutan cada noche en la hora de menos carga operacional del clúster. Los resultados son almacenados en tablas en la instancia de base de datos Apache Cassandra.

El proceso de visualización de analíticas y KPI utiliza una GUI web para compilar los KPI y mostrar a los usuarios finales los resultados en forma de gráficas. La aplicación GUI es realizada a través de una aplicación basada en Nodejs. En este caso Nodejs cumple la función de *backend* de la aplicación. En este caso el servidor Nodejs se encarga de toda la lógica de la aplicación web. El programa servidor de la aplicación GUI establece las conexiones con la instancia de Apache Cassandra donde se almacenan los resultados de las analíticas. Además, el servicio de aplicación contiene las consultas a

las tablas de Apache Cassandra, consultas que fueron definidas utilizando el lenguaje CQL proporcionado por Apache Cassandra. Los datos recopilados por el servidor son enviados al *frontend* utilizando la librería de Javascript Socket.io.

Por otro lado, el *frontend* es implementado utilizando el *framework* Bootstrap de HTML y la librería Javascript Angular. Conjuntamente con estas herramientas, se ha empleado la librería Javascript Chart.js para el desarrollo de gráficos. Los datos recibidos en el *frontend* son transformados en gráficos para una mejor interpretación de los resultados. Además, los KPI son mostrados en forma de indicadores o valores numéricos para una fácil interpretación. Las gráficas seleccionadas para mostrar los resultados de análisis son el histograma, diagrama de barras, y diagrama de líneas. El histograma permite conocer la distribución de los conjuntos de datos de las actividades físicas de los pacientes, y estados del sueño. El diagrama de barras es empleado para visualizar la severidad de la apnea presente en los pacientes. Este tipo de gráfico permite comparar los niveles de severidad de la apnea presente en los pacientes analizados y determinar si los casos se están agravando o reduciendo. Y el diagrama de líneas presenta la cantidad de pasos realizados durante la semana.

Servicio de predicción del mejor lugar para realizar actividades de ocio

Las actividades al aire libre presentan una respuesta favorablemente al mejoramiento de la condición de la apnea del sueño, pero si estas actividades son realizadas en condiciones atmosféricas y de medio ambiente perjudiciales pueden causar un efecto inverso al deseado. Por ello, la necesidad de un servicio de predicción de los lugares menos contaminado y con adecuadas condiciones atmosféricas para que los adultos mayores que padecen apnea del sueño realicen las actividades de ocio. De esta manera, los adultos mayores toman el control del tratamiento basados en una autogestión del mismo.

El servicio de predicción del mejor lugar para realizar actividades de ocio está estructurado por tres procesos: el proceso de extracción de información enfocado en un análisis predictivo, el proceso de distribución de modelos predictivos como servicios y el proceso de predicción en tiempo real. La Figura 4.9 muestra el flujo de datos durante los procesos de extracción de información, el proceso de distribución de modelos predictivos como servicios y el proceso de predicción en tiempo real.

El proceso de extracción de información definido en la arquitectura es utilizado para cargar los datos históricos recolectados de la plataforma CKAN,

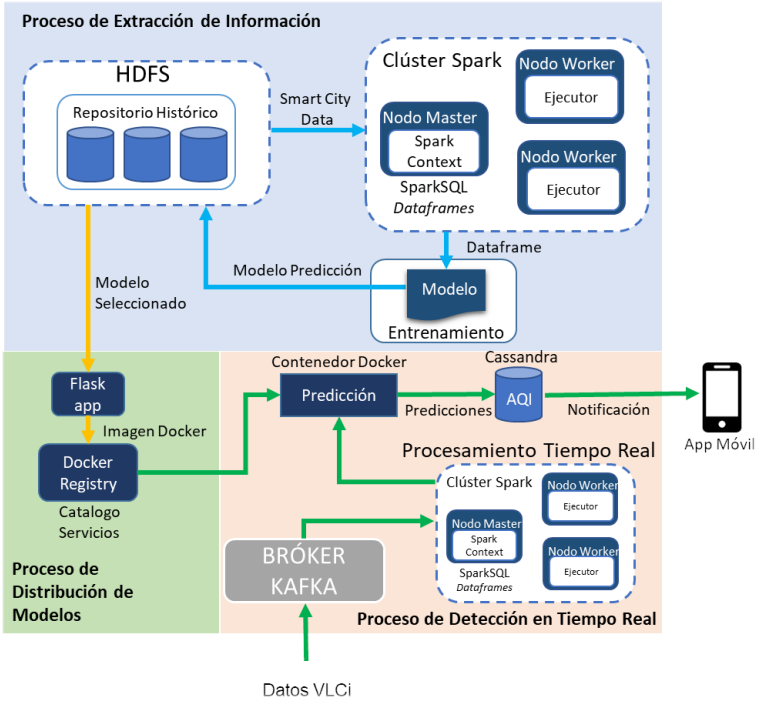


Figura 4.9: Flujo de datos del servicio de predicción

realizar tareas de pre-procesamiento (limpieza de datos, valores no nulos, y valores atípicos) y realizar un análisis descriptivo. Los subprocesos de carga de datos y pre-procesamiento son los mismos que se usaron para el servicio de visualización. Es decir, los conjuntos de datos en HDFS se cargan en *Dataframes* usando el API SparkSQL y con sus funcionalidades se realiza la limpieza de datos.

El subproceso de análisis predictivo utiliza algoritmos de *machine learning* para encontrar patrones y establecer modelos que permitan predecir los valores de los contaminantes. En este caso se utilizan los modelos de regresión lineal para predecir los valores de un conjunto de variables. Las variables principales de entrada son los valores de los contaminantes provistos por la ciudad inteligente. Dado que no todas las estaciones de monitorización de la ciudad de Valencia tienen la capacidad de monitorizar a todos los contaminantes, solo los SO_2, NO_2, NO y O_3 son seleccionados para predecir la calidad del aire en la ciudad.

Además a estas, las condiciones atmosféricas (temperatura, humedad, presión atmosférica, velocidad del viento, dirección del viento, precipitaciones) son determinantes en la predicción del comportamiento de los contaminantes en la atmósfera. Dado que el comportamiento de los contaminantes dependerá de varias variables, un análisis de correlación es realizado para determina los grados de dependencia entre variables. De esta manera se seleccionan las variables con mayor dependencia de cada contaminante para establecer el modelo de regresión lineal.

El análisis de correlación es realizado empleado el coeficiente de correlación de Pearson, el cual establece el grado de covariancia de las variables. La correlación de Pearson está definida por la ecuación 4.1. Donde σ_{xy} es la covarianza de (x, y), σ_x desviación estándar de la variable x, y σ_y desviación estándar de la variable y.

$$\rho_{x,y} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{4.1}$$

El valor del coeficiente ρ estará en el rango de -1 a 1 siendo los valores más próximos a 1 y -1 considerados como un grado de dependencia directa e inversa, correspondientemente. Mientras que si el valor es más cercano a 0 se considera una relación independiente entre las variables. Para cada contaminante se seleccionaron las variables con mayor dependencia entre sí para la generación de los modelos de predicción. La Figura 4.10 muestra la matriz de correlación de las variables consideradas para la predicción de los contaminantes.

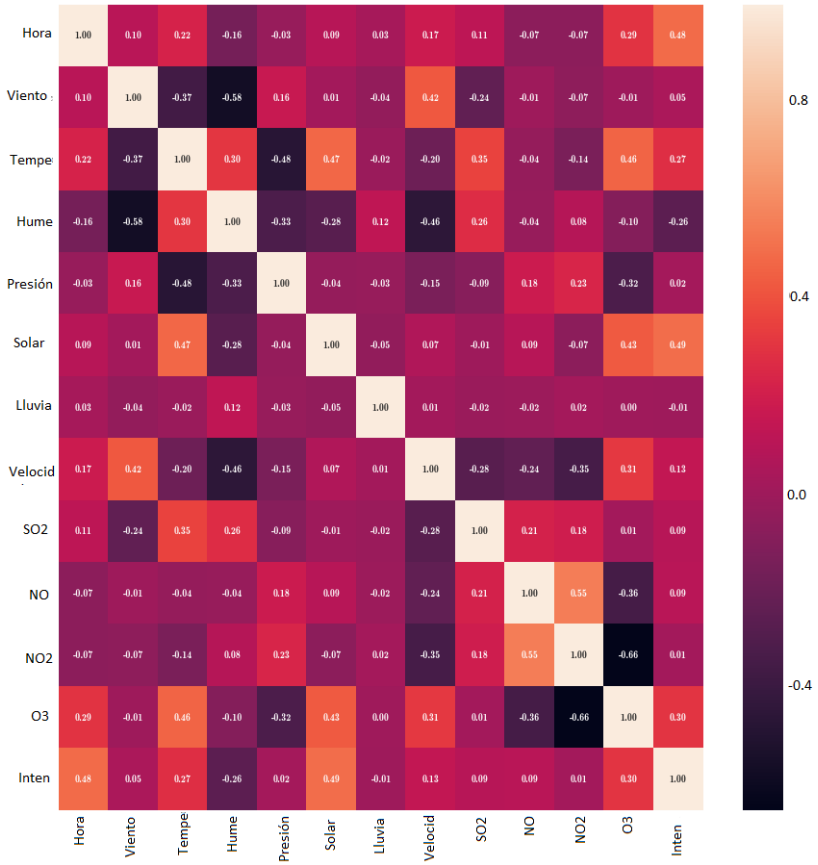


Figura 4.10: Matriz de correlaciones de las variables para predecir los contaminantes

Varios modelos de regresión lineal son desarrollados y entrenados para cada contaminante con el fin de elegir el que menos error produce durante la predicción de los datos. Las técnicas de regresión lineal con gradiente descendente, árbol de decisión, bosque aleatorio y ANN son las empleadas para generar los distintos modelos. A continuación se describe los modelos desarrollados.

- **Regresión lineal:** la regresión lineal analiza la dependencia entre variables independientes para explicar la relación que existe entre ellas [82]. En este caso se aplica una regresión lineal multivariable donde se analiza la dependencia de una variable independiente con múltiples variables independientes. La regresión lineal multivariable sigue la ecuación:

$$Y = X_1W_1 + X_2W_2 + X_3W_3 + ... + b \tag{4.2}$$

  Donde Y es la variable de interés, $X_1, X_2, X_3, ..., X_p$ son el conjunto de variables regresoras y b es la parcialidad o *bias*.

  Cada contaminante tendrá su modelo de regresión lineal, el cual sigue la ecuación descrita, y donde cada variable regresora corresponde a las variables con mayor dependencia de acuerdo a la matriz de correlacción.

- **Árbol de decisión:** los árboles de decisión son mayormente utilizados en problemas de clasificación, pero en problemas de regresiones lineales también han sido utilizados mostrando buenos resultados. El algoritmo de árbol de decisión divide el espacio de las variables regresoras en varias regiones distintas y no sobrepuestas. El modelo predice los valores de la variable de interés siguiendo reglas de decisión las cuales son inferidas de las variables regresoras [80].

- **Bosque aleatorio:** es un algoritmo que utiliza una combinación de árboles de decisión. La cantidad de árboles de decisión que componen el bosque aleatorio es conocido como número de estimadores [80]. En este caso se utilizó un modelo de bosque aleatorio con 10 estimadores.

- **ANN:** este tipo de redes compuestas por perceptrones presentan mejoras en la precisión de predicciones con un consiguiente incremento de complejidad en la arquitectura del modelo. El modelo entrenado está compuesto por una capa de entrada, una capa intermedia de 50 perceptrones y una capa de salida con un perceptrón. El número de perceptrones utilizados en la capa de entrada depende de la cantidad de variables para la predicción de cada contaminante. Las funciones de

activación utilizadas de cada perceptrón son *tanh* y *ReLU*, por lo que se tiene dos redes ANN (una basada en *tanh* y otra basada *ReLU*).

Los modelos son implementados usando la librería Scikit-learn el cual provee las funciones necesarias para implementar los modelos indicados y otras funciones previas al entrenamiento de los modelos. Como se indicó anteriormente, cada contaminante presenta un conjunto de datos diferente entre sí debido a que se agrupan en el conjunto de datos las variables con mayor correlación para cada contaminante. Posteriormente, los conjuntos de datos son divididos aleatoriamente en un 80 % para entrenamiento de los modelos y un 20 % para pruebas de los modelos. Esta división del conjunto de datos es la más común realizada durante el proceso de entrenamiento de modelos de aprendizaje automático y está destinada a evitar un sobre-entrenamiento o especialización del modelo a la misma vez que sirve para conocer la precisión del mismo. Los modelos entrenados son almacenados en un directorio dentro del repositorio habilitado con HDFS.

El proceso de distribución de modelos predictivos como servicios utiliza estos modelos para distribuirlos internamente y habilitar la predicción en tiempo real. El proceso está compuesto por el subproceso de desarrollo del servicio y registro del servicio. El subproceso de desarrollo de servicio utiliza los modelos entrenados y almacenados en el repositorio HDFS para integrarlos a una aplicación web basada en Python Flask. De esta manera, la aplicación web expone una API REST para receptar solicitudes del tipo POST con los datos de entrada del modelo. La aplicación web es aislada en un contenedor Docker para brindar el servicio. El subproceso de registro del servicio permite cargar la imagen Docker de la aplicación web al catálogo de servicios (Docker registry).

El proceso de predicción en tiempo real está constituido por los subprocesos de procesamiento en tiempo real, y predicción. El subproceso de procesamiento en tiempo real tiene el objetivo de cargar y preparar los datos previos a la solicitud de predicción. Los datos procedentes de VLCi en tiempo real son enviados al bróker de comunicaciones (Apache Kafka) tal y como se detalló en la subsección 4.4.2. El subproceso de procesamiento en tiempo real utiliza el API SparkSQL para cargar estos datos en *dataframes*. El *dataframe* es temporal y tiene una validez de una hora, la cual es la tasa de generación de datos por parte de la fuente VLCi. Los datos del *dataframe* son enviados a través de una petición POST a la aplicación que contiene el modelo de predicción. Las predicciones generadas contienen los valores de los contaminantes de las próximas 24 horas.

Sin embargo, estos valores por si solos son de difícil interpretación por lo cual se requiere de una escala de calidad del aire. La escala es utilizada para

etiquetar la calidad del aire en excelente, buena, mejorable, y deficiente basándose en los valores de los contaminantes. La escala utilizada en la aplicación es la presentada por la red valenciana de vigilancia y control de la contaminación atmosférica, el cual sigue el Real Decreto 102/2011 relativo a la mejora de la calidad del aire [154]. Las Tablas 4.2 y 4.3 muestran los valores empleados para clasificar las predicciones de los contaminantes.

Tabla 4.2: Concentración de contaminantes y relación Calidad Aire

Calidad	O_3 **(ppm)**	NO **(ppm)**	NO_2 **(ppm)**	SO_2 **(ppm)**
Excelente	0 - 90	0 - 100	0 - 100	0 - 175
Buena	91 - 180	101 - 200	101 - 200	176 - 350
Mejorable	181 - 270	201 - 300	201 - 300	351 - 525
Deficiente	>270	>300	>300	>525

Tabla 4.3: Calidad Aire y representación

Calidad	**Contaminación**	**Color Asociado**
Excelente	Muy Baja	Azul
Buena	Baja	Verde
Mejorable	Elevada	Amarillo
Deficiente	Muy Elevada	Rojo

Una vez transformados los valores de los contaminantes a los índices de calidad del aire correspondiente, estos son almacenados en una tabla dentro de Apache Cassandra. Por otro lado, una aplicación web basada en Nodejs (*backend*) se conecta a la tabla de resultados de Apache Cassandra y consulta los datos empleando el lenguaje CQL. Por otro lado, el *frontend* de la aplicación utiliza la librería Leaflet para generar mapas interactivos. Los resultados de la consulta CQL son enviados al *frontend* utilizando la librería de Javascript Socket.io. Los datos recibidos son presentados en el mapa interactivo del *frontend*.

Del lado del sistema de IoT, el paciente cuenta con un *Smartphone* con el que se recibe los servicios propios del sistema IoT a través de una aplicación para dispositivos móviles basada en el protocolo MQTT. La aplicación recibe un enlace a la aplicación web (*frontend*) para la visualización de la predicción de la calidad del aire. La aplicación web basada en Nodejs despliega en el mapa la predicción de la calidad del aire. La predicción corresponde a la calidad del aire cerca de la ubicación del usuario en la próxima hora.

4.5 Evaluación del sistema de la apnea

La evaluación del caso de uso tiene el objetivo de validar el funcionamiento de la integración de la arquitectura de Big Data con un sistema IoT y con los servicios de una ciudad inteligente con el fin de extraer información útil para soportar y guiar el tratamiento de la apnea. El escenario de pruebas fue planteado para evaluar el servicio de predicción de los contaminantes en la ciudad, la extracción de la información de los datos del sistema IoT y el rendimiento del procesamiento de la arquitectura propuesta.

En las siguientes subsecciones se plantea el escenario de pruebas, la selección del mejor modelo para el servicio de predicciones del mejor lugar para realizar actividades de ocio, la verificación del uso de una instancia de la arquitectura de Big Data para extraer varios KPI relevantes y su visualización, y el rendimiento del procesamiento del Big Data generado por IoT.

4.5.1 Escenario de pruebas

La evaluación es realizada a través de varios conjuntos de datos extraídos del sistema IoT de la apnea y de la ciudad inteligente VLCi de Valencia. Los conjuntos de datos extraídos del sistema IoT de la apnea fueron recolectados de dos voluntarios que han sido diagnosticados por un especialista con apnea del sueño. Los voluntarios firmaron un consentimiento para que sus datos puedan ser empleados para experimentación. Los datos son anonimizados como parte del acuerdo de firmado por los voluntarios, es decir el conjunto de datos no contiene ningún dato identificativo del paciente (nombre, documento de identidad, número de seguro social, dirección de domicilio, etc.). La Tabla 4.4 detalla los conjuntos de datos recolectados para el análisis de datos. El conjunto de datos de la información de contexto proveniente de la ciudad inteligente fueron empleados para el desarrollo de los servicios de predicción de contaminantes para guiar el tratamiento de la apnea.

La arquitectura de Big Data fue integrada al sistema de la apnea IoT siguiendo el patrón de implementación de la arquitectura basado en *cloud computing* definida en el capítulo 3. La implementación fue realizada en un entorno de pruebas controlado. El sistema fue desplegado sobre varias máquinas virtuales en un servidor con las siguientes características: servidor FUJITSU Intel Xeon E3-1220 v5 3,00 *Gigahercio* (GHz) CPU, memoria RAM 64 *Gigabyte* (GB), y plataforma VMware vSphere. La Figura 4.11 muestra un diagrama del despliegue de la arquitectura para las pruebas sobre las diferentes máquinas virtuales creadas. La máquina virtual 1 co-

Tabla 4.4: Detalle conjuntos de datos del sistema apnea

Categoría	Variable	No. Records	Tamaño
Actividades Físicas	Contador de pasos	185	10 MB
Estado del sueño	Severidad Apnea	177	24,3GB
	Etapas del sueño	185	10 MB
	Ritmo cardíaco	13450	4,3 GB
Ambiente del sueño	Temperatura	13450	
	Humedad	13450	
Información de contexto	Clima	4248	735 MB
	Contaminantes	4248	845 MB

rresponde al sistema IoT de la apnea basado en la plataforma IoT FIWARE que se encuentra operativo y funcionando. Las máquinas virtuales 2, 3 y 4 corresponden a los módulos de los componentes de la arquitectura de Big Data para IoT.

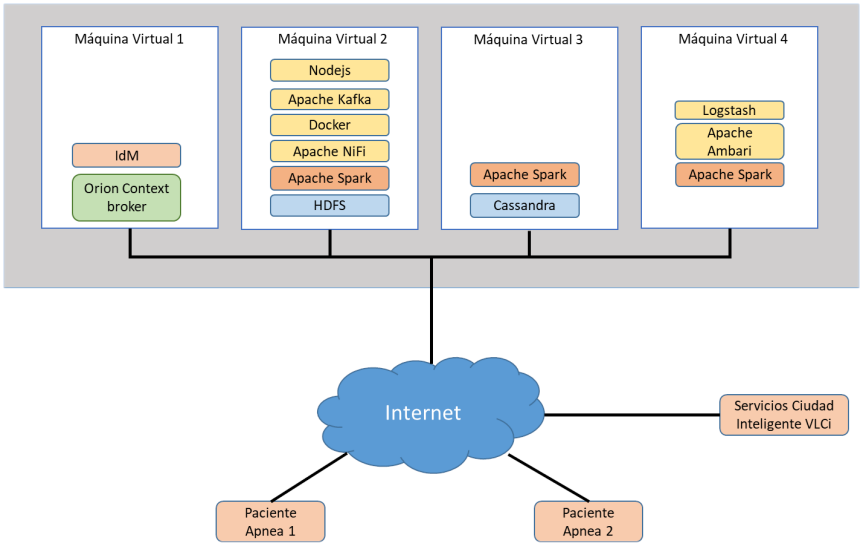


Figura 4.11: Esquema del despliegue de la arquitectura para el sistema de la apnea

4.5.2 Selección mejor modelo predicción de contaminantes

El error cuadrático medio relativo (RMSE, del inglés *Root-Mean-Squared Error*), el error medio relativo (MSE, del inglés *Mean-Squared Error*) y los coeficientes de determinación R^2 son los estimadores empleados para medir el promedio de error generado en los modelos de regresión lineal. Las Tablas 4.5, 4.6 y 4.7 muestran las comparativas de los errores cuadráticos medios relativos, errores cuadráticos medios y los coeficientes de determinación de cada modelo para cada contaminante. Los modelos que producen menos error son los basados en redes neuronales con función de activación *ReLu* y *tanh*. La elección de los modelos para cada contaminante se lo realiza evaluando los coeficientes de determinación R^2. Los coeficientes de determinación más próximos a 1 son los que predecirán los contaminantes con menor error. Los modelos seleccionados de acuerdo a las Tablas 4.5, 4.6 y 4.7 son los modelos MLP *tanh* para los contaminantes O_3 y NO_2, y MLP *ReLu* para los contaminantes NO y SO_2.

Tabla 4.5: Comparativa RMSE modelos de regresión

Contaminante	**Regresión lineal**	**Árbol Decisión**	**Bosque Aleatorio**	**MLP *tanh***	**MLP *ReLU***
O_3	13,45	16,64	15,91	12,69	12,73
NO	15,57	20,38	15,74	14,78	14,56
NO_2	15,92	19,90	18,09	14,98	14,96
SO_2	2,41	3,52	2,50	2,24	2,23

Tabla 4.6: Comparativa MSE modelos de regresión

Contaminante	**Regresión lineal**	**Árbol Decisión**	**Bosque Aleatorio**	**MLP *tanh***	**MLP *ReLU***
O_3	13,45	16,64	15,91	12,69	12,73
NO	15,57	20,38	15,74	14,78	14,56
NO_2	15,92	19,90	18,09	14,98	14,96
SO_2	2,41	3,52	2,50	2,24	2,23

Tabla 4.7: Comparativa de los coeficientes de determinación R^2 modelos de regresión

Contaminante	**Regresión lineal**	**Árbol Decisión**	**Bosque Aleatorio**	**MLP *tanh***	**MLP *ReLU***
O_3	0,7374	0,6028	0,6325	0,7654	0,7638
NO	0,2193	-0,4434	0,1854	0,3430	0,3540
NO_2	0,4992	0,2342	0,3530	0,5566	0,5564
SO_2	0,2022	-1,0435	0,1405	0,3054	0,3213

4.5.3 Verificación del uso de la instancia de la arquitectura de Big Data para IoT

Los conjuntos de datos son usados para extraer los KPI empleando la arquitectura diseñada. Los KPI proveen información acerca de las actividades física, el ambiente del sueño, las etapas del sueño y la evolución de la apnea en los pacientes. Esta información permite evaluar si el tratamiento es el adecuado para tratar la apnea. El análisis de los datos y el resultado es mostrado a continuación:

Actividad física

El indicador provee información a los médicos especialistas sobre la cantidad de actividad física ejecutada por sus pacientes. La evaluación se la realiza semanalmente debido a que no todos los días los pacientes realizan una actividad física (caminar). La Figura 4.12 muestra de ejemplo una semana la cantidad de pasos realizada por los pacientes. En ella se observa que los adultos mayores no realizan la actividad física todos los días.

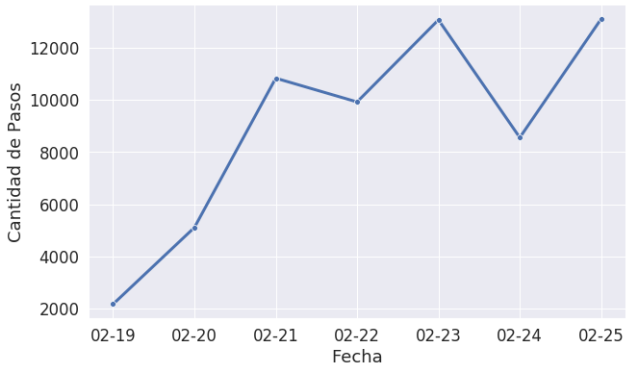


Figura 4.12: Actividad física realizada semanalmente por los pacientes

La analítica descriptiva aplicada a estos datos muestra que los pacientes en promedio realizan 8914 pasos a la semana, y que en días con menos actividad física (sedentarios) realizan 2165 pasos. La cantidad de pasos a la semana fue agrupada en las clases sedentario, actividad moderada, y muy activo con un rango de 5000 pasos de diferencia entre clases. Para la semana evaluada se visualizó que los adultos mayores estuvieron activos durante 5 días y 2 días fueron sedentarios. Esta clasificación permite determinar si los adultos mayores están siguiendo la recomendación de los doctores de realizar mayor actividad física a la semana.

Estado del sueño

El indicador provee información a los médicos especialistas sobre la calidad del sueño de sus pacientes. La Figura 4.13 muestra de ejemplo una semana la cantidad de minutos diarios con un sueño profundo y un sueño ligero.

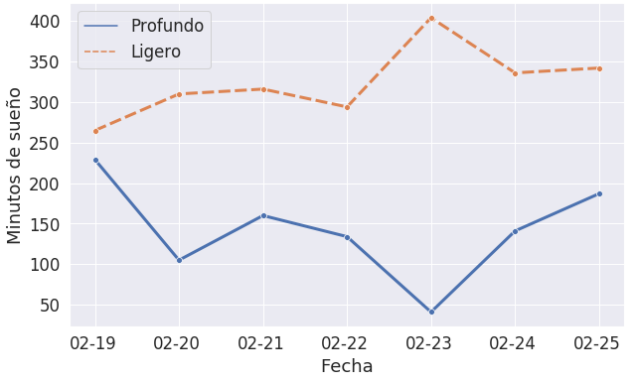


Figura 4.13: Estado del sueño de semanal en minutos de los pacientes

La analítica descriptiva aplicada a estos datos muestra que los pacientes en promedio duermen en promedio 487,7 minutos a la semana, y con un promedio de 151,88 minutos con un sueño profundo y 329,88 minutos con un sueño ligero. La apnea afecta a la cantidad de minutos con un sueño profundo. La información sobre el promedio a la semana del sueño profundo y sueño ligero indica a los doctores que sus pacientes están durmiendo mejor.

Severidad apnea

El indicador provee información a los médicos especialistas sobre la evolución de los eventos de la apnea de sus pacientes. La Figura 4.14 muestra la clasificación de la severidad de la apnea por paciente. En ella se observa que una persona con apnea presenta una alta cantidad de eventos de apnea con una severidad moderada. La información de la cantidad de eventos de apnea brinda información a los doctores acerca de la evolución favorable o desfavorable del tratamiento aplicado.

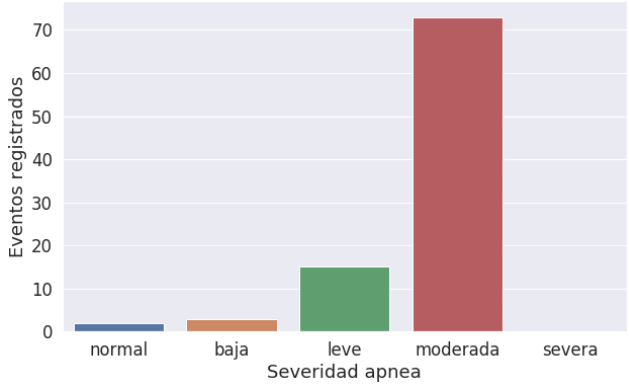


Figura 4.14: Clasificación de la severidad de la apnea

Ambiente del sueño

El indicador provee información a los médicos especialistas sobre si el sistema de apnea IoT se encuentra trabajando adecuadamente, es decir, si el sistema está controlando mediante el uso de los actuadores que la temperatura y la humedad de la habitación se mantenga en los niveles recomendados para pacientes con apnea del sueño. Del análisis de los datos recolectados sobre el ambiente del sueño se observa que la temperatura de las habitaciones de los pacientes está en promedio en $21,7°\text{C}$ y $63{,}5\,\%$ de humedad. Estos valores son los recomendados para el tratamiento de la apnea.

Visualización de los KPI y servicio de predicciones de contaminantes

Los KPI son compilados en el *dashboard* desarrollado para la visualización de los resultados. El *dashboard* muestra los KPI en 6 cuadros descriptivos y muestra los gráficos de líneas (actividad física) y barras (severidad apnea). La Figura 4.15 muestra la interfaz web para la visualización de los datos.

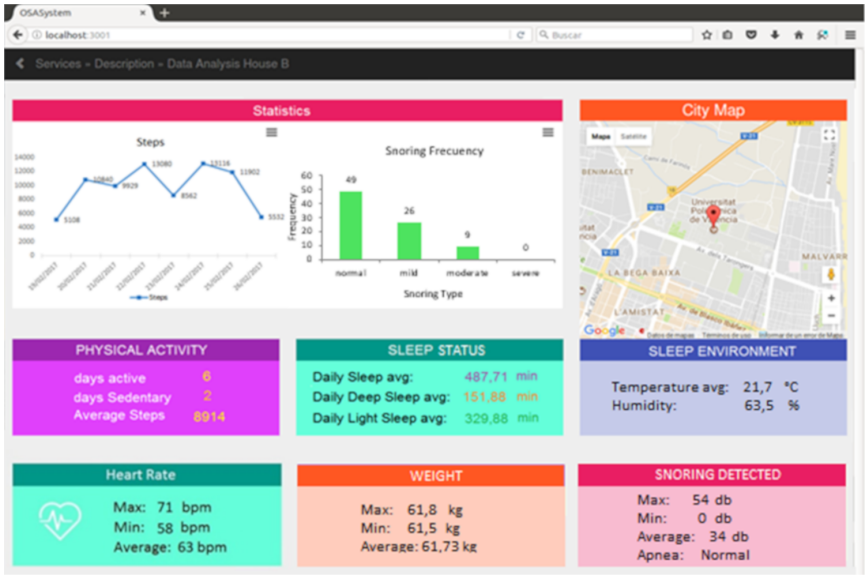


Figura 4.15: Interfaz web de usuario para el servicio de visualización de datos

El servicio de predicciones envía las notificaciones de la contaminación a los *smartphones* empleando los modelos seleccionados para la predicción. El envío de notificaciones con la predicción de la contaminación del aire fue probado con los datos consultados a la plataforma VLCi cada hora. En la Figura 4.16 se observa las predicciones que llegan a la aplicación móvil y la visualización en un mapa de los índices de calidad del aire distribuidos en la ciudad para una fácil interpretación.

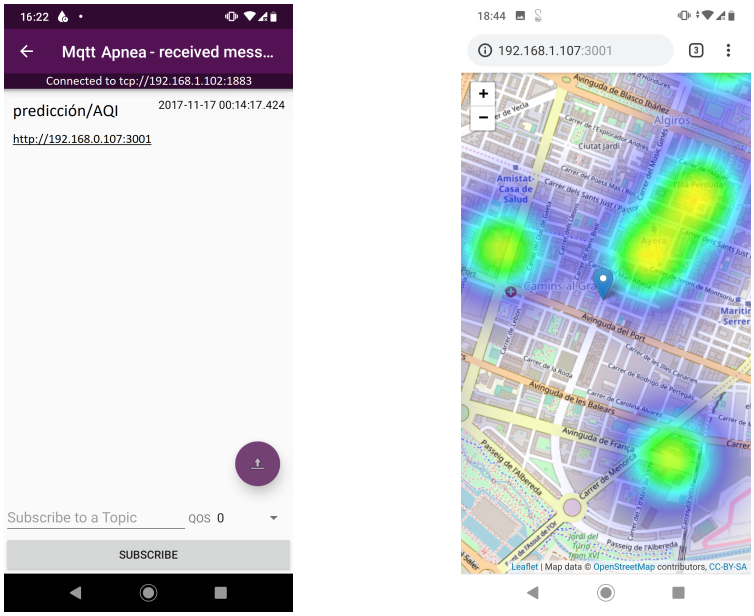


Figura 4.16: Servicio de predicción lugar menos contaminado en aplicación móvil

4.5.4 Rendimiento en el procesamiento de Big Data generado por IoT

La capacidad de procesamiento de la arquitectura de Big Data es evaluada empleando los conjuntos de datos descritos en la Tabla 4.4 La evaluación se centra en conocer el rendimiento cuando la arquitectura ejecuta los trabajos de extracción de los KPI para soportar y guiar el tratamiento de la apnea. El conjunto de datos fue dividido en varios subconjuntos con diferentes tamaños (3, 6, 16, 34, 68, 135, 270, 540, 1080, 2150 MB). El rendimiento es calculado evaluando para cada subconjunto de datos el tiempo que toma la arquitectura en procesar los datos. La Figura 4.17 muestra el rendimiento en (Mbps) para cada subconjunto de datos.

El rendimiento del procesamiento de los KPI presenta un incremento conforme se incrementa el tamaño del conjunto de datos. Podemos observar que el rendimiento de la arquitectura es bajo cuando el conjunto de datos procesado tienen un tamaño pequeño. A medida que incrementamos el tamaño del conjunto de datos el rendimiento no se ve afectado, es decir no se reduce y por el contrario se va incrementando. Además, se observa que a partir de los 135 MB el rendimiento se incrementa aproximadamente en 2,0 Mbps. Estos resultados son los esperados principalmente debido al uso

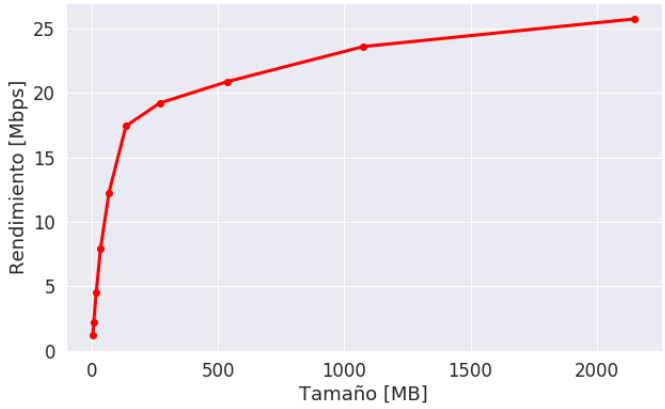


Figura 4.17: Rendimiento del procesamiento de KPI para el soporte de la apnea

del procesamiento paralelo y distribuido usando el modelo RDD de Apache Spark.

4.5.5 Comparativa de la arquitectura general con arquitecturas similares aplicadas a la salud

En el ámbito de la salud inteligente se han realizado varias propuestas de arquitecturas para la gestión del gran volumen de datos generados por los dispositivos IoT con el fin de extraer información útil. En esta evaluación consideramos las siguientes características importantes relacionadas con las 3Vs del Big Data y de acuerdo a los requerimientos en este tipo de sistemas:

- Integración con múltiples fuentes: muestra la capacidad de la arquitectura para soportar varias fuentes de datos de IoT.
- Procesamiento de datos en tiempo real: muestra la capacidad de la arquitectura para soportar la velocidad de los datos con el fin de generar un tratamiento eficiente de los datos.
- Procesamiento de datos *batch*: muestra la capacidad de la arquitectura para soportar el tratamiento de un gran volumen de datos eficientemente.
- Análisis de los datos: muestra la capacidad de la arquitectura para generar análisis del tipo descriptivo, predictivo, de diagnóstico o prescriptivo del gran volumen de datos recolectado.

- Almacenamiento escalable: muestra la capacidad de la arquitectura para soportar el gran volumen de datos y que estos puedan ser reutilizados en los análisis.
- Generación y distribución de servicios: muestra la capacidad de la arquitectura para distribuir los servicios generados de la explotación de los datos a los usuarios finales.
- Visualización: muestra la capacidad de la arquitectura para desplegar los resultados de los datos de forma visual para una mejor interpretación de los datos.
- Seguridad: muestra la capacidad de la arquitectura para proteger los diferentes módulos de la arquitectura.
- Gestión: muestra la capacidad de la arquitectura para administrar los diferentes módulos de la arquitectura.

Varias arquitecturas han sido propuestas para la gestión del Big Data en entornos de IoT para la salud. Din et al. en [155] propusieron una arquitectura de Big Data para gestionar los datos generados por IoT en aplicaciones de la salud con soporte de procesamiento *batch*. Posteriormente, una actualización de la arquitectura permitió soportar la capacidad de procesamiento en tiempo real. La principal característica es patrón de arquitectura basada en flujo y filtros que lo hacen eficiente. A diferencia de la arquitectura de Big Data presentada en esta tesis que considera la visualización y la distribución eficiente de los servicios generados del análisis de datos para mejorar la toma de decisiones. Además de considerar la seguridad y gestión de la arquitectura.

Por otro lado, Suciu et al. en [156] proponen una arquitectura para el almacenamiento eficiente de los datos y un procesamiento a través de consultas a base de datos indexadas. La arquitectura provee seguridad y gestión de las plataformas que proveen el almacenamiento. Sin embargo, la arquitectura carece de un enfoque integral donde se consideren el procesamiento en tiempo real, la visualización, la integración de múltiples fuentes de IoT (plataformas IoT, dispositivos IoT, entre otras). Asimismo, Liu et al. en [158] propone una arquitectura donde no se consideran todas las etapas del ciclo de vida de los datos del Big Data. En comparación con la arquitectura propuesta en esta tesis que considera todos estos puntos y destaca por la posibilidad de integrarse con servicios provistos por una ciudad inteligente.

Finalmente, la arquitectura propuesta en [157] por Coban et al. considera todo el ciclo de vida del Big Data, pero no se considera la seguridad y gestión los cuales son requerimientos primordiales en sistemas de salud. En

comparación con la arquitectura propuesta en esta tesis, se considera la distribución de servicios derivados del análisis de los datos para cubrir el requerimiento de los sistemas de salud en sitios remotos o para la integración con otros sistemas de salud.

La Tabla 4.8 muestra un cuadro comparativo de las características destacadas en relación con arquitecturas similares.

Tabla 4.8: Comparativa arquitectura general con arquitectura similares en la salud

Característica	Arquitectura				
	[155]	[156]	[157]	[158]	General
Integración múltiples fuentes	•		•		•
Procesamiento Tiempo Real	•		•	•	•
Procesamiento *Batch*	•	•	•	•	•
Almacenamiento eficiente	•	•	•	•	•
Análisis	•		•	•	•
Visualización			•		•
Seguridad		•			•
Gestión		•			•
Distribución Servicios					•

4.6 Conclusiones

La monitorización remota de enfermedades crónicas es una de las aplicaciones IoT con más interés en el ámbito de la salud en los últimos años. La principal motivación es la mejora de los servicios de la salud con el uso de las tecnologías emergentes. Actualmente, los sistemas de monitorización de la apnea no están preparados para afrontar la generación de grandes volúmenes de datos por dispositivos IoT desplegados a gran escala. Por ello, en este capítulo se ha presentado el uso de una instancia de arquitectura de Big Data para IoT con el fin de resolver las limitaciones actuales de los sistemas de monitorización de la apnea del sueño.

En este primer caso se ha validado la gestión de los datos generados por dos fuentes distintas de datos de IoT (plataforma IoT y datos abiertos). La gestión de los datos masivos de la aplicación de IoT permitió extraer información útil que permita monitorizar y controlar la enfermedad respiratoria crónica de la apnea del sueño. El sistema de la apnea de IoT incorpora dos nuevos servicios basados en la información extraída. El primer servicio

extrae varios KPI para controlar el avance en el tratamiento de la apnea de cada paciente. De esta manera, los especialistas médicos pueden verificar que tan efectivo es el tratamiento y tomar decisiones. Por otro lado, el segundo servicio predice el nivel de calidad del aire de la ciudad para que los adultos mayores puedan determinar el mejor momento y lugar para realizar sus actividades de ocio sin que esto repercuta en su tratamiento. De esta forma, los adultos mayores tienen autocontrol de su enfermedad crónica y toman decisiones por su cuenta.

Por otro lado, las plataformas *open source* utilizadas en la instanciación de la arquitectura han demostrado ser adecuadas para superar los retos en el manejo del gran volumen de datos generados por IoT. La herramienta Apache NiFi permitió desarrollar un mecanismo de integración con la plataforma IoT de manera fácil e intuitiva. Por otro lado, el uso de la API SparkSQL de Apache Spark para el procesamiento de datos permitió el desarrollo de KPI relevantes para el control de la apnea. Los *dataframes* permiten la extracción de un análisis descriptivo de manera rápida y sencilla con apoyo de la abstracción RDD de Apache Spark. Los resultados en el procesamiento muestran la eficiencia del uso de un clúster Apache Spark para tratar los datos generados en este caso de uso.

Capítulo 5

Sistema AAL para la detección de caídas usando IA

5.1 Introducción

Uno de los casos de uso que más interés ha despertado de los entornos IoT para la salud es la monitorización de las actividades del diario vivir. Este caso está enmarcado en lo que se conoce como AAL, el cual está destinado a mejorar la calidad de vida y bienestar de los adultos mayores a través del uso de IoT. Además, la aplicación está alineada con los objetivos que persigue el programa europeo AAL - *Active and Healthy Ageing* (AHA) para garantizar un envejecimiento activo y saludable a los adultos mayores.

Los adultos mayores corresponden a una población vulnerable que puede ser afectada por enfermedades y afecciones graves. Una de estas afecciones que sufre esta población vulnerable son las caídas. Las caídas en adultos mayores son consideradas de alta peligrosidad debido a las múltiples lesiones y fracturas que pueden ocasionar. De acuerdo con la Organización Mundial de la Salud (OMS), cerca del 30 % de personas de edades superiores a los 65 años sufre al menos una caída por año [159]. Estos accidentes ocurren mayormente en el interior de sus hogares por lo que resulta difícil detectar y

alertar a los servicios médicos de manera oportuna. Para ello, los sistemas AAL están destinados a detectar los eventos de caídas en tiempo real y alertar a los sistemas de emergencia y cuidadores a tiempo.

Los enfoques actuales para afrontar este requerimiento de alerta temprana están basados en *cloud computing* en su mayoría y en la actualidad presentan muchas limitaciones cuando son desplegados a gran escala. Entre las limitaciones se encuentra la latencia de red, la congestión de red y el mayor consumo de ancho de banda. Estas limitaciones son causadas por el gran volumen de datos generados por los sensores destinados a la monitorización remota. Además, estas limitaciones afecta directamente a los tiempos de detección de las caídas y por consiguiente dilatan el tiempo de respuesta de los servicios de emergencia para atender estás alertas. En este caso de uso se aplica una instancia de la arquitectura de Big Data para IoT especificada en el Capítulo 3 siguiendo un patrón de implementación basado en la combinación de *fog computing* y *cloud computing* para reducir las limitaciones de los enfoques actuales y mejorar los tiempos de detección de las caídas.

En este capítulo se presenta la motivación y trabajos relacionados con la detección de caídas usando IoT, Big Data e IA. Además, el capítulo presenta la relación existente entre el caso de uso y la arquitectura descrita en el capítulo 3 y presenta una instancia de la arquitectura para solucionar los problemas del volumen generado por IoT en este tipo de sistemas. Por otra parte el capítulo detalla la implementación del sistema y el diseño de los servicios y aplicaciones para potencializar las actuales soluciones. Finalmente, el capítulo presenta una evaluación del sistema donde se identifican las ventajas y desventajas del uso de una instancia de la arquitectura propuesta en este tipo de casos.

5.2 Motivación y trabajos relacionados

En la literatura actual, los sistemas de caídas son clasificados en sistemas basados en vídeo, basados en el ambiente, y basados en *wearables*. Los sistemas basados en vídeo utilizan cámaras para monitorizar las actividades diarias de las personas [160] [161] [162]. Este enfoque presenta un alto costo, limitada área de cobertura e irrumpen en la privacidad de las personas. Por otro lado, los sistemas basados en el ambiente están caracterizados por el uso de sensores de ambiente para detectar vibraciones en el suelo y ondas electromagnéticas o medir la temperatura usando sensores infrarrojos [163] [164] [165]. Este enfoque presenta ventajas como el bajo costo, mayor área de cobertura, no irrumpe la privacidad de las personas a costa de una baja precisión en la detección de las caídas. Finalmente, los sistemas basados

en *wearables* están caracterizados por el uso de sensores embebidos que pueden ser llevados por las personas en el pecho, cintura, o muñecas [166] [167] [168]. Este enfoque presenta ventajas como el bajo costo, ligeros de llevar, bajo consumo de energía a costa de la dependencia en la batería y que las personas deben llevar obligatoriamente los sensores para la detección de las caídas.

Por otro lado, la detección de las actividades del diario vivir son realizadas a través del uso de modelos basados en algoritmos de *machine learning* y *deep learning*. Los algoritmos como SVM y K-NN en combinación con mecanismos de extracción de variables han permitido alcanzar una precisión del 98 % y 96,13 %, respectivamente [169]. Mientras tanto que los algoritmos de *deep learning* basados en RNN-LSTM han permitido alcanzar una precisión del 98.57 % [167]. El uso de algoritmos de *deep learning* ha mejorado la precisión de los modelos. Dado que los algoritmos de *machine learning* y *deep learning* requieren mayores capacidades computaciones, estos son desplegados en una infraestructura de *cloud computing*.

Las limitaciones del uso de *cloud computing* restringen la capacidad de estos sistemas. Por ello, algunos sistemas han sido propuestos basados en *fog computing*. La idea principal de estos sistemas es llevar el servicio de detección más cerca de los dispositivos IoT. Los sistemas están basados en el uso de un *gateway* IoT [170] o el uso de teléfonos inteligentes como dispositivos *fog computing* [171]. Estos nodos *fog* presentan la capacidad de detectar las caídas a través del despliegue de los modelos de predicción basados en algoritmos de *machine learning* (SVM, K-NN, *Self-Organizing Maps* (SOM), árboles de decisión). Este enfoque ha demostrado un mejor tiempo de inferencia (tiempo de detección de los patrones) en relación a los basados en el *cloud* [171]. Si bien se logra bajar los tiempos de inferencia en relación a los desplegados en el *cloud*, la precisión de los modelos desplegados en nodos *fog* se considera mejorable a través del despliegue de los modelos basados en *deep learning*.

Sin embargo, el despliegue de modelos basados en *deep learning* en nodos *fog* es una tarea retadora debido a que los dispositivos utilizados mayormente como nodos *fog* (*gateways* IoT) presentan limitadas capacidades computacionales y de energía. Los pocos esfuerzos propuestos enfocados en *fog computing* para solventar los problemas de los sistemas basados en *cloud* no utilizan algoritmos de *deep learning* para la detección de caídas. Por el contrario, utilizan algoritmos de *machine learning* los cuales no son muy precisos en sus detecciones.

Varios trabajos han sido propuestos para distribuir la inteligencia a nodos *fog*. Uno de los enfoques utilizados es la distribución parcial de los modelos

de *deep learning* en nodos *fog* [172]. Esta estrategia plantea la división de los modelos y distribuir parte del modelo a los nodos *fog* y el resto del modelo se mantiene en el *cloud*. Este enfoque mostró una reducción del volumen de datos enviados al *cloud*; como resultado, el consumo de ancho de banda se redujo considerablemente. Si bien este enfoque mostró un mejoramiento en las limitaciones de los sistemas actuales, la partición de los modelos agregó complejidad al desarrollo de aplicaciones. Un enfoque más simple es la distribución completa de los modelos a dispositivos de borde. El uso cotidiano de los *Smartphone* han concebido la idea de utilizar estos dispositivos para la ejecución de los modelos de *deep learning* [173]. Por otro lado, el uso de las tecnologías de virtualización ligera en los nodos *fog* han abierto las posibilidades para el despliegue de aplicaciones al borde de la red [132]. Utilizando este enfoque se ha estudiado el despliegue de aplicaciones basadas en reglas para procesar y analizar los datos de sensores en nodos *fog* [174] y para el despliegue de modelos SVM [175].

La distribución de los modelos *deep learning* en nodos *fog* apoyados con las tecnologías de virtualización ligera no ha sido analizado y constituye la principal motivación en el desarrollo de este trabajo. A diferencia de los trabajos relacionados, este trabajo incluye una instancia de la arquitectura para gestionar el volumen de datos generados de IoT, detalla los procesos de extracción de información usando técnicas de Big Data y el despliegue de los modelos de *deep learning* a los nodos *fog*.

5.3 Relación de la arquitectura del sistema con la arquitectura general

El principal objetivo de este capítulo es demostrar la utilidad de la arquitectura general presentada en el capítulo 3 por medio de la definición de una instancia de la arquitectura para el desarrollo de un sistema de detección de caídas usando modelos de *deep learning* distribuidos en nodos *fog*. Además, el caso de uso en particular permite la verificación del patrón basado en *Fog-Cloud Computing* para la implementación de la arquitectura y su particular distribución de procesos y funcionalidades tal y como se definió en la subsección 3.7.2.

Los requerimientos funcionales del sistema en este caso de uso y su relación con los requerimientos generales definidos en la sección 3.2 son estructurados en la Tabla 5.1. En este caso, el requerimiento funcional particular es la detección temprana de situaciones de riesgo (caídas) típicas de los entornos AAL basados en IoT. Hay que mencionar que el requerimiento de visualización R8 no es considerado para este caso de uso, pero puede ser

implementado en un futuro dado que la arquitectura global si consideró en su diseño el soporte al requerimiento de visualización.

Tabla 5.1: Requerimientos del sistema AAL para detección de caídas

Requerimiento	**Relación Requerimientos Generales**
El sistema debe ser capaz de soportar las conexiones de *wearables* IoT y de *gateways* IoT.	**R1.** Soporte de conexión con las múltiples fuentes de IoT.
El sistema debe ser capaz de gestionar y normalizar el formato JSON utilizado por los *wearables* IoT para el envío de datos.	**R2.** Soporte para la conversión de formatos de datos de IoT.
El sistema debe ser capaz de procesar en tiempo real los datos para detectar los patrones de caídas. Este tipo de procesamiento debe ser realizado en etapas tempranas en la generación de datos para cumplir con el requerimiento particular del sistema.	**R3.** Procesamiento de datos de IoT en tiempo real.
El sistema debe ser capaz de soportar el preprocesamiento masivo de datos para generar conjuntos de datos que permitan generar modelos de *deep learning*.	**R4.** Procesamiento de datos de IoT tipo *batch*.
El sistema debe ser capaz de soportar el almacenamiento de datos en etapas tempranas de la generación de datos para reducir la cantidad de datos enviados al *cloud*. Además, debe soportar el almacenamiento de los conjuntos de datos utilizados en la generación de los modelos de *deep learning*.	**R5.** Soporte de almacenamiento.
El sistema debe ser capaz de realizar un análisis predictivo de los datos con el fin de detectar los patrones de caídas.	**R6.** Análisis de datos.
El sistema debe ser capaz de distribuir el servicio de detección de caídas a nodos *fog* (*gateways* IoT).	**R7.** Soporte en la generación y distribución de servicios.
El sistema debe ser capaz de gestionar los *gateways* IoT remotamente para verificar el correcto funcionamiento durante la detección de caídas.	**R9.** Gestión de los sistemas IoT.
El sistema debe ser capaz de tomar ventaja de las características que proporciona las tecnologías *fog computing* y *cloud computing* para la implementación del sistema.	**R10.** Soporte de múltiples tecnologías computacionales.

El sistema debe ser capaz de proteger los datos compartidos por los *gateways* IoT, así como de los servicios que proporciona el sistema.	**R11.** Seguridad.
El sistema debe ser capaz de garantizar la privacidad de los datos a través del procesamiento de los datos cercano al dueño de los datos. De esta manera, el dueño de los datos controla en todo momento el buen uso de sus datos.	**R12.** Soberanía de datos.

Desde el punto de vista conceptual de la arquitectura, la instancia de la arquitectura de Big Data cumple el rol de proveedor de tecnologías habilitadoras de Big Data y de proveedor de servicios. Mientras tanto que los proveedores de datos IoT son los dispositivos *wearables* IoT y sus correspondientes *gateways* IoT. El rol de consumidor de servicios es protagonizado por los propios proveedores de datos IoT, en este caso *gateways* IoT que utilizan el servicio de detección de caídas generado y distribuido por el proveedor de servicios.

5.4 Sistema de detección de caídas usando IA y con soporte de Big Data

La detección de caídas de adultos mayores se realiza remotamente (en sus hogares) a través de un sistema basado en IoT. La aplicación AAL de este caso de uso está alineada con los objetivos del proyecto ACTIVAGE para el soporte del envejecimiento activo y saludable. En esta sección se describe la instancia de arquitectura de Big Data para IoT del sistema de caídas, la implementación de la arquitectura basada en el patrón de implementación *fog-cloud computing* de la arquitectura, definida en la sección 3.7.2. Además, la sección detalla la instanciación e implementación de los procesos para recolección de datos de las fuentes de IoT, extracción de información y predicción en tiempo real.

5.4.1 Instanciación de la arquitectura de Big Data

La arquitectura del sistema de detección de caídas con soporte de Big Data es una instancia de la arquitectura de Big Data para IoT descrita en el capítulo 3. La instancia detalla las tecnologías utilizadas para brindar las funcionalidades de la arquitectura según el punto de vista funcional. La Figura 5.1 muestra las tecnologías utilizadas en la instanciación de la arquitectura para este caso de uso en particular, conforme el patrón arquitectural de la vista funcional de la arquitectura (sección 3.5).

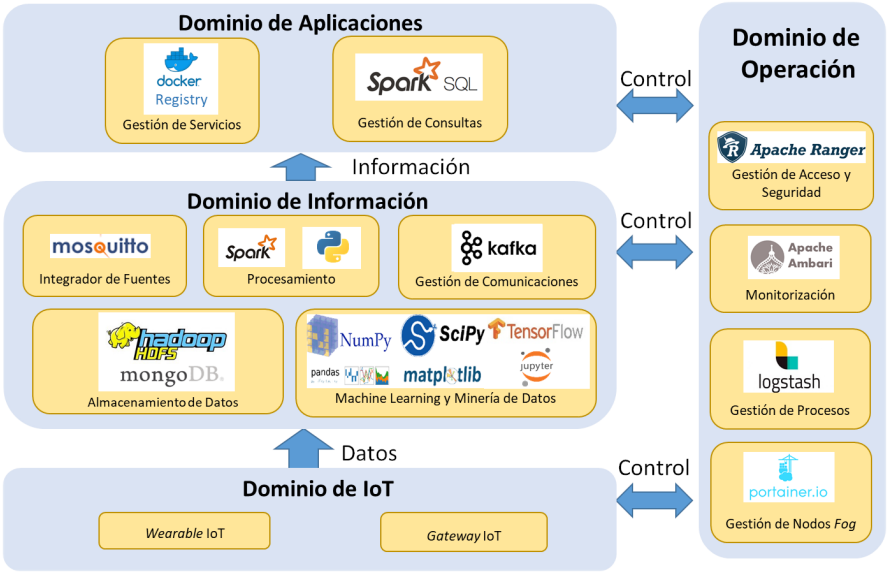


Figura 5.1: Instancia de la arquitectura de Big Data para IoT para el caso de uso de detección de caídas

Dominio IoT

El dominio IoT está compuesto por los dispositivos *wearables* IoT y su correspondiente *gateway* IoT.

El *wearable* IoT es un prototipo de cinturón diseñado especialmente para este sistema. El cinturón prototipo es equipado con un sensor acelerómetro para percibir los movimientos de los adultos mayores. Este dispositivo IoT está compuesto por: (a) un acelerómetro ADXL345 configurado para +/- 16g y 13 bits para convertir de análogo a digital, (b) por un microcontrolador NodeMcu v1.0 V3 basado en el módulo WiFi ESP8266, y (c) una batería Power Pack V1.2 de 3.7V y 3800 mA. La Figura 5.2 muestra el prototipo de dispositivo IoT antes de ser montado en un cinturón.

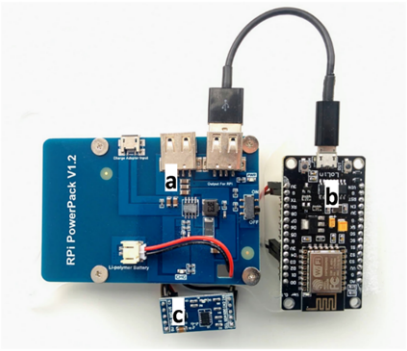


Figura 5.2: Prototipo *wearable* IoT

El *gateway* IoT actúa como dispositivo intermedio cuya funcionalidad es la de servir como pasarela de conexión entre los dispositivos IoT e Internet. Para ello, el *gateway* IoT está equipado con la tecnología inalámbrica WiFi para facilitar la conexión con los dispositivos IoT. Además, el *gateway* provee limitadas capacidades computacionales que serán aprovechadas por el sistema para distribuir algunos componentes funcionales de la arquitectura de Big Data para IoT. El *gateway* IoT fue implementado usando una Raspberry Pi 2 (RPi) model B con características: CPU Quad Core @900MHz ARMv7 Cortex-A7, memoria 1 GB LP-DDR2 400 MHz, y una MicroSD de 8GB para almacenamiento. El sistema operativo empleado fue Raspbian Stretch.

Dominio de información

- **Integrador de fuentes:** las funcionalidades de integración con las fuentes de datos (*wearables* IoT) son implementadas utilizando un bróker MQTT para habilitar la comunicación M2M con los dispositivos IoT y facilitar el intercambio de mensajes. Entre las características más destacadas del protocolo MQTT tenemos a sus niveles de QoS que garantizan la comunicación y sus niveles de seguridad que garantizan la privacidad en las comunicaciones. MQTT utiliza como mecanismos de seguridad la autenticación basada en usuario y contraseña o con certificados digitales y el transporte empleando SSL y TLS. La aplicación Mosquitto es utilizada para habilitar las funcionalidades de comunicación M2M [176]. Mosquitto es un bróker MQTT *open source* y ligero.
- **Almacenamiento:** las funcionalidades de almacenamiento consideradas para este caso de uso son de tres tipos: histórico de datos,

repositorio de modelos de *deep learning*, y repositorio de analíticas predictivas. El histórico de datos almacena los registros de caídas y no caídas que sirven para entrenar el modelo y también almacena los resultados de las consultas realizadas cada día al repositorio de analíticas predictivas con los resultados de las caídas por adulto mayor. El repositorio de modelos de *deep learning*, como su nombre lo indica, almacena los modelos generados luego del entrenamiento. Estos repositorios son implementados utilizando HDFS con el fin de proveer redundancia, escalabilidad y alta disponibilidad. Por otro lado, el repositorio de analíticas predictivas almacena de manera temporal los resultados del procesamiento en tiempo real y del servicio de predicción. El repositorio de analíticas predictivas es implementado a través de una instancia del sistema NoSQL MongoDB. Este sistema NoSQL permite el almacenamiento de documentos *Bynary Java Script Object Notation* (BSON) [177].

- **Procesamiento:** las funcionalidades de procesamiento son proporcionadas por Apache Spark para procesamiento tipo *batch* y por la librería Streamz de Python para procesamiento en tiempo real. Por un lado, la plataforma Apache Spark permite la ejecución de trabajos del tipo *batch* para transformar los datos almacenados (registros de caídas y no caídas). La transformación de los datos es realizada utilizando el módulo SparkSQL y su API *DataFrame* para la generación de conjuntos de datos de entrenamiento de modelos de *deep learning*. Por otro lado, la librería Streamz es utilizada para realizar un procesamiento en tiempo real debido a que es una librería ligera y que puede ser utilizada en dispositivos con limitados recursos computacionales. La librería Streamz basa su funcionamiento en DAG para el procesamiento de flujos de datos.

- ***Machine learning* y minería de datos:** las funcionalidades de este componente facilita el desarrollo de modelos de *deep learning* para la detección de caídas. Este módulo es implementado empleando las librerías Scikit-Learn, TensorFlow y Keras. En este caso, TensorFlow es empleado como *backend* de Keras para que ambas plataformas trabajen conjuntamente. Además, la interfaz web provista por Jupyter Notebook permite la creación de líneas de código interactivas que facilitan el desarrollo de los modelos de *deep learning* para la detección de caídas.

- **Comunicación:** estas funcionalidades permiten la comunicación interna entre *frameworks* y plataformas de Big Data. Para ello, el módulo es implementado utilizando el sistema de comunicación publicación/suscripción de Apache Kafka.

Dominio de aplicaciones

- **Gestión de servicios:** las funcionalidades de gestión de servicios de la arquitectura son implementados utilizando la plataforma Docker. La aplicación Docker registry es utilizada para gestionar los servicios generados. Además, Docker registry es configurada para uso particular y privado. De esta manera, las imágenes de las aplicaciones con los modelos de *deep learning* se mantienen privadas, seguras y alcanzables remotamente.
- **Gestión de consultas:** la gestión de consultas de los resultados del análisis predictivo es realiza utilizando las operaciones CRUD (*Create Read Update Delete*) de la instancia de MongoDB. Además, las API provistas por SparkSQL permitirán generar consultas puntuales sobre los datos almacenados en HDFS.

Dominio de operaciones

- **Gestión de acceso y seguridad:** las funcionalidades de seguridad son implementadas por Apache Ranger. Está herramienta facilita la gestión de seguridad y acceso a las plataformas basadas en Hadoop y Spark a través de la definición de políticas de acceso y seguridad.
- **Gestión de monitorización:** la monitorización del buen funcionamiento de las plataformas de Big Data utilizadas para implementar las funcionalidades de la arquitectura es realizada a través de Apache Ambari. Esta herramienta está orientada a la monitorización del clúster HDFS y Spark destinados al almacenamiento y procesamiento. De esta manera se controla la ejecución de los programas aplicación para procesar datos y el aprovisionamiento de recursos computacionales durante su ejecución.
- **Gestión de procesos:** las funcionalidades de gestión de procesos son implementadas usando la aplicación LogStash. Los *logs* generados durante la ejecución de los procesos de la arquitectura son concentrados en un solo repositorio para facilitar su análisis y resolución de fallos en los procesos de la arquitectura.
- **Gestor de nodos *fog*:** la implementación de las funcionalidades de gestión de nodos *fog* se realiza utilizando la aplicación Portainer UI [178]. Esta aplicación ligera habilita una GUI desde donde se puede gestionar remotamente a los nodos *fog* basados en contenedores y desplegar las aplicaciones y servicios aislados en contenedores.

5.4.2 Implementación arquitectura del sistema basado en el patrón Fog-Cloud Computing

El patrón seleccionado para la implementación de la instancia de la arquitectura es el basado en *Fog-Cloud Computing* con el fin de aprovechar los recursos computacionales provistos por los *gateways* IoT y extender las funcionalidades y procesos de la arquitectura más cerca de la fuente de generación del Big Data. En este caso de uso en particular la mejor opción es este patrón de implementación para poder distribuir los servicios de detección más cerca de la fuente y mejorar los tiempos de detección de caídas. La Figura 5.3 muestra la implementación del sistema siguiendo el patrón basado en *fog-cloud computing*.

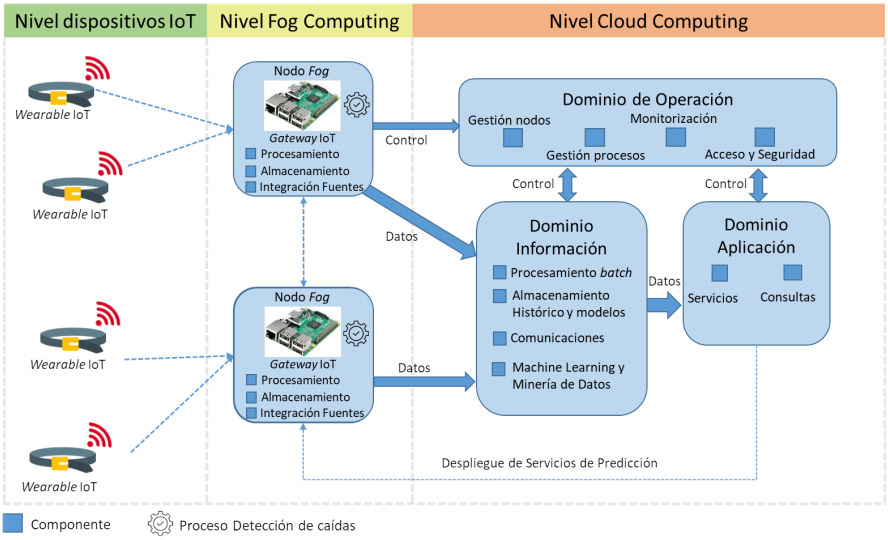


Figura 5.3: Patrón de implementación de la arquitectura basado en *fog-cloud computing*

El nivel de dispositivos IoT está encargada de percibir las actividades diarias a través de los dispositivos IoT *wearables* (cinturones). El nivel *Fog Computing* está compuesto por varios *gateways* IoT los cuales son considerados como nodos *fog* debido a su capacidad computacional. Los nodos *fog* implementan algunas funcionalidades del dominio funcional detalladas en la instancia de la arquitectura con el fin de soportar el proceso de predicción en tiempo real. El detalle de la distribución de las funcionalidades se verá en la subsección 5.4.3.

Por otro lado, el nivel *Cloud Computing* (centro de datos privado) está encargado de concentrar el gran volumen de datos generados por los nodos

IoT, procesar los datos y generar modelos basados en *deep learning* para la detección de las caídas a través de las funcionalidades de la arquitectura de Big Data para IoT. En este caso, las funcionalidades que se implementan en el nivel *cloud computing* son almacenamiento, comunicación, procesamiento, *machine learning* y minería de datos y las funcionalidades del dominio de control y aplicaciones de la arquitectura.

5.4.3 Nodos fog

Los nodos *fog* del sistema son diseñados para soportar la instanciación del proceso de predicción en tiempo real y la distribución de los servicios generados en el *cloud*. En este caso, los componentes funcionales de la arquitectura de Big Data para IoT que se extiende al nivel *fog computing* son: el procesamiento en tiempo real, el almacenamiento y la integración de fuentes.

Los recientes avances realizados para la implementación de la tecnología de virtualización de contenedores en dispositivos con limitados recursos han permitido que los conocidos *gateways* IoT sean reinventados con el fin de soportar eficientemente el despliegue de aplicaciones [132] [106]. Dado que las plataformas empleadas como *gateway* IoT presentan restricciones computacionales que pueden llegar a dificultar el procesamiento, el uso de las técnicas de virtualización en estos casos mejora sustancialmente la gestión de los limitados recursos computacionales. Si bien la aplicación de los contenedores en *gateways* IoT está en una etapa inicial de desarrollo, el uso de esta tecnología a nivel *fog computing* presenta un futuro prometedor. En tal virtud, el diseño de los nodos *fog* del sistema está basado en la tecnología de virtualización de contenedores para soportar la distribución de los servicios de predicción de manera eficiente.

La plataforma de virtualización de contenedores Docker es empleada en el diseño de los nodos *fog* debido a que es una plataforma madura, de código abierto y cuenta con una amplia comunidad para brindar soporte. Además, Docker presenta un mecanismo para la gestión de las imágenes de los contenedores (Docker Registry) que complementado con el mecanismo *pull/push* facilita la distribución de los contenedores. La Figura 5.4 muestra técnicamente el diseño de un nodo *fog*.

El nodo *fog* requiere de 4 contenedores para cubrir las funcionalidades necesarias para ejecutar el proceso de predicción en tiempo real. A continuación se detallan los 4 contenedores Docker:

- El contenedor de la aplicación de pre-procesamiento aísla un programa aplicación escrito en Python usando la librería Streamz. Además, el

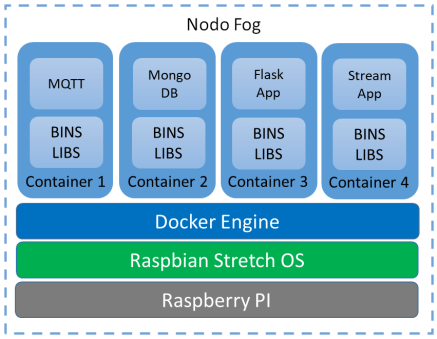


Figura 5.4: Nodo *fog* basado en Docker

programa aplicación utiliza la librería Paho MQTT para comunicarse con el contenedor de la aplicación Mosquitto y extraer los datos.

- El contenedor de la aplicación MongoDB aísla una instancia de base de datos que se encuentra escuchando en el puerto 27017. El servidor contiene una base de datos para almacenar los datos de las caídas.
- El contenedor del bróker MQTT aísla la aplicación Mosquitto que se encuentra en la escucha de peticiones de subscripción/publicación en el puerto 1883.
- El contenedor de la aplicación de predicción aísla un programa aplicación del tipo web que contiene el modelo de *deep learning*. Esta aplicación se encuentra escuchando en el puerto 5000 a por peticiones POST a través de su REST API para la predicción de caídas.

5.4.4 Desarrollo de los modelos de predicción

La explotación de los datos de IoT empleando técnicas de IA genera servicios de predicción más fiables y robustos en comparación con los sistemas tradicionales basados en reglas. Sin embargo, el rendimiento (alta precisión y robustez) de los modelos es directamente proporcional a la complejidad de los algoritmos empleados para generar los modelos. Las técnicas de *deep learning* permiten desarrollar modelos con mejores rendimientos que los desarrollados con las técnicas de *machine learning*, pero requieren de un gran volumen de datos y mayores capacidades computaciones para su desarrollo. En este caso de uso se plantea el desarrollo de modelos de predicción utilizando técnicas de *deep learning* para mejorar la eficacia de las predicciones.

El desarrollo de los modelos de predicción está soportado por la instancia de la arquitectura de Big Data y sus procesos bien definidos. El proceso de la arquitectura relacionado con el desarrollo de los modelos de predicción de caídas es el proceso de extracción de información. Este proceso está compuesto por tres subprocesos: carga de datos, pre-procesamiento, y análisis predictivo. La Figura 5.5 muestra la instancia del proceso de extracción de información implementada para este caso de uso.

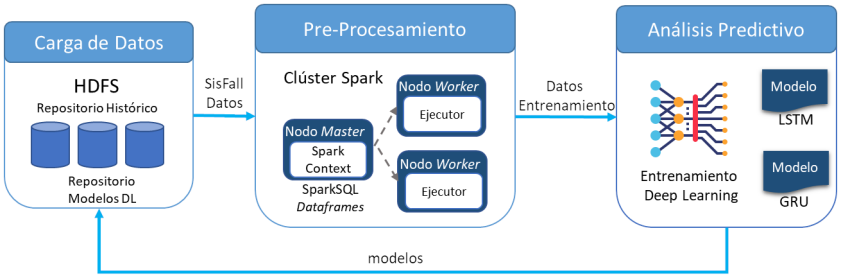


Figura 5.5: Instancia del proceso de extracción de información

El subproceso de carga de datos lee el conjunto de datos almacenado en el repositorio histórico HDFS que será utilizado para la generación del análisis predictivo. El conjunto de datos almacenado es el presentado en el desarrollo de SisFall por Sucerquia et al. [179] debido a que posee una cantidad de registros aceptable para entrenar modelos de *deep learning*. El conjunto de datos está compuesto por un total de 4497 registros tomados de 23 voluntarios adultos y 15 voluntarios ancianos. Cada uno de los ficheros o registro contiene datos del tipo serie de tiempo que describe una actividad del diario vivir (2701 registros) y tipos de caídas (1796 registros). Entre las actividades del diario vivir encontramos actividades como caminar, pararse, sentarse, trotar, acostarse; mientras que los tipos de caídas tenemos caídas de lado, de frente y de espalda. Cada registro está compuesto por una serie de marcas de tiempo, que se refieren a una posición en la secuencia y no a una porción de tiempo en el mundo real [82]. Cada marca de tiempo en los registros del conjunto de datos de SisFall representa 0,005 segundos del mundo real. Cada marca de tiempo contiene los valores del acelerómetro triaxial en las coordenadas x, y, z. Además, cada registro presenta una etiqueta que identifica la actividad que representa. La Figura 5.6 muestra un registro con los datos de una caída. Los registros tienen longitudes variables de marcas de tiempo por lo que requieren un tratamiento con el cual se pueda homogenizar el conjunto de datos.

El subproceso de pre-procesamiento de los datos prepara el conjunto de datos que será usado para el entrenamiento del sistema. La primera tarea en este subproceso es la homogenización de la longitud de los registros.

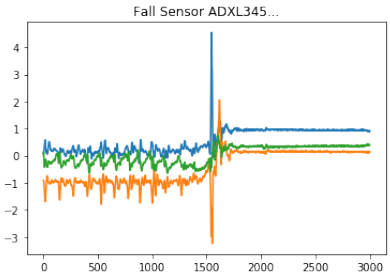


Figura 5.6: Registro que contiene los datos de una caída.

La menor longitud encontrada en el conjunto de datos cargado es de 2999 marcas de tiempo, mientras que el mayor es de 30000 marcas de tiempo. La homogenización de todos los registros a la longitud de 2999 asegura que todos los registros sean utilizados durante el subproceso de análisis predictivo. Los registros que superen esta longitud mínima son divididos en varios registros con lo que se obtiene más registros que serán usados durante el proceso de entrenamiento. De esta manera, todos los registros presentan las dimensiones $2999x3$ que hace referencia a los 2999 marcas de tiempo y a las 3 dimensiones (x, y, z) del sensor acelerómetro tri-axial.

La segunda tarea es identificar las marcas de tiempo que contienen los datos relacionados con las caídas y etiquetar las marcas de tiempo. Para ello, un nuevo registro es generado con dimensiones $747x1$ que contiene los valores 0 si la marca de tiempo no representa una caída y el valor de 1 si la marca de tiempo representa una caída. Las dimensiones $747x1$ de las etiquetas se corresponde con la salida del modelo.

Las dos tareas del subproceso de pre-procesamiento de datos son desarrollados empleando PySpark, el cual es la API de Apache Spark para el uso de lenguaje Python. Además, la utilización de librerías Python como Numpy o Matplot facilita la generación de los conjuntos de datos. El resultado del procesamiento son dos conjuntos de datos con las siguientes dimensiones $4000x2999x3$ y $4000x747x1$ que corresponden a un conjunto de datos con 4000 registros cada uno con dimensiones de 2999 marcas de tiempo y con valores x, y, z del sensor acelerómetro, y un conjunto de etiquetas de 4000 registros con 747 como longitud de las etiquetas con un valor de 0 o 1 como identificador de las marcas de tiempo referentes a las caídas.

El subproceso de análisis predictivo utiliza el algoritmo de *deep learning* RNN debido a que es el que mejor se adapta a los conjuntos de datos del tipo serie de tiempo [82]. Las dos variantes del algoritmo (GRU, LSTM) serán implementadas para posteriormente seleccionar el modelo más óptimo tras

un análisis del rendimiento de cada modelo. La implementación es realizada empleando la librería Keras y TensorFlow. El modelo a implementarse sigue la estructura por capas como se muestra en la Figura 5.7. La arquitectura del modelo está compuesto por cuatro capas principales y se detallan a continuación:

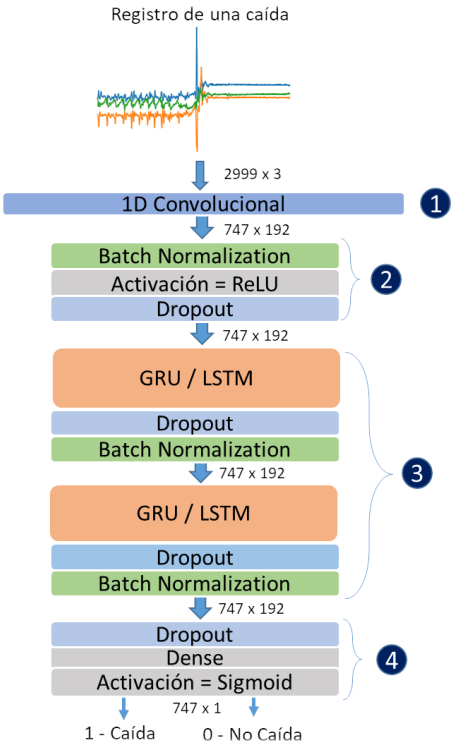


Figura 5.7: Modelo RNN (LSTM/GRU)

1. La primera capa corresponde a una etapa de filtros convolucionales. Estos filtros tienen su origen en el diseño de redes CNN para extraer variables de bajo nivel en la clasificación de imágenes. Sin embargo, la capa de filtros convolucionales de una dimensión (1D) ha sido usada en combinación con una arquitectura RNN en la literatura especializada para reducir las dimensiones de los datos y extraer variables de bajo nivel [180]. Las dimensiones de los filtros convolucionales están relacionadas con el tamaño de la matriz llamada *kernel* que recorrerá la matriz de datos en el proceso de convolución. El tamaño de *kernel* seleccionado para nuestro caso es de 15, lo que corresponde a un filtro 1D con un desplazamiento de 4 saltos en una sola dimensión (saltos de tiempo), ver Figura 5.8. La nueva matriz como resultado de

la convolución (operación producto escalar entre la matriz de datos de entrada y la matriz *kernel*) tendrá una dimensión que depende del tamaño del *kernel* k, el desplazamiento o saltos del *kernel* s al recorrer la matriz de entrada de tamaño n (ecuación 5.1), y del número de filtros aplicados a la matriz de entrada [181].

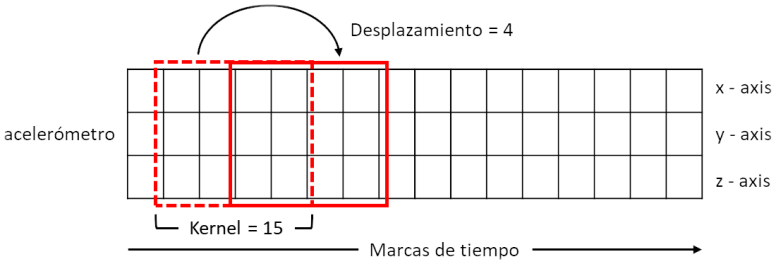


Figura 5.8: Funcionamiento Filtro 1D

$$size = \frac{n-k}{s} + 1 \tag{5.1}$$

La cantidad de filtros aplicados es producto de un proceso de experimentación donde se evalúan varios valores hasta encontrar el que mejores resultados produce (menos errores del modelo). En nuestro caso se ha determinado que con 192 filtros se obtienen los mejores resultados. Aplicando la ecuación 5.1 obtenemos como dimensiones $747x192$.

2. La segunda capa es una etapa *batch normalization*, activación con función $ReLU$ y regularización *Dropout*. La etapa *batch normalization* realiza una normalización o estandarización de los valores de entrada a través de un ajuste o escalado [182]. La etapa de activación con función $ReLU$ contiene la función de rectificación lineal $R(z) = max(0, z)$, la misma que permite el paso de los valores positivos sin cambiarlos, pero cambia los valores negativos por cero [183]. Por otro lado, la etapa de regularización *Dropout* desactiva aleatoriamente neuronas con cada entrenamiento. De esta manera se asegura que las neuronas no se especialicen o sobre-entrenen [184].

3. La tercera capa presenta una etapa de celdas LSTM o compuertas GRU dependiendo de la variante del modelo. La etapa está compuesta por 32 celdas en el modelo LSTM o 32 compuertas en el modelo GRU que leen la secuencia de tiempo de izquierda a derecha. La entrada es una secuencia de datos x_t y produce una salida con marcas de tiempo $t = 1, 2, 3, ..., 192$.

Cada celda de memoria LSTM presenta tres compuertas que están destinadas a controlar cuando nueva información puede entrar en la memoria (*input gate*), cuando parte de la información es olvidada (*update gate*) y cuando la información es usada en el resultado (*output gate*). Estas redes son capaces de retener información por un periodo largo de tiempo [185] [186]. La Figura 5.9 muestra la arquitectura de una celda de memoria LSTM.

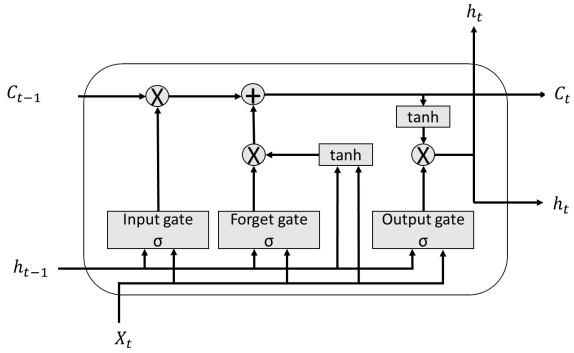


Figura 5.9: Arquitectura celdas de memoria LSTM

Las celdas de memoria LSTM ejecutan las siguientes funciones para controlar la información que pasa por cada compuerta (*gate*) [185]:

$$i_t = \sigma(W_{ix}X_t + W_{ih}h_{t-1} + b_i) \tag{5.2}$$

$$f_t = \sigma(W_{fx}X_t + W_{fh}h_{t-1} + b_f) \tag{5.3}$$

$$O_t = \sigma(W_{ox}X_t + W_{oh}h_{t-1} + b_o) \tag{5.4}$$

$$\tilde{C}_t = \tanh(W_{cx}X_t + W_{ch}h_{t-1} + b_c) \tag{5.5}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.6}$$

$$h_t = O_t * \tanh(C_t) \tag{5.7}$$

Donde i, f, o, c corresponde a la compuerta de entrada (*input gate*), compuerta de salida (*output gate*), compuerta de olvido (*forget gate*), y los vectores de activación de celda, respectivamente. Además, h_{t-1} es el estado de la celda escondida previa, y W corresponde a la matriz de pesos de cada compuerta (por ejemplo, W_{ih} es la matriz de pesos de la compuerta de entrada). Además, b se considera la parcialidad (*bias*) para cada compuerta y σ corresponde a la función *sigmoid*.

Cada compuerta GRU presenta una arquitectura similar a las LSTM pero con la diferencia de que estas no poseen una memoria interna y no usan una compuerta de salida. Por el contrario, las celdas GRU presentan dos compuertas que están destinadas a actualizar y reiniciar la información almacenada [187]. En la Figura 5.10 se puede observar la arquitectura de una celda de memoria GRU.

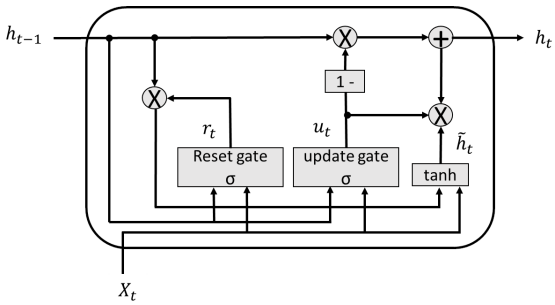


Figura 5.10: Arquitectura compuertas de memoria GRU

Las celdas de memoria GRU ejecutan las siguientes funciones para controlar la información que pasa por cada compuerta (*gate*) [187]:

$$u_t = \sigma(W_{ux}X_t + W_{uh}h_{t-1} + b_u) \tag{5.8}$$

$$r_t = \sigma(W_{rx}X_t + W_{rh}h_{t-1} + b_r) \tag{5.9}$$

$$\tilde{h}_t = \tanh(W_{cx}X_t + W_{hh}(r_t * h_{t-1} + b_c) \tag{5.10}$$

$$h_t = u_t * \tilde{h}_t + (1 - u_t) * h_{t-1} \tag{5.11}$$

Donde u, r, h corresponde a la compuerta de actualización(*update gate*), compuerta de reinicio (*reset gate*), y los vectores de activación

de celda, respectivamente. Además, h_{t-1} es el estado de la celda escondida previa, y W corresponde a la matriz de pesos de cada compuerta (por ejemplo, W_{uh} es la matriz de pesos de la compuerta de actualización). Además, b se considera la parcialidad (*bias*) para cada compuerta y σ corresponde a la función *sigmoid*.

Al igual que la segunda capa, una etapa de regularización *Dropout* y una de *batch normalization* son añadidas para reducir la complejidad del modelo y evitar el sobre-entrenamiento de la red neuronal.

Este conjunto de etapas GRU/LSTM, *Dropout* y *Batch Normalization* son duplicadas para alargar la capa con el fin de extraer más información.

4. La última capa está compuesta por una etapa densa (*dense*) o también llamada completamente conectada. Esta etapa está compuesta por neuronas perceptrones que se conectan entre sí. Estas neuronas emplean la función de activación *Sigmoid* con el fin de regular los valores entre 0 y 1 que son los valores a predecir. Los modelos emplean la variante unidireccional en lugar de la bidireccional para detectar casi inmediatamente la ocurrencia de una caída por lo cual no se espera que la secuencia termine para la detección. Esta configuración es particularmente seleccionada para reducir aún más el retardo que pueda generar el proceso de predicción.

Los modelos RNN (LSTM y GRU) son entrenados empleando el conjunto de datos SisFall propuesto por [179] y que previamente fue preparado durante el subproceso de pre-procesamiento de datos. El conjunto de datos posee 4000 registros de los cuales el 60 % de los registros corresponde a la etiqueta caídas (representada por 1) y el 40 % a la etiqueta actividad diaria cotidiana (representada por 0). Este porcentaje es importante puesto que si la diferencia entre etiquetas es muchísimo mayor, la red neuronal sería sobre-entrenada y provocaría errores. El conjunto de datos es dividido de manera aleatoria en datos para entrenamiento (90 % - 3600 registros) y datos para prueba (10 % - 400 registros). La división típica suele estar en proporciones 80-20, pero en este caso empleamos 90-10 puesto que los modelos de *deep learning* suelen requerir una mayor cantidad de datos para extraer información en comparación con los modelos de *machine learning*. Los modelos son entrenados empleando el optimizador *Adam* para el aprendizaje de los pesos de cada neurona en la red [188]. Finalmente, los modelos son almacenados en el directorio destinado a modelos de predicción en HDFS.

5.4.5 Distribución de modelos de predicción a nodos fog

La distribución del servicio de predicción a los nodos *fog* tiene como propósito reducir la latencia de tiempo que ocurre durante la detección de las caídas y la cantidad de datos que son enviados al *cloud*. Este proceso utiliza una instancia del proceso de distribución de modelos de predicción como servicios definidos por la arquitectura en la subsección 3.6.4.

El proceso de distribución de los modelos de predicción o detección de caídas como servicio sigue el esquema que muestra la Figura 5.11. Este proceso tiene como eje central el gestor de nodos *fog* y el módulo de registro de servicios de la arquitectura propuesta, implementados con la aplicación Portainer UI y Docker registry, respectivamente. Portainer UI explota la API remota de Docker a través del puerto 2375 y presenta una interfaz web segura empleando SSL/TLS y autenticación con usuario y contraseña. Las principales ventajas de utilizar este esquema de distribución son la gestión remota de los nodos *fog* y la distribución eficiente y segura de las imágenes Docker.

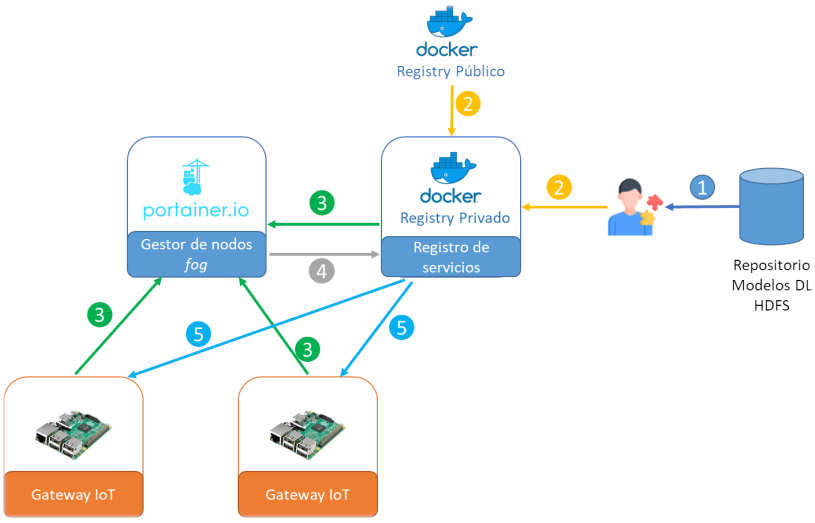


Figura 5.11: Proceso de distribución del servicio de detección de caídas

A continuación describimos los pasos en el proceso de distribución y gestión remota de los nodos *fog*:

1. El desarrollador de aplicaciones genera el programa aplicación para la detección de caídas usando los modelos de *deep learning* y genera el programa aplicación para el pre-procesamiento de flujos.

2. Las aplicaciones desarrolladas son aisladas en imágenes Docker y almacenadas en el módulo de registro de servicios. A su vez, el módulo de registro de servicios almacena las imágenes de las aplicaciones MongoDB y Mosquitto descargadas desde el Docker registry público.

3. Los nodos *fog* son registrados en el gestor de nodos *fog* para su gestión remota. El proceso de registro involucra la adición de un identificativo único por cada nodo *fog* y la dirección IP remota. Además, el repositorio del módulo de registro de servicios es registrado en el gestor de nodos *fog* para la posterior distribución de las imágenes Docker.

4. Desde el gestor de nodos *fog* se envían las solicitudes de descarga de las imágenes Docker a los nodos *fog* IoT.

5. Los nodos *fog* reciben las imágenes Docker. Y desde el gestor de nodos *fog* se ejecutan los contenedores que brindarán los servicios en el nodo *fog* remoto.

La Figura 5.12 muestra, como ejemplo, los contenedores ejecutados en un nodo *fog* a través de la aplicación Portainer UI de gestión remota de los contenedores virtuales.

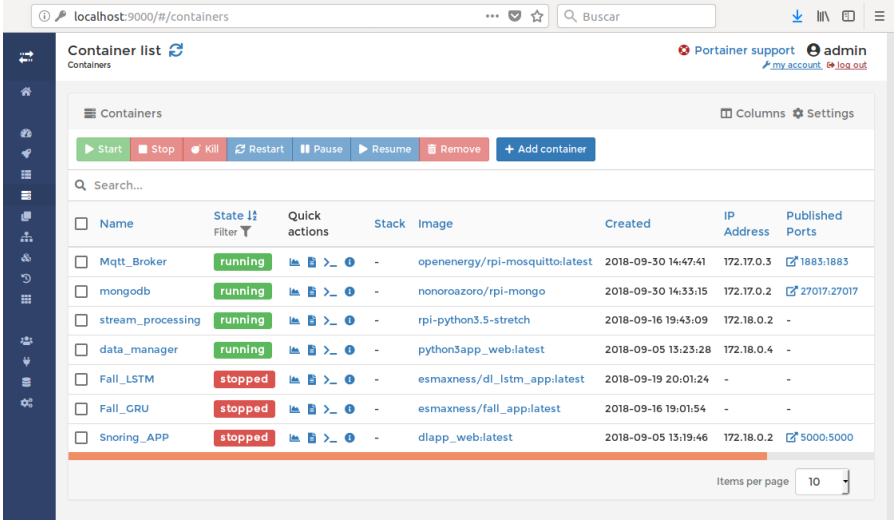


Figura 5.12: Gestión remota del nodo *fog* IoT empleando Portainer

5.4.6 Detección de caídas en tiempo real

La detección de caídas en tiempo real es realizada a través de una instancia del proceso de detección en tiempo real definida por la arquitectura en la subsección 3.6.3.

El proceso de detección de caídas en tiempo real es ejecutado en nodos *fog* y tiene como componente crítico el integrador de fuentes de IoT. El cual concentra las comunicaciones de los dispositivos IoT, de los módulos internos del nodo *fog* y el envío de notificaciones a los cuidadores y especialistas médicos. El bróker MQTT facilita estas interacciones a través del mecanismo publicación/suscripción. Los clientes MQTT se subscriben o publican los mensajes a través de tópicos. Estos tópicos siguen una estructura jerárquica y están separados por el limitador (/). En este caso, dos tópicos han sido creados: (i) Sensor/cinturon, el cual concentra los mensajes enviados por los dispositivos *wearables* IoT; y (ii) Prediccion/caida, el cual concentra los mensajes enviados por el módulo de análisis predictivo [189].

Adicionalmente, el protocolo MQTT brinda tres niveles de QoS para contrarrestar los problemas de pérdidas de mensajes que puedan aparecer durante la comunicación y pueden provocar ineficacia en el sistema. Los niveles QoS son acuerdos que garantizan la transmisión de mensajes entre el bróker y los clientes de publicación o suscripción. El nivel QoS 0 transmite los mensajes como máximo una vez, siguiendo un nivel de confiabilidad del mejor esfuerzo. En este nivel de servicio no existirán mensajes duplicados puesto que el mensaje no es almacenado ni re-enviado al destinatario. El nivel QoS 1 transmite los mensajes al menos una vez, es decir el bróker garantiza que el mensaje llegó al destinatario. En este nivel es posible que los mensajes se envíen más de una vez. Finalmente, el nivel QoS 2 transmite los mensajes exactamente una vez, es decir el bróker garantiza que el mensaje llegó al destinatario una sola vez. Si bien la confiabilidad de este nivel es alta, este afecta al tiempo de envío de mensajes puesto que requiere del intercambio de mensajes de confirmación de recepción a través de dos flujos de solicitud/respuesta [189].

El proceso de predicción de caídas en tiempo real sigue el esquema presentado en la Figura 5.13. A continuación se describe la interacción de los módulos en el proceso de predicción de caídas en tiempo real:

1. El dispositivo *wearable* IoT se comunica con el *gateway* IoT usando como tecnología inalámbrica WiFi y bajo el protocolo MQTT. Este dispositivo publica sus datos correspondientes a los valores del acelerómetro tri-axial (x, y, z) en el tópico Sensor/cinturon/id, donde id corresponde a un identificativo único de cada cinturón *wearable*. Es-

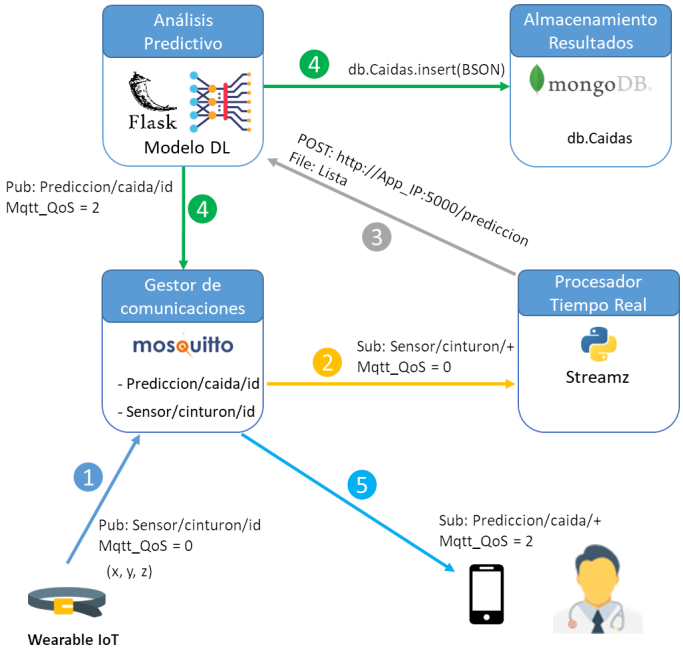


Figura 5.13: Esquema del proceso de detección de las caídas en tiempo real

te identificativo único permite gestionar adecuadamente el envío de mensajes en caso de que el nodo *fog* gestiona la comunicación de varios dispositivos *wearables* IoT. En este caso, el QoS configurado en esta comunicación es del tipo nivel 0 para evitar el envío de mensajes duplicados que puedan afectar el proceso de detección y para evitar la introducción de retardo.

2. El programa aplicación para el procesamiento en tiempo real se subscribe al tópico Sensor/cinturon/id del servidor MQTT. En caso de que el nodo *fog* gestione la comunicación de varios dispositivos *wearables* IoT, la suscripción puede realizarse empleando el comodín (+) al final del nivel cinturón (Sensor/cinturon/+). Los mensajes MQTT le llegan como un flujo constante de datos debido a que la actualización del tópico se realiza constantemente (cada 0.005 s). Al igual que el anterior caso, el QoS configurado es del tipo nivel 0 para evitar retransmisiones que puedan generar lentitud.

3. El programa aplicación para el procesamiento es implementado usando la librería Streamz de Python, la cual es una librería ligera que sigue el modelo de programación DAG para gestionar el flujo continuo de datos. La Figura 5.14 muestra el DAG usado para pre-procesar

los datos que serán utilizados para detectar las caídas. El DAG está compuesto por varios nodos que se encargan de realizar una tarea específica para tratar el flujo de datos de entrada. El primer nodo es la entrada del flujo de datos, es decir los mensajes MQTT. El segundo nodo utiliza el operador *map*, el cual aplica la función *split_value()* a cada elemento del flujo y devuelve un nuevo flujo con los elementos modificados. La función *split_value()* separa los elementos en valores separados por comas (x, y, z) y los transforma en tipo de dato punto flotante. El tercer nodo es el operador *sliding_window* el cual acumula los valores separados y se desliza en el tiempo. El tamaño de la ventana es de 2999, el cual corresponde a las dimensiones de entrada que requiere el modelo para detectar las caídas. El cuarto nodo es el operador *delay* el cual añade un retardo de un segundo. El quinto nodo utiliza el operador *map* y aplica la función *split_append()* a cada elemento del flujo. La función *split_append()* compila en una lista los elementos del flujo de datos. Finalmente, el operador *sink* realiza las solicitudes POST a la aplicación web. Las solicitudes POST adjunta como archivo la lista de valores generada por el quinto nodo.

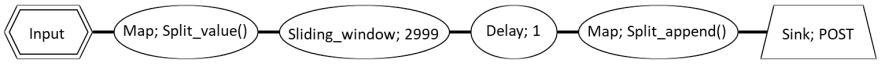


Figura 5.14: DAG programa aplicación procesamiento tiempo real

4. El programa aplicación para la detección de las caídas recibe las solicitudes POST a través de la exposición de una API REST. Como primer punto, el programa aplicación verificará que las dimensiones de la lista de valores enviadas como archivo adjunto a la solicitud POST sean del tamaño $2999x3$. Si las dimensiones son incorrectas, la lista de valores no son procesadas por el modelo de *deep learning*. Por el contrario, la lista de valores es enviada al modelo *deep learning* como valores de entrada para que emita una predicción. El modelo responde con un valor entre 0 y 1. El programa aplicación utiliza un filtro por el cual pasa el valor respuesta para determinar si existe o no una caída. Este filtro se encuentra configurado en 0,75; así si un valor respuesta es menor a 0,75 se determina como no caída, caso contrario se determina como caída. El valor del filtro está configurado de acuerdo a pruebas experimentales para obtener la máxima sensibilidad del modelo de predicción. En caso de que se detecte una caída, el programa aplicación envía un mensaje al bróker MQTT usando el tópico Prediccion/caida/ y con un tipo de QoS de nivel 2 debido a que se requiere confiabilidad en la notificación de la detección de caída a los servicios de emergencia. Además, el programa aplicación insertará en la base de datos *Caidas.db*, creada en MongoDB, un documento BSON. El

documento BSON contiene la fecha y hora de la caída detectada, el id del *wearable* IoT del cual fue detectado, y la ubicación del archivo con el que se predijo la caída. Este archivo es enviado al *cloud* posteriormente para ser analizado con el fin de ampliar el conjunto de datos de entrenamiento del modelo.

5. El teléfono móvil inteligente de los cuidadores de ancianos o de los profesionales de la salud recibe los mensajes MQTT a través de una aplicación cliente. La aplicación cliente se suscribe al tópico Prediccion/caída/ bajo un nivel de QoS de 2 para que se garantice la recepción de los mensajes en caso de caída. La Figura 5.15 muestra una notificación enviada al teléfono móvil mediante notificaciones MQTT.

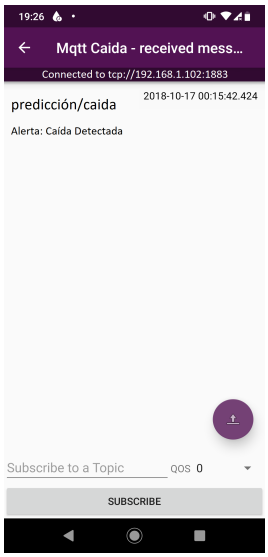


Figura 5.15: Envío de notificaciones caída a teléfono móvil

5.5 Evaluación del sistema

5.5.1 Escenario de pruebas

La evaluación en este caso de AAL tiene el objetivo de determinar las ventajas y desventajas del despliegue de los servicios de predicción en nodos *fog*. El escenario de pruebas fue planteado para analizar el rendimiento del modelo de detección de caídas, comparar con el servicio de detección basado en el *cloud* y verificar las limitaciones del servicio desplegado en el *fog*.

El análisis del rendimiento del modelo de detección de caídas fue realizado empleando el conjunto de datos SiSFall propuesto por Sucerquia et al. [179] y que sirvió para el entrenamiento del modelo. Los resultados son presentados en la subsección 5.5.2.

La comparación de la arquitectura propuesta y el enfoque basado en el *cloud* fue realizada a través de dos escenarios. El primer escenario planteó el despliegue del servicio de detección de caídas en una infraestructura *cloud computing* pública. En este caso, el servicio de infraestructura *cloud computing* provisto por AWS fue empleado para definir una máquina virtual donde se ejecute la aplicación Docker de detección de caídas. La máquina virtual fue generada empleando una instancia T2.micro (Variable ECUs, 1 vCPUs, 2,5 GHz, Intel Xeon Family, 1 GiB memoria), 8GB SDD, y sistema operativo Ubuntu Server 16.04 LTS (HVM). En este caso, el nodo *fog* recibía los datos del *wearable* IoT y los reenviaba al servidor en el *cloud* para determinar si existió una caída.

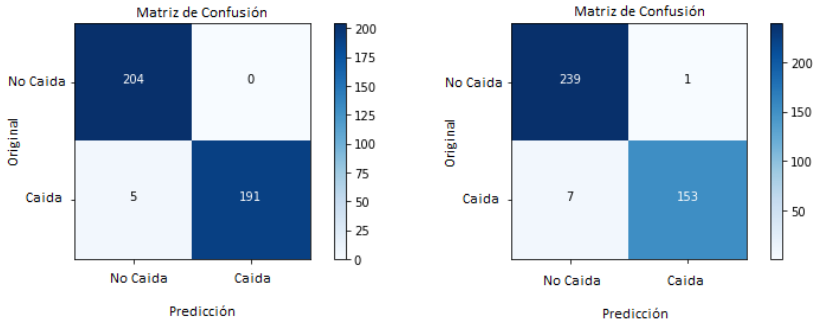
El segundo escenario planteó el despliegue del servicio de detección de caídas en los nodos *fog*. En este caso, el nodo *fog* recibía los datos del *wearable* IoT y detectaba el patrón de caídas según el modelo entrenado previamente en el *cloud*. Los resultados de la comparativa son presentados en la subsección 5.5.3.

Dado que para la realización de estas pruebas implicaba un simulacro de caídas de los adultos mayores llevando el prototipo *wearable* IoT y sus riesgos relacionados, el simulacro de las caídas fue realizado por tres voluntarios con edades entre los 30 y 33 años. Las caídas fueron simuladas en un ambiente seguro y controlado para que los voluntarios no sufran ningún tipo de percance durante la realización de las pruebas. Los voluntarios firmaron un acuerdo de privacidad para recolectar sus datos de forma anónima, así como para garantizar que los datos recolectados serán usados para experimentación.

La verificación de las limitaciones del servicio desplegado en nodos *fog* fue realizada mediante el incremento de los *wearables* IoT conectados al nodo *fog* solicitando el servicio de detección de caídas. Dado que la implementación de una gran cantidad de *wearables* IoT puede ser costosa para la evaluación, un programa computacional fue desarrollado para emular la cantidad de dispositivos necesarios para las pruebas. El programa simula *wearables* IoT enviando datos de flujo constante. La simulación es realizada empleando el conjunto de datos SiSFall propuesto por Sucerquia et al. [179]. Los resultados son presentados en la subsección 5.5.4.

5.5.2 Rendimiento del modelo de detección de las caídas

El rendimiento de los modelos de *machine learning* y *deep learning* es determinado basado en los cálculos de la precisión, sensibilidad y especificidad. El cálculo de estos parámetros está basado en la matriz de confusión, la cual permite definir la cantidad de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN). La matriz de confusión para el modelo LSTM es mostrada en la Figura 5.16a, mientras que la matriz de confusión para el modelo GRU es mostrada en la Figura 5.16b.



(a) LSTM **(b)** GRU

Figura 5.16: Matriz de confusión para los modelos RNN

El cálculo de los parámetros de rendimiento basado en la matriz de confusión sigue las ecuaciones 5.12, 5.13, 5.14. Los resultados de la evaluación del rendimiento de los conjuntos de datos son resumidos en la Tabla 5.2 y comparados con trabajos similares.

$$sensibilidad = \frac{TP}{TP + FN} \tag{5.12}$$

$$\text{precisión} = \frac{TP + TN}{TP + TN + TP + FN} \tag{5.13}$$

$$especificidad = \frac{TN}{TN + FP} \tag{5.14}$$

Los modelos de *deep learning* presentan mejor rendimiento en comparación con los modelos SVM, K-NN y árbol de decisión utilizados por sistemas similares donde se utiliza nodos *fog* para la detección de las caídas. En

Tabla 5.2: Comparación rendimiento con propuestas similares

Parámetros	**Nuestro Sistema**		**[171]**			**[170]**
	LSTM	**GRU**	**SVM**	**K-NN**	**SOM**	**Árbol Decisión**
Precisión [%]	**98,75**	98,40	95,6	70,6	98,7	91,7
Sensibilidad [%]	**97,60**	97,15	82,7	84,2	91,2	93,7
Especificidad [%]	**97,44**	95,63	95,0	41,0	98,1	-

comparación con el modelo SOM donde se alcanza una precisión similar, pero con una sensibilidad y especificidad superior y con mejor balance. Este balance entre los parametros de sensibilidad y especificidad garantiza un mejor funcionamiento del sistema en la detección de caídas. Una diferencia considerable entre estos parámetros, como la que se observa en el modelo SOM, no es lo deseable puesto que el modelo pasará por alto varios eventos verdaderos de caídas y en otros casos detectará eventos de caídas cuando no lo son. Por estas razones, el modelo LSTM propuesto en este trabajo presenta un balance en todos sus parámetros evaluados, a la vez que son los más altos alcanzados en sistemas de este tipo (basado en *fog computing*).

5.5.3 Prestaciones del servicio de predicción en nodos fog

El principal objetivo de la arquitectura planteada para el caso de salud es la mejora de los tiempos de detección. Estos tiempos son críticos en el caso de la salud. El tiempo es medido desde que llegan los mensajes al bróker MQTT hasta que la aplicación de predicción devuelve la respuesta indicando si existe o no una caída para los dos escenarios descritos en la sección 5.5.1. La Tabla 5.3 muestra en resumen el tiempo de inferencia alcanzado tanto para el primer escenario (predicciones en el *cloud*) como para el segundo escenario (predicciones en el *fog*). En ella se observa que los tiempos de inferencia son menores cuando las predicciones se realizan en dispositivos cercanos a la fuente de datos.

Tabla 5.3: Comparación entre los escenarios evaluados

Parámetro	**Gateway IoT**		**Cloud Computing**	
	LSTM	**GRU**	**LSTM**	**GRU**
Tiempo Inferencia [s]	1,21	1,14	2,655	2,389

La Tabla 5.4 muestra la comparativa de los tiempos de inferencia de nuestro sistema con propuestas similares de predicción en el *fog*. En ella se observa que el tiempo de inferencia es menor al sistema basado en *Smartphone*

para los modelos SVM y K-NN [171]. Excepto con el modelo SOM donde claramente se observa que el tiempo de detección es inferior al conseguido en este trabajo. Esta diferencia se debe principalmente a que el modelo SOM utiliza muchos menos parámetros que los usados por LSTM/GRU por consiguiente necesita menos recursos computacionales. Además, los recursos computaciones usados como nodos *fog* son considerablemente mejores los presentados en el nodo *fog* (teléfono móvil inteligente Huawei P8 Lite) donde se desplegó el modelo SOM en comparación con el nodo *fog* (Raspberry Pi 2) utilizado por nuestro sistema. A pesar de que el tiempo de detección de caídas utilizando el modelo SOM es menor al nuestro, el rendimiento del modelo SOM no es el adecuado, tal y como se evidenció en la Tabla 5.2.

Tabla 5.4: Comparación tiempos con propuestas similares

Parámetro	**Nuestro Sistema**		**[171]**		
	LSTM	**GRU**	**SVM**	**K-NN**	**SOM**
Tiempo Inferencia [s]	1,21	1,14	9	9	0,04

Asimismo, el consumo del CPU, memoria y energía eléctrica son analizados en el nodo *fog* y comparados en los dos escenarios descritos en la sección 5.5.1. Las mediciones realizadas siguiendo las metodologías propuestas en [190] [191] recolectaron 500 valores durante 3 horas como tiempo de observación para cada parámetro.

La Figura 5.17 muestra en resumen los valores promedio del porcentaje de utilización obtenidos durante la observación para el CPU y memoria del nodo *fog*. En ella se observa que el consumo de CPU se incrementa cuando la detección de las caídas es realizada en el *fog*. Además, el uso de memoria se incrementa en un 40 %. La Figura 5.18 muestra en resumen los valores promedio del consumo de electricidad del nodo *fog*. Finalmente, el consumo de electricidad muestra que el enfoque de detección en el *fog* es más eficiente. Los resultados de CPU y memoria eran los esperados puesto que el modelo de predicción demanda mayor consumo de recursos. Los resultados del consumo de electricidad en el escenario del *cloud* es mayor debido a la utilización de la interfaz de red *Ethernet* para el reenvío de los datos al *cloud*.

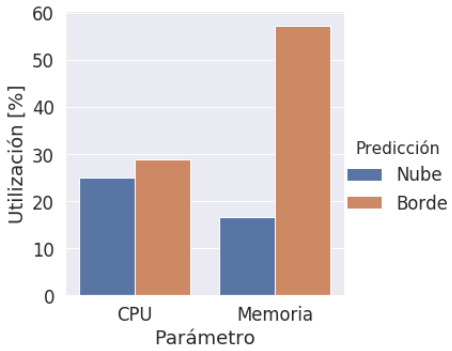


Figura 5.17: Comparativa de utilización de CPU y memoria del nodo *fog* para los escenarios de predicción en la nube y borde.

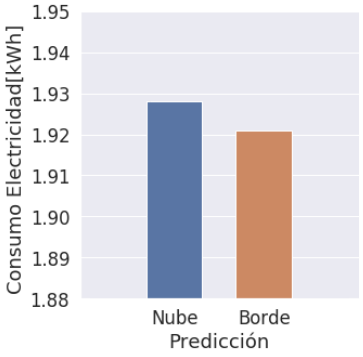


Figura 5.18: Comparativa del consumo de electricidad del nodo *fog* para los escenarios de predicción en la nube y borde.

5.5.4 Limitaciones de la predicción en nodos fog

Las pruebas se definieron simulando 5, 10, 20, 30, 50, 100, 150 *wearables* IoT conectados al nodo *fog* solicitando el servicio de detección de caídas. Los parámetros evaluados son el consumo de CPU, memoria, y electricidad y tiempo de inferencia. La metodología empleada para la recolección de datos de los parámetros CPU, memoria y consumo de electricidad fue la propuesta por [190] [191]. Las Figuras 5.19, 5.20, 5.21, 5.22 muestran el comportamiento de estos parámetros conforme se incrementa el número de dispositivos *wearables* IoT conectados al nodo *fog*.

En la Figura 5.19 se observa que el uso de CPU sube a partir de los 50 *wearables* pero se mantiene dentro de rangos aceptables (entre 35-45 % de

uso). Además, la tendencia del uso del CPU es creciente dependiendo al número de *wearables*, este comportamiento es el esperado.

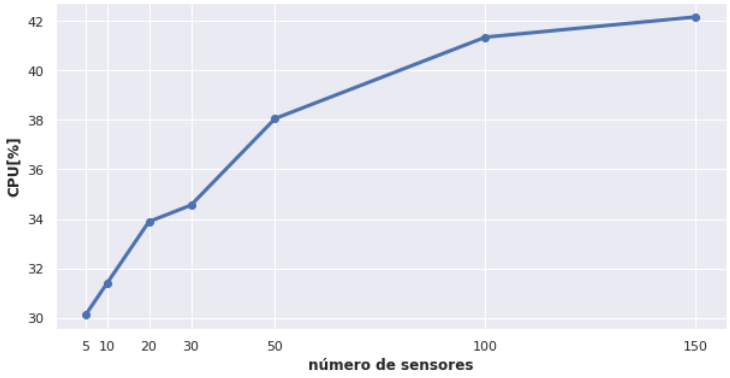


Figura 5.19: Uso de CPU

En la Figura 5.20 se observa que el consumo de electricidad se incrementa con el número de *wearables*. El consumo se encuentre entre rangos aceptables (1,620 - 1,655 Watts por hora). El uso de la interfaz WiFi y la interfaz de red de la Raspberry pi provocan un mayor consumo de electricidad. Si la utilización de la interfaz WiFi aumenta, el consumo de electricidad también lo hará. Sin embargo, a mayor número de *wearables* el consumo de electricidad no es exageradamente alto, por lo contrario se mantiene en rangos aceptables.

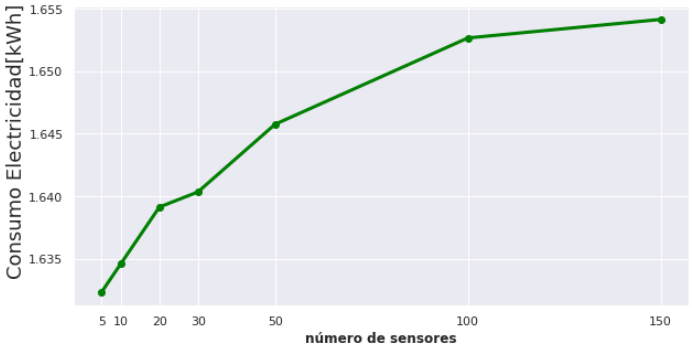


Figura 5.20: Consumo de Electricidad

En la Figura 5.21 se observa que el consumo de memoria es proporcional al número de *wearables* y este se incrementa de manera lineal. Sin embargo, el incremento es bajo y no llega a saturar la memoria a pesar de que se incrementa el número de usuarios.

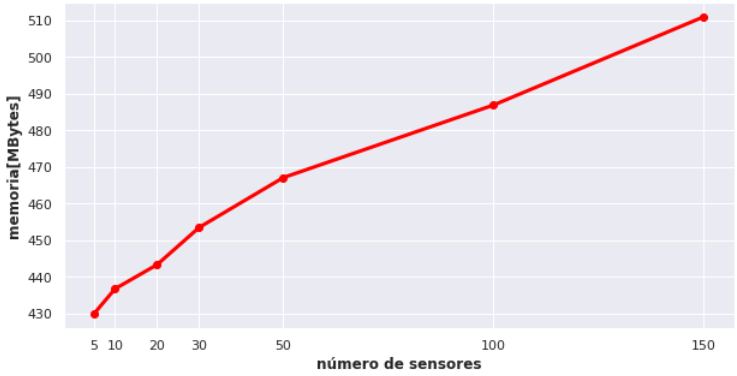


Figura 5.21: Uso de memoria

Finalmente, en la Figura 5.22 se observa que el tiempo de inferencia es el que más sufre las consecuencias del incremento en las conexiones del número de *wearables*. El crecimiento es exponencial, llegando a 200 segundos cuando se tienen 150 *wearables*. De esta manera, el nodo *fog* es capaz de soportar hasta 20 dispositivos *wearables* conectados solicitando simultáneamente el servicio de detección de caídas. Posterior a este número de dispositivos, el servicio se degrada y no es el deseado para este tipo de aplicaciones.

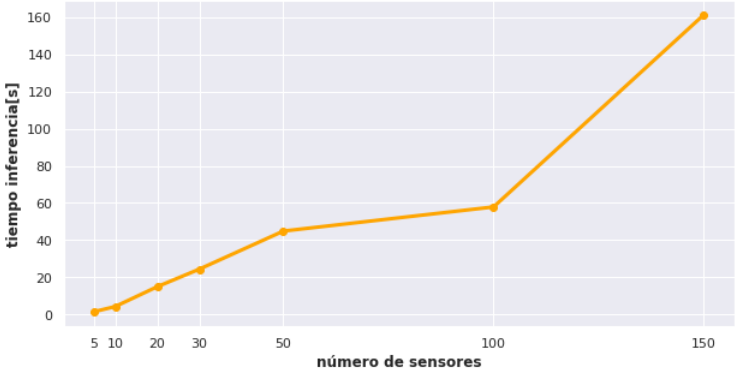


Figura 5.22: Tiempo de inferencia del modelo RNN en el borde

5.6 Conclusiones

El uso de habilitadores tecnológicos de IoT en el ecosistema de AAL permite habilitar un entorno inteligente con el fin de que los adultos mayores puedan vivir una vida sin depender de otras personas y de manera activa y saludable. Una de las aplicaciones de mayor estudio en el entorno AAL es la detección de caídas en adultos mayores. El principal objetivo de este tipo de aplicaciones es detectar las caídas y alertar a los servicios de emergencia para reducir los problemas relacionados, tales como fracturas, fisuras, pérdidas del conocimiento, e incluso la muerte. Los actuales enfoques utilizan *cloud computing* y algoritmos de *machine learning* para la detección de caídas. Sin embargo, este enfoque afronta las limitaciones del *cloud* y la baja precisión de los modelos de *machine learning*. Por ello, en este capítulo se ha presentado el uso de una instancia de la arquitectura de Big Data para IoT para mejorar las prestaciones de los sistemas de detección de caídas basados en IoT a través de la distribución de servicios de detección de caídas basados en algoritmos de *deep learning* en nodos de *fog computing*.

En este caso de uso se ha validado la implementación de la arquitectura siguiendo el patrón arquitectural basado en *fog-cloud computing*. La distribución de los componentes funcionales y procesos de la arquitectura a lo largo del *fog* demostró ser eficaz para soportar el despliegue de los modelos basados en *deep learning*. Como resultado, el patrón arquitectural permitió reducir el tiempo transcurrido desde la caída a la detección de la misma a 1.14 segundos, siendo este uno de los tiempos más bajo logrado en este tipo de sistemas y el más bajo alcanzado por sistemas basados en el despliegue de modelos de *deep learning* en nodos *fog*.

El despliegue de los modelos de *deep learning* cuenta con el apoyo de la virtualización ligera de contenedores como tecnología para *fog computing*. La virtualización ligera de contenedores en nodos *fog* demostró ser adecuada para manejar los limitados recursos computacionales. El uso de memoria, CPU y potencia eléctrica presenta muy poca variación y se mantiene en rangos óptimos, a pesar de que se incrementaron la cantidad de dispositivos IoT conectados al nodo *fog*.

Sin embargo, las pruebas realizadas para evaluar la escalabilidad arrojaron resultados que se pueden mejorar. El servicio de detección presenta una degradación en el tiempo de inferencia a medida que se incrementa el número de dispositivos de IoT. El sistema es capaz de soportar hasta 20 dispositivos IoT conectados simultáneamente alcanzando tiempos no superiores a los 20 segundos en la detección de la caída.

Capítulo 6

Sistema monitorización transporte marítimo usando un espacio de datos portuarios

6.1 Introducción

El transporte marítimo ha constituido el eje central para la comercialización de productos entre regiones desde hace muchos siglos. Hoy en día, el transporte marítimo sigue siendo el más importante medio de transporte soportando cerca del 80 % del comercio mundial [192]. Como resultado, los puertos marítimos se han convertido en los nodos intermodales con mayor relevancia en el transporte de mercancías a nivel mundial. En los próximos años, el esperado crecimiento en el tráfico marítimo afectará directamente a las operaciones portuarias debido principalmente a los insuficientes recursos portuarios para soportar este inminente crecimiento [193]. Por ello, los puertos marítimos necesitan estrategias que les permitan gestionar los recursos eficientemente.

Las nuevas tecnologías como IoT, Big Data, IA, *Cloud Computing*, entre otras, aparecen como las soluciones tecnológicas para cubrir las necesidades

de los puertos marítimos. La aplicación de estas tecnologías con el fin de mejorar la eficiencia engloba el concepto de la Industria 4.0. Este concepto orienta a la industria en el camino de una nueva revolución industrial donde la digitalización y automatización de sus procesos son la base fundamental para mejorar la eficiencia industrial [42]. La aplicación de estas tecnologías en las actividades portuarias es fundamental en la inminente transformación de un puerto marítimo a un puerto inteligente o puerto 4.0.

Los puertos inteligentes se sostienen sobre dos pilares fundamentales: la automatización de las operaciones y equipos portuarios, y la interconexión de los actores involucrados en la cadena de transporte y logística portuaria [194]. La arquitectura de referencia RAMI 4.0 está siendo utilizada como referencia para el diseño de sistemas cuyo objetivo es el de conectar los equipos portuarios al mundo virtual. Estos sistemas requieren de una interacción con tecnologías habilitadoras de *cloud computing* y Big Data para gestionar el gran volumen de datos generados en el entorno portuario con el fin de mejorar la eficiencia en las operaciones y añadir valor al sistema de transporte marítimo. Sin embargo, la integración de todos los actores involucrados en las operaciones portuarias y sus sistemas presenta una gran complejidad. Estos sistemas son aislados entre sí con formatos de datos e interfaces diferentes. Además, una de las mayores preocupaciones en la industria es la pérdida de control de los datos. En la actualidad, la industria considera a los datos como su bien más preciado por lo que se muestran reacios en el momento de compartir sus datos. Estos factores han afectado a la interacción de los sistemas basados en arquitecturas de referencia de IoT para la industria (RAMI 4.0 e IIoT) con tecnologías habilitadoras de Big Data.

Una de las soluciones para encarar los problemas en la integración de los diversos actores involucrados en procesos industriales son los espacios de datos IDS. La arquitectura de referencia IDS tiene como objetivo habilitar un ambiente seguro e interoperable para compartir datos. IDS garantiza el buen uso de los datos controlando en todo momento su uso a través de políticas. IDS es una potencial fuente de Big Data debido a que concentra los datos generados por varias aplicaciones IoT industriales y otras fuentes relacionadas con la gestión y planificación de procesos industriales. En este caso de uso de la arquitectura propuesta en el capítulo 3 analizamos a IDS como fuente de Big Data generado por aplicaciones IoT en la industria y la explotación de estos datos para la generación de KPI relevantes en la gestión y planificación de las operaciones marítimas. La explotación de los datos compartidos en IDS permitió el desarrollo de un sistema de monitorización de las operaciones de la flota de buques de una naviera.

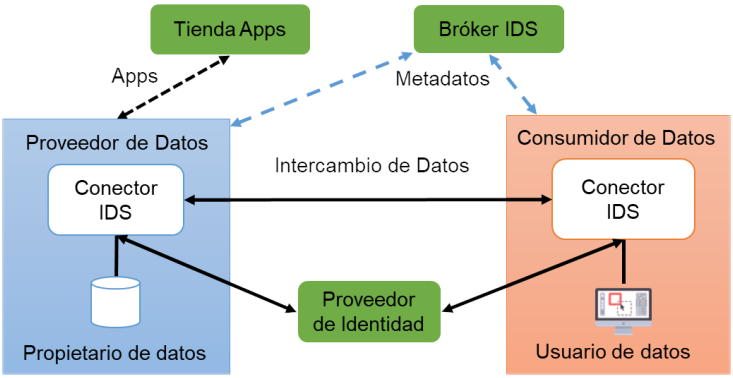
En la primera sección se introduce a la arquitectura IDS y su elemento principal, el conector IDS. En la segunda sección se plantea la motivación y trabajos relacionados. En la tercera sección se describe la relación de la arquitectura del sistema con la arquitectura global. En la cuarta sección se describe el diseño del Espacio de Datos Portuario (EDP) como fuente de Big Data. En la quinta sección se presenta el sistema de monitorización de la flota de buques de una naviera, basada en la instancia de la arquitectura de Big Data. Finalmente, la sexta sección presenta los resultados de la evaluación experimental del caso de uso.

6.2 Arquitectura IDS

IDS es una arquitectura de referencia propuesta por la *International Data Spaces Association* (IDSA) para facilitar la compartición de datos en un entorno de confianza [195]. Esta arquitectura tiene el objetivo de cubrir los requerimientos industriales de confianza, seguridad, interoperabilidad, valor añadido, y mercado de datos. La arquitectura basa su funcionamiento en la idea de la soberanía de los datos. El concepto de soberanía de datos hace referencia al control de los datos por parte de los propietarios, es decir que los propietarios de los datos conocen en todo momento como se están usando sus datos y controlan que parte de los datos quieren que se comparta.

La arquitectura de referencia provee información de los roles, entidades, funcionamiento y consideraciones para la implementación de la misma. Mientras que, el desarrollo de interfaces y la implementación de la arquitectura no es detallado por la arquitectura. Por ello, los esfuerzos de la comunidad científica y de la industria están enfocados en proveer detalles sobre la implementación y validación de la arquitectura a través de casos de uso [195].

La arquitectura IDS está compuesta por diversas entidades que presentan un rol específico para habilitar el espacio de datos. Los roles principales en el intercambio de datos son el propietario de datos, el proveedor de datos, el consumidor de datos y el usuario de datos. Además a esto roles, los roles intermedios son considerados para el despliegue de tareas administrativas y de seguridad, y estos son el proveedor de identidad, tienda de aplicaciones, tienda de vocabulario, cámara de mediación y bróker proveedor de servicios [196]. La Figura 6.1 muestra el modelo operacional de la arquitectura IDS.



Fuente: Otto, B. et al., 2018 [196]

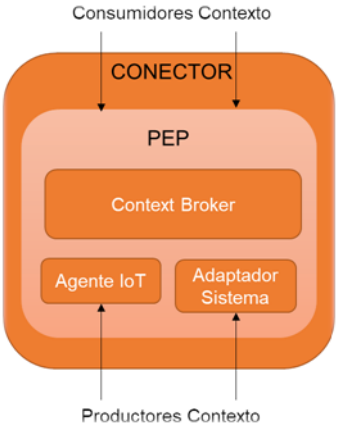
Figura 6.1: Modelo operacional de la arquitectura IDS

6.2.1 Conector y proveedor de identidad IDS

El conector IDS basado en la plataforma IoT FIWARE propuesto por Alonso, A. et al. [197] ha sido implementado en varios casos de uso de la arquitectura IDS en entornos industriales. El conector IDS tiene como principal componente al GE Orion Context Broker. Además, el GE Wilma es empleado para proveer las funcionalidades de *Policy Enforcement Point* (PEP)-proxy necesarias para asegurar las políticas de uso y su aplicación.

El componente proveedor de identidad de la arquitectura IDS es implementado con el GE IdM-Keyrock. Este habilitador gestiona la información de identidad de cada conector IDS. El IdM Keyrock emplea el protocolo Oauth2 para la autenticación de los conectores IDS y mantener registro de las operaciones de cada conector (*logs*).

Además, un servidor AuthZForce GE *Policy Administration Point*/*Policy Decision Point* (PAP/PDP) provisto por FIWARE es empleado para implementar las funcionalidades para mantener y controlar las políticas de uso de datos del IDS bróker. El AuthZForce define las políticas de control de acceso a aplicaciones y servicios mediante el uso del lenguaje *eXtensible Access Control Markup Language* (XACML). Las políticas definidas con este lenguaje permiten controlar las autorizaciones de petición y respuesta durante el intercambio entre conectores IDS. La Figura 6.2 muestra la arquitectura de un conector IDS basado en el GE Orion Context Broker de la plataforma FIWARE.



Fuente: Alonso, A. et al., 2018 [197]

Figura 6.2: Conector IDS basado en la plataforma FIWARE IoT

6.3 Motivación y trabajos relacionados

Los sistemas de transporte marítimo son fuentes inagotables de grandes volúmenes de datos [198]. Uno de estos ejemplos es el sistema AIS utilizado para el seguimiento de embarcaciones marítimas. AIS tiene el objetivo de habilitar la comunicación entre todo tipo de embarcaciones marítimas con los puertos marítimos. El sistema AIS utiliza un canal de radio frecuencia *Very High Frequency* (VHF) para el intercambio de mensajes. Estos mensajes contienen información acerca del *Maritime Mobile Service Identity* (MMSI), modo de operación, posición (latitud, longitud), *Course Over Ground* (COG), *Speed Over Ground* (SOG), *Rate of Turn* (ROT), entre otros. Los sistemas de transporte marítimo actuales emplean esta información con el fin de habilitar aplicaciones para predecir trayectorias y evitar posibles colisiones entre buques, detectar barcos pesqueros en zonas restringidas, la seguridad marítima, y la búsqueda y rescate [192].

Las aplicaciones basadas en AIS utilizan modelos para determinar las condiciones de las operaciones de los buques. Existen modelos para la estimación del consumo de combustible [192], contaminación ambiental [199] [200], detección de anomalías [201], predicciones de rutas [202], entre otros. Sin embargo, el problema con estos sistemas aparece cuando no se puede gestionar adecuadamente el gran volumen generado por el sistema AIS. Dado que los mensajes AIS son intercambiados en intervalos cortos (cada 2 - 10 segundos, o cada 3 minutos), el repositorio de mensajes AIS puede alcanzar los TB de información en un año [203]. El procesamiento de este gran volumen de datos resulta poco práctico con herramientas tradicionales y consume mucho

tiempo. Como resultado, los modelos no son explotados eficientemente y deben ser adaptados para que sean usados en la extracción de información empleando tecnologías de Big Data [204].

El proyecto Big Data Ocean ha sido creado con el objetivo de superar las limitaciones en la gestión del Big Data generado por el sistema AIS [205]. La iniciativa propone un ecosistema de Big Data para la solución de los problemas relacionados con el procesamiento en tiempo real y *batch* de los datos, gestión de los datos del tipo semi-estructurado y no estructurado; así como, del volumen de los datos generados por sistemas del entorno marítimo [206].

De la misma forma, la iniciativa datAcron ha propuesto una arquitectura de Big Data para entornos marítimos [207] [208], basada en la arquitectura Lambda. El uso de la arquitectura Lambda evidenció un efectivo manejo de las características del Big Data presentes en sistemas basados en AIS. La explotación de los datos en estos sistemas permitió generar servicios de detección de trayectorias y sobre todo proporcionar conciencia de la situación a través de la visualización de la ubicación de los buques.

Por otro lado, las fuentes de Big Data en las actividades portuarias no solo provienen del sistema AIS, sino también de sistemas transaccionales, sistemas IoT de monitorización de mareas y medio ambiente, sistemas IoT para el control de grúas de carga y descarga de contenedores, entre otros [209]. Estas fuentes de Big Data no han sido consideradas principalmente porque provienen de sistemas propios de las terminales portuarias y de la autoridad portuaria. De esta manera nos encontramos con un aislamiento entre sistemas y aplicaciones. Por un lado, los sistemas de transporte destinados al seguimiento de los buques, camiones, trenes y por otro los sistemas portuarios destinados al control de las operaciones de la infraestructura portuaria.

La comunicación entre estos sistemas es una tarea retadora debido a factores como la interoperabilidad entre sistemas y la falta de compromiso entre actores para compartir la información. El uso de sistemas basados en *Electronic Data Interchange* (EDI) han facilitado el intercambio de los datos bajo un mismo formato superando en gran parte la interoperabilidad entre sistemas [210]. Otra de las formas de compartir información portuaria es a través de los sistemas *Port Community Systems* (PCS). Los PCS han mejorado la comunicación entre actores portuarios al centralizar la información de los buques y los productos que transporta, a la misma veces que redujeron la complejidad de la interconexión entre actores portuarios [211]. Similarmente, la iniciativa *Port Collaboration for Decision Making* (PortCDM) propuesta por el *Sea Traffic Management* (STM) permitió la gestión inte-

ligente del tráfico marítimo [212]. El sistema PortCDM provee información acerca del retardo de los buques de carga, y procesos de carga y descarga de las terminales de transporte para facilitar la planificación de las operaciones portuarias. Sin embargo, la comunicación entre partes interesadas en las operaciones y actividades portuarias siguiendo estos métodos sigue mostrando limitaciones debido a que los actores portuarios no intercambian la información a tiempo, con precisión y eficiencia, y carecen de mecanismos de seguridad y privacidad [213].

Las implementaciones realizadas de la arquitectura de IDS han resuelto este tipo de limitación en la compartición de datos en áreas como la manufactura, energía renovable, transporte, entre otros [195]. En cuanto al transporte, la arquitectura IDS ha sido implementada en sistemas de camiones de carga con el objetivo de optimizar los tiempos de carga y descarga [214]. Asimismo, la arquitectura ha sido implementada para habilitar un ambiente seguro para compartir datos de las vías de trenes de carga con el objetivo de generar un mantenimiento preventivo de las vías [215]. En el caso del transporte marítimo y portuario no ha sido implementada la arquitectura IDS. Sin embargo, el instituto oceánico SINTEF está estudiando las implicaciones del uso de IDS en los entornos marítimos. Las recientes conclusiones de sus estudios resaltan la importancia de solucionar los problemas relacionados con la integración entre la información de los buques basado en AIS con IDS, y la complejidad y naturaleza del sistema de transporte marítimo internacional [216]. Como resultado, IDS aparece como el mecanismo para solucionar los problemas de comunicación entre los sistemas portuarios y facilitar la liberación del Big Data marítimo para que pueda ser explotado con el fin de generar servicios que ayuden a mejorar los procesos portuarios.

El trabajo propuesto en este capítulo utiliza la arquitectura IDS para habilitar un entorno de compartición del Big Data generado por aplicaciones marítimas. Además, una instancia de la arquitectura de Big Data para IoT propuesta en el capítulo 3 es utilizada para explotar el Big Data compartido en IDS y generar un sistema de monitorización de la flota de buques de una naviera con el fin de optimizar las operaciones de transporte. Por otro lado, el capítulo adapta algunos modelos asociados con la monitorización basada en AIS para ser utilizados aprovechando las metodologías y técnicas de Big Data.

6.4 Relación de la arquitectura del sistema con la arquitectura General

El principal objetivo de este capítulo es demostrar la utilidad de la arquitectura general presentada en el capítulo 3 por medio de la definición de una instancia de la arquitectura para el desarrollo de un sistema de monitorización de la flota de buques de una naviera usando a un EDP como fuente de Big Data en la industria marítima. En este caso de uso se verifica a los almacenes de datos de IoT (IDS) como fuente de datos y el uso del patrón de implementación de la arquitectura basado en cloud-cloud computing. Este patrón es útil en este caso principalmente porque las aplicaciones propias de cada parte interesada están alojadas en sus respectivas infraestructuras privadas o públicas de cloud computing.

Los requerimientos funcionales del sistema en este caso de uso y su relación con los requerimientos generales definidos en la sección 3.2 son estructurados en la Tabla 6.1. En este caso de uso no se han considerado los requerimientos globales relacionados con la distribución de servicios de predicción (R7), gestión de los sistemas IoT (R9), y soporte de múltiples tecnologías computacionales (R10) debido a que no son parte de las necesidades del sistema propuesto.

Tabla 6.1: Requerimientos del sistema de monitorización de la flota de buques de una naviera

Requerimiento	**Relación Requerimientos Generales**
El sistema debe ser capaz de soportar la conexión con el EDP para la carga de datos de varios actores involucrados en las actividades portuarias, tales como el sistema de operaciones de la terminal portuaria y el sistema de transporte marítimo basado en AIS.	**R1.** Soporte de conexión con las múltiples fuentes de IoT.
El sistema debe ser capaz de gestionar y normalizar el formato CSV y NMEA utilizado por el sistema de operaciones de la terminal portuaria, y el sistema de transporte basado en AIS, respectivamente.	**R2.** Soporte para la conversión de formatos de datos de IoT.
El sistema debe ser capaz de procesar en tiempo real los datos generados por AIS, considerando a los buques como dispositivos IoT.	**R3.** Procesamiento de datos de IoT en tiempo real.

El sistema debe ser capaz de soportar el procesamiento de datos tipo *batch* y en tiempo real. El procesamiento *batch* proporciona información explotando los datos históricos tanto de las operaciones del terminal de contenedores como del sistema de transporte basado en AIS.	**R4.** Procesamiento de datos de IoT tipo *batch*.
El sistema debe ser capaz de soportar el almacenamiento de datos enviados en tiempo real a través del EDP y de las analíticas resultantes del procesamiento de datos.	**R5.** Soporte de almacenamiento.
El sistema debe ser capaz de realizar un análisis descriptivo de los datos con el fin de extraer KPI que proporcionen información útil para la planificación de las operaciones de la naviera.	**R6.** Análisis de datos.
El sistema debe ser capaz de resumir los resultados de los KPI obtenidos con la extracción de información con el fin de proveer servicios de visualización para mejorar la toma de decisiones.	**R8.** Visualización de resultados de los análisis de datos.
El sistema debe ser capaz de proteger los datos compartidos por los miembros del EDP, así como de los servicios que proporciona el sistema.	**R11.** Seguridad.
El sistema debe ser capaz de garantizar la privacidad de los datos a través de políticas y roles. De esta manera, el dueño de los datos controla en todo momento el buen uso de sus datos.	**R12.** Soberanía de datos.

Desde el punto de vista conceptual de la arquitectura, el EDP basado en la arquitectura IDS cumple el rol de proveedor de datos de IoT. Mientras tanto que el rol de proveedor de tecnologías habilitadoras de Big Data y de proveedor de servicios es asumido por la instancia de la arquitectura de Big Data para IoT. El rol de consumidor de servicios en este sistema de monitorización de la flota de buques de la naviera es asumido por los planificadores y gestores de las operaciones de la naviera, quienes guiados por la información provista por el sistema tomarán las mejores decisiones.

6.5 Espacio de datos portuarios

El EDP tiene el objetivo de habilitar un entorno seguro para que los actores involucrados en los procesos portuarios compartan los datos de sus sistemas. El EDP ha sido diseñado siguiendo la arquitectura IDS. En tal virtud, los actores portuarios requieren implementar un conector IDS para formar parte del EDP. En este caso de uso, se ha implementado un EDP para el puerto de Valencia. La Figura 6.3 muestra una visión general del EDP.

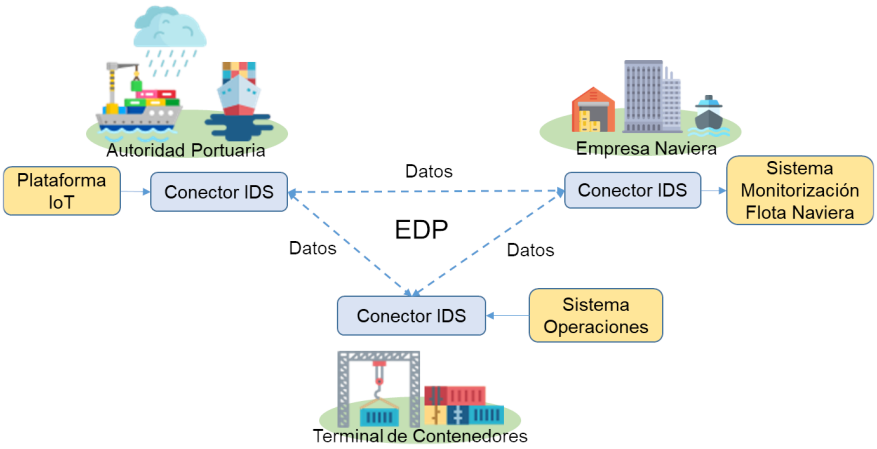


Figura 6.3: Visión general de EDP

A continuación se describen los actores que participan en el intercambio de datos en el EDP y los datos que compartirán.

- La autoridad portuaria del puerto de Valencia realiza el control de la navegación y monitorización de las condiciones meteorológicas y marítimas en la zona cercana al puerto. La autoridad portuaria comparte los datos del sistema AIS (control de navegación). Esta información es relevante para que otros agentes portuarios puedan planificar sus operaciones.
- La terminal de contenedores del puerto gestiona la carga y descarga de contenedores de los buques que llegan a la terminal. El terminal de contenedores comparte los datos de su sistema de operaciones basado en SQL. Este sistema posee información relacionada con el tiempo de las operaciones por cada buque (tiempo de carga y descarga). Esta información es de vital importancia para que las navieras puedan estimar el tiempo que sus buques tardan en moverse de un puerto a otro.

- La naviera implementa una instancia de la arquitectura de Big Data para IoT con el fin de gestionar el gran volumen de datos compartidos por parte de la autoridad portuaria y la terminal de operaciones. La arquitectura de Big Data es empleada para extraer información útil en forma de KPI. Los resultados del análisis de datos compartidos son presentados en un *dashboard* para visualizar la situación actual portuaria referente a su flota de buques y mejorar la toma de decisiones.

6.5.1 Implementación del espacio de datos portuario

A continuación se describen los detalles para la conexión de los sistemas de cada actor involucrado en este caso de uso con el conector IDS, así como la descripción del intercambio de datos en IDS.

Conexión del sistema AIS con el conector IDS

Los buques de contenedores pueden ser considerados como grandes objetos que comparten información de navegación a través del sistema AIS. Esta abstracción permite que los buques sean modelados dentro de la plataforma IoT FIWARE como dispositivos IoT. El modelo de datos empleado presenta la entidad *buqueObservado*. La Figura 6.4 muestra un ejemplo del modelo de datos de la entidad *buqueObservado* empleado en la plataforma IoT FIWARE en formato JSON-ld normalizado.

```
1  {
2   "id":"urn:ngsi-ld:Buque:246252000",
3   "type": “buqueObservado",
4   "cog": 14.4,
5   "sog": 2.4,
6   "rot": 2.1,
7   “modoOperacion": "anchorage",
8   “fechaObservacion":"2018-11-24-T08:24:04.00Z",
9   "location":{
10      "type": "Point",
11      "coordinates": [-4.75434343,41.64580099232]
12    }
13 }
```

Figura 6.4: Ejemplo JSON-ld normalizado del modelo de datos de la entidad *buqueObservado*

La entidad *buqueObservado* representa a un buque de carga con sus características individuales. Entre las características individuales tenemos el MMSI

el cual es convertido a identificativo URI siguiendo las recomendaciones de la plataforma FIWARE (línea código 2), la latitud y longitud representado por un punto bajo la característica *location* (línea código 9-12), el COG, el SOG, el ROT, modo de operación, y la fecha de observación (línea código 4-8). La entidad *buqueObservado* se actualiza con cada mensaje AIS enviado por los buques observados en la zona cercana al puerto de Valencia.

Dado que los mensajes del sistema AIS son intercambiados usando frases AIVDM/AIVDO bajo el estándar NMEA 0183, estos mensajes deben ser decodificados al formato JSON-ld utilizado por la plataforma IoT FIWARE antes de actualizar la entidad *buqueObservado*. La decodificación y la actualización de la entidad *buqueObservado* en la plataforma IoT FIWARE son realizadas a través de un flujo configurado en la plataforma Node-RED. La plataforma Node-RED permite la interconexión de dispositivos *hardware*, API y servicios online a través de la programación visual de flujos [217]. El flujo de datos está compuesto por varios nodos capaces de realizar transformaciones sobre los datos. La Figura 6.5 muestra el flujo de datos empleado para interconectar el sistema AIS con el conector IDS basado en la plataforma IoT FIWARE.

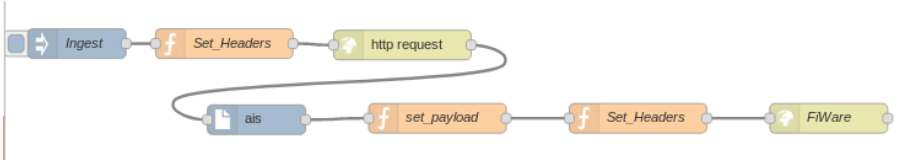


Figura 6.5: Flujo Node-RED para interconexión de AIS y plataforma IoT FIWARE

El flujo consulta el servidor HTTP del sistema AIS cada 10 segundos con un nodo *http_request*, añadiendo previamente las cabeceras HTTP respectivas. Cada mensaje NMEA del sistema AIS que haya llegado al sistema AIS en los últimos 10 segundos es decodificado y convertido a formato JSON a través del nodo AIS. Posteriormente, los mensajes en formato JSON son estructurados de acuerdo al modelo de datos de la entidad *buqueObservado* mediante el uso del nodo (*set_payload*). El nodo (*set_headers*) agrega las cabeceras necesarias según la interfaz NGSI de FIWARE. Finalmente, el nodo *FIWare* envía la solicitud POST al GE Orion Context Broker utilizado como conector IDS.

Conexión del Sistema de Operaciones con el conector IDS

El terminal de contenedores está compuesto por varias grúas que se encargan de la carga y descarga de contenedores de su correspondiente muelle. El sistema de operaciones del terminal de contenedores almacena en su sistema la información sobre la ocupación de cada muelle. En este caso, el modelo de datos empleado presenta la entidad muelle con sus características individuales (número de muelle, fecha y hora de inicio de operaciones y la fecha y hora de fin de operaciones). La Figura 6.6 muestra un ejemplo del modelo de datos JSON-ld en formato *keyValue* de la entidad muelle empleado en la plataforma IoT FIWARE.

```
1 {
2       "id":"urn:ngsi-ld:muelle:2",
3       "type": "muelle",
4       "inicioOperacion": "2018-02-24T22:45:09.00Z",
5       "finOperacion":  "2018-02-25T06:49:35.00Z",
6 }
```

Figura 6.6: Ejemplo JSON-ld normalizado del modelo de datos de la entidad *muelle*

Al igual que el caso anterior, la integración entre el sistema de operaciones con el conector IDS se realiza empleando Node-RED. La Figura 6.7 muestra el flujo de interconexión entre el sistema de operaciones y la plataforma IoT FIWARE.

Figura 6.7: Flujo Node-RED para interconexión sistema operaciones y plataforma IoT FIWARE

El flujo consulta la base de datos SQL del sistema de la terminal de operaciones cada 10 minutos. Estos 10 minutos son establecidos debido a que los datos generados por el sistema no tienen una alta frecuencia. La consulta filtra los datos para que se seleccionen las últimas actualizaciones del sistema. Los valores resultantes de la consulta son estructurados en formato JSON de acuerdo al modelo de datos utilizando el nodo (*set_payload*). Posteriormente, un nodo (*Set_Headers*) agregar las cabeceras correspondientes según la interfaz NGSI de FIWARE. Finalmente, el nodo *FIWare* envía la solicitud POST al GE Orion Context Broker utilizado como conector IDS.

6.5.2 Intercambio de datos entre conectores IDS

El intercambio de datos entre conectores IDS se produce empleando el mecanismo publicación/suscripción y las interfaces NGSI9 (disponibilidad de contexto) y NGSI10 (gestión contexto) provistas por FIWARE. Para ello, la funcionalidad de federación de Orion Context Broker es habilitada bajo el modo *push*. De esta manera, el Orion Context Broker enviará notificaciones de contexto de una entidad a otro Orion Context Broker, como muestra la Figura 6.8. El IdM (Keyrock) y el PAP/PDP AuthZForce para los dos conectores IDS son los mismos y son implementados por el proveedor de identidad del EDP, pero por razones didácticas se muestran en la Figura 6.8 como si fueran independientes para cada conector IDS.

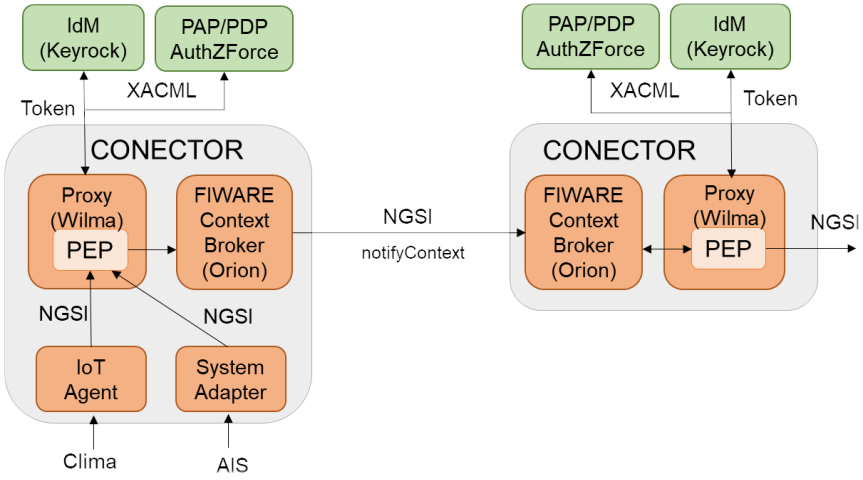


Figura 6.8: Intercambio de datos entre conectores IDS

El flujo de mensajes entre los componentes de los conectores IDS y del proveedor de identidad, al igual que el intercambio de notificaciones de contexto se puede visualizar en la Figura 6.9. El primer paso dentro del flujo es la autenticación de los conectores IDS. El conector IDS de la empresa naviera necesita ser autenticado como medida de seguridad para pasar a formar parte del EDP. Este proceso de autenticación garantiza que el conector IDS de la naviera es quién dice ser. Para ello, el conector envía un mensaje *AuthenticationRequest()* al IdM KeyRock solicitando la autenticación. El IdM KeyRock responderá utilizando Oauth 2.0 con un *Token* que será utilizado para sus comunicaciones. Posteriormente, el conector IDS de la naviera envía un mensaje *ContextRequest(Token)* solicitando información de contexto de una determina entidad perteneciente al IDS de la autoridad portuaria y cuyo mensaje tiene como adjunto el *Token* de autenticación. El proxy PEP gestiona esta solicitud, pero primero realiza una verificación

con el IdM para comprobar si el conector IDS que le realiza la solicitud es quien dice ser a través del envío de un mensaje con el *Token* enviado por conector IDS de la naviera. El IdM validará la identidad del conector IDS de la naviera y adjunta la información del conector IDS de la naviera y sus roles. Luego, el proxy PEP solicitará la revisión de las políticas asignadas a ese rol a través del envío de un mensaje en formato XACML al PDP. El PDP consultará las políticas asignadas al conector IDS de la naviera almacenadas en la base de datos de políticas. El PDP enviará la respuesta al PEP de la decisión referente a si ese conector IDS puede acceder a información de contexto de la entidad solicitada. Si la decisión del PDP es afirmativa, el PEP responderá con la información de contexto solicitada.

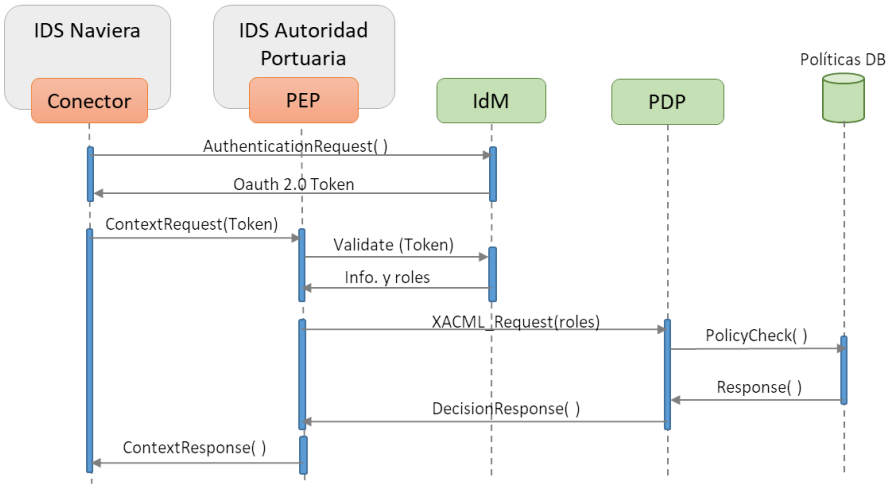


Figura 6.9: Flujo de mensajes entre los conectores IDS

6.6 Sistema de monitorización de la flota de buques de una naviera

El sistema de monitorización de la flota de buques de una naviera toma ventaja del Big Data compartido por los actores que participan en el EDP para generar servicios de monitorización y control de las operaciones de transporte. En esta sección se describe la instancia de la arquitectura de Big Data para IoT del sistema de monitorización de la naviera, la implementación de los procesos de recolección de datos, de extracción de información y de visualización de analíticas y KPI.

6.6.1 Instancia arquitectura de Big Data

La arquitectura del sistema de monitorización de la naviera utiliza una instancia de la arquitectura de Big Data para IoT propuesta en el capítulo 3 para cubrir las necesidades de gestión del Big Data compartido en el EDP. La Figura 6.10 muestra una representación de la distribución de las plataformas y tecnologías utilizadas para cubrir las funcionalidades de la arquitectura en los diferentes dominios funcionales y en concordancia con el patrón arquitectural de la vista funcional de la arquitectura (sección 3.5).

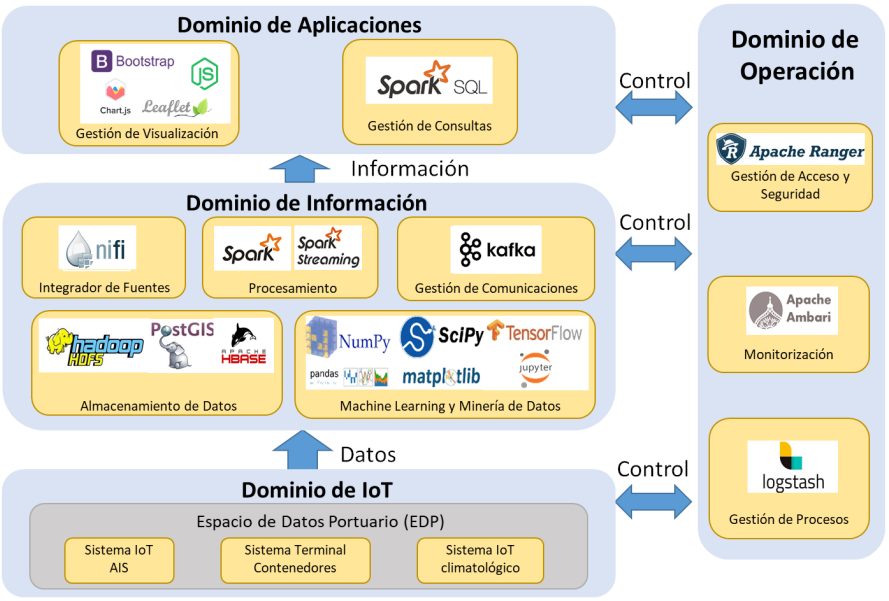


Figura 6.10: Instancia arquitectura de Big Data para IoT

Dominio IoT

El dominio IoT está compuesto por los sistemas correspondientes a la autoridad portuaria y la terminal de contenedores que se integran entre sí utilizando el EDP.

En este caso de uso son considerados como fuentes de Big Data al sistema AIS y el sistema de monitorización de las condiciones meteorológicas y marítimas provistas por la autoridad portuaria, y el sistema de operaciones del terminal de contenedores. Sin embargo, el EDP está en capacidad de permitir la integración de otros sistemas de diferentes actores portuarios. Como se describió en la sección 6.5, los sistemas considerados en este caso de uso son agregados al EDP utilizando los conectores IDS. Cada objeto de

los diferentes sistemas fue modelado para que se integren al conector IDS basado en el GE Orion Context Broker de la plataforma FIWARE IoT. Estos sistemas comparten grandes volúmenes de datos en tiempo real que requieren ser gestionados mediante la explotación de las funcionalidades de los dominios de información, de aplicaciones y de control de la arquitectura de Big Data para IoT.

Dominio información

- **Integrador de Fuentes:** las funcionalidades de conexión y conversión de formato de este componente funcional son implementados utilizando Apache NiFi. Esta herramienta permite extraer los datos de la plataforma FIWARE IoT y almacenarlos en el módulo de almacenamiento, como ya se observó en el capítulo 4.
- **Almacenamiento:** las funcionalidades de almacenamiento son divididas en tres tipos: repositorio histórico de datos, repositorio de datos procesados y repositorio de resultados. El repositorio histórico de datos es implementado usando un clúster HDFS. El clúster HDFS es creado usando tres máquinas virtuales. El clúster HDFS almacenará bloques de datos de 64 MB, el cual es el valor por defecto en HDFS. El repositorio de datos procesados es implementado usando una instancia de la base de datos NoSQL Apache HBase. Como la mayoría de base de datos NoSQL, esta provee escalabilidad y alta disponibilidad y sobre todo es compatible con el clúster HDFS. El repositorio de resultados es implementado usando una instancia de la base de datos PostGIS. El sistema SQL Postgres y su extensión PostGIS permiten almacenar los datos del tipo geoespacial y han sido ampliamente empleados en los sistemas tradicionales. La elección de PostGIS responde a la baja frecuencia de generación de los datos y la baja cantidad de datos que deberá almacenar producto del procesamiento del Big Data previo al almacenamiento.
- **Procesamiento:** las funcionalidades para el soporte del procesamiento tipo *batch* y en tiempo real son implementadas usando Apache Spark. Este *framework* de Big Data permite ejecutar aplicaciones que requieren procesamiento *batch* y en tiempo real. Apache Spark puede ser utilizado conjuntamente con clúster HDFS por medio del uso del gestor de recursos YARN.
- **Comunicación:** las funcionalidades de comunicación entre plataformas de Big Data son implementadas usando Apache Kafka. Apache Kafka garantiza escalabilidad, tolerancia a fallos y robustez.

- **Machine Learning y Minería de Datos:** las funcionalidades para la implementación de algoritmos de *machine learning* y explotación de datos usando técnicas de minería de datos son implementadas usando las librerías Geopandas, Scikit-Learn, TensorFlow y Keras. Al igual que en la instancia de la arquitectura de Big Data en los sistemas de salud y bienestar analizados en capítulos anteriores, TensorFlow es empleado como *backend* de Keras para facilitar el desarrollo de prototipos de modelos de *machine learning*. Además, el módulo implementa un entorno de desarrollo a través de la aplicación web Jupyter Notebook.

Dominio de aplicaciones

- **Gestión de visualización:** las funcionalidades para la visualización son implementadas a través de una GUI web capaz de desplegar gráficas visuales de los datos. La GUI es diseñada usando Bootstrap, Chartjs, Socket.io, Angular y Leaflet para el despliegue de mapas. Además, un servidor web basado en Nodejs es implementado como *backend*.
- **Gestión de consultas:** las funcionalidades para realizar consultas puntuales sobre los datos emplea las interfaces y librerías JavaScript que permiten la consulta de datos del tipo geoespacial de la instancia de base de datos HBase, y el lenguaje SQL para consultar la instancia de base de datos PostGIS.

Dominio de operaciones

- **Gestión de acceso y seguridad:** Las funcionalidades de seguridad son implementadas por Apache Ranger. Está herramienta soporta la definición de políticas de acceso y seguridad para controlar las plataformas de procesamiento como Hadoop, Spark y las instancias de almacenamiento como HDFS, HBase y Postgres y PostGIS).
- **Gestión de monitorización:** la monitorización de los recursos utilizados por las plataformas y herramientas involucradas en los procesos de la arquitectura es realizada a través de Apache Ambari. La monitorización utilizando Apache Ambari permite mantener bajo supervisión el clúster Hadoop (HDFS) y de Apache Spark.
- **Gestión de procesos:** la gestión de los registros (*logs*) de los procesos es realizado a través de la aplicación LogStash. De esta manera,

el control de los *logs* permite identificar fácilmente fallos en la ejecución de los procesos implementados en este sistema y corregirlos para cumplir los acuerdos de nivel de servicio.

6.6.2 Integración de la arquitectura de Big Data con el espacio de datos portuarios

El proceso de recolección de datos definido en la arquitectura (sección 3.6) es utilizado para integrar la arquitectura de Big Data con el conector IDS del EDP. Para ello, una instancia de este proceso es implementado utilizando las funcionalidades provistas por la plataforma Apache NiFi. La instancia tiene el objetivo de recolectar datos del EDP, almacenar los datos en HDFS, y reenviar los datos al sistema de comunicaciones de la arquitectura implementado con Apache Kafka. El proceso definido por la arquitectura está compuesto por los subprocesos de: conexión con fuente IoT, extracción de datos, conversión de formatos, y la carga de datos. La instancia del proceso es implementada a través de un flujo de datos en la plataforma Apache NiFi. El flujo está compuesto por procesadores capaces de realizar una tarea específica. Cada procesador implementa un subproceso, así tenemos a los procesadores: ListenHTTP (conexión con fuente IoT y extracción de datos), PutParquet (conversión de formato), y PublishKafka (carga de datos a la plataforma de procesamiento en tiempo real). La Figura 6.11 muestra el proceso de recolección de datos con sus procesadores y su integración con el EDP.

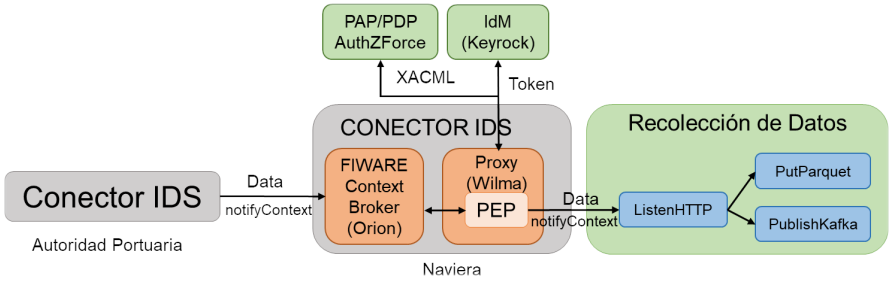


Figura 6.11: Integración de la Arquitectura Big Data con el conector IDS

El procesador ListenHTTP permite implementar los subprocesos de conexión con fuente IoT y extracción de datos. Para ello, el procesador ListenHTTP implementa un servidor HTTP que se encuentra a la espera de solicitudes POST escuchando en el puerto 8080. La extracción de datos se produce a través del envió de notificaciones de contexto usando las interfaces NGSI de la plataforma FIWARE. El flujo de mensajes entre los

diferentes conectores IDS durante el subproceso de extracción de datos se muestra en la Figura 6.12.

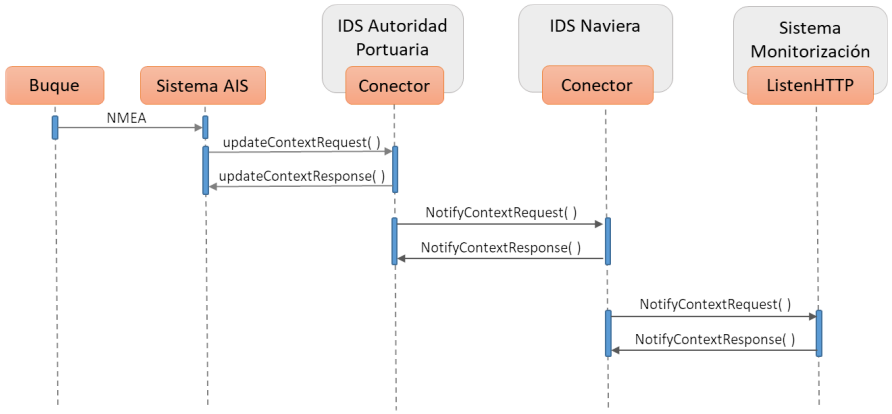


Figura 6.12: Flujo de mensajes entre los conectores IDS y procesador ListenHTTP para la recolección de datos

De acuerdo con el ejemplo mostrado en la Figura 6.12, la entidad *buqueObservado* es actualizado con cada mensaje enviado por el buque de la naviera a través del sistema AIS de la autoridad portuaria. Esta entidad se actualiza en el conector IDS de la autoridad portuaria. La actualización se reenvía automáticamente al conector IDS de la naviera luego de que el PEP verifícase las autorizaciones respectivas bajo las políticas PAP/PDP, tal y como se detalló en la subsección 6.5.3. Posteriormente, el conector IDS de la naviera reenvía la actualización de contexto al procesador ListenHTTP. Este reenvío de notificaciones de contexto es posible con la federación entre GE Orion Context Bróker de cada conector IDS y su modalidad tipo *push* y la suscripción para recibir las actualizaciones de contexto de las entidades autorizadas para el conector IDS de la naviera.

El subproceso de conversión de formato es implementado en el flujo NiFi a través del procesador PutParquet. Este procesador convierte los datos en formato JSON-ld enviados por el conector IDS de la naviera al formato Apache Parquet. La conversión es realizada utilizando el esquema de datos Avro, el cual se infiere del archivo JSON-ld enviado. Los datos convertidos en formato Parquet son enviados al repositorio de datos históricos en el clúster HDFS.

El subproceso de carga de datos es implementado usando el procesador PublishKafka. La información de contexto que llega al procesador ListenHTTP es utilizada por el procesador PublishKafka para re-direccionar los datos hacía la plataforma de Big Data destinada al procesamiento en tiempo real.

Los datos en formato JSON-ld son publicados en el *topic* correspondiente configurados en el bróker Kafka. En este bróker se encuentran configurados los *topics* "ais" y "terminal" que hacen referencia a los sistemas de orígenes de los datos. Posteriormente, una instancia del proceso de extracción de información utilizará estos datos a través de la suscripción al respectivo *topic* para recibir los datos y ejecutar un análisis descriptivo, como se verá en la siguiente subsección.

6.6.3 Extracción de los KPI

Los datos compartidos a través del EDP presentan varias oportunidades para extraer información útil que describa la situación actual de los buques de la naviera. La extracción de los KPI del Big Data compartido en EDP se realiza a través de una instancia del proceso de extracción de información definido por la arquitectura de Big Data para IoT.

El proceso de extracción de información se realiza utilizando los flujos de datos en tiempo real y los datos almacenados en el repositorio histórico. La instancia del proceso de extracción de información está compuesto por los subprocesos de carga de datos, pre-procesamiento y análisis descriptivo en tiempo real o el análisis descriptivo del histórico de datos. La combinación de los dos tipos de procesamiento permite desarrollar diversos KPI de importancia para la naviera. La Figura 6.13 muestra un diagrama lógico de la instancia del proceso de extracción que siguen los KPI desarrollados. A continuación se describen los KPI desarrollados empleando los datos compartidos en el EDP.

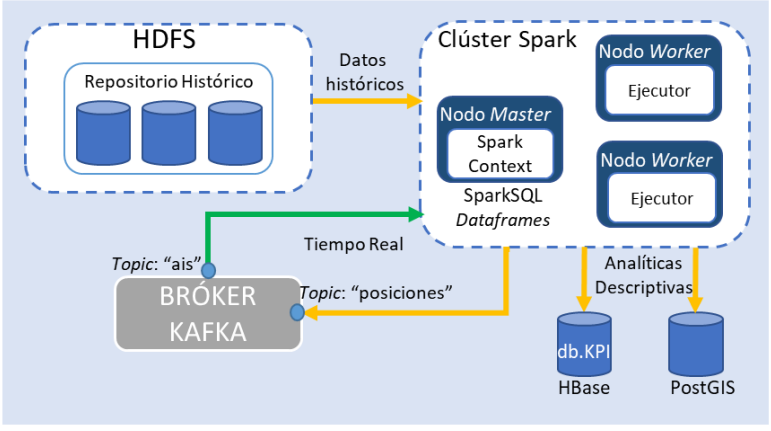


Figura 6.13: Diagrama de la instancia del proceso de extracción de información

- **KPI.1**: Tiempo promedio ocupado por buque en el terminal de contenedores a la semana.
- **KPI.2:** Ocupación de la terminal de contenedores a la semana.
- **KPI.3:** Tiempo promedio de espera de los buques de la naviera en las inmediaciones del puerto a por un terminal libre.
- **KPI.4:** Consumo de combustible promedio de los buques de la naviera durante el tiempo de espera a por un terminal de operaciones.

KPI.1

El tiempo promedio de ocupación por buque de un muelle en la terminal de contenedores es calculado empleando los datos históricos almacenados en HDFS provenientes del conector IDS de la terminal de contenedores. El pseudocódigo mostrado en la Figura 6.14 muestra una aproximación del código desarrollado para generar este KPI. Inicialmente, los datos son cargados de manera tabular en un (*Dataframe*) empleando SparkSQL (línea código 1). El *Dataframe* contiene la marca de tiempo (fecha-hora) del atracado del buque en la terminal y la marca de tiempo (fecha-hora) de la salida del buque de la terminal de contenedores. El tiempo de permanencia del buque en el terminal de contenedores es el resultado de la resta de la marca de tiempo de salida menos la de entrada (línea código 2-3). El resultado agrega una nueva columna llamada "delta" al *dataframe* inicial donde indica en horas el tiempo de ocupación de un buque en la terminal de contenedores (línea código 2-3). Finalmente, los datos contenidos en el *dataframe* son almacenados en la instancia de base de datos HBase a través de la función (*save_to_HBase*) (línea código 4).

```
1 df_KPI1 = spark.read.parquet("terminal_contenedores.parquet")
2 df_KPI1 = df_KPI1.withColumn("delta",
3       df_KPI1("finOperaciones")  - df_KPI1("inicioOperaciones"))
4 df_KPI1.save_to_HBase("db.KPI1")
```

Figura 6.14: Pseudocódigo extracción KPI.1 usando *dataframes* de SparkSQL

KPI.2

La ocupación de la terminal a la semana es calculada mediante una agregación por semana de los datos utilizados previamente calculados en el KPI.1. De esta manera se puede conocer el número de buques que ocupan la terminal de contenedores a la semana. El pseudocódigo mostrado en la Figura 6.15 muestra una aproximación del código desarrollado para

generar este KPI. La agregación es realizada empleando las funcionalidades provistas por SparkSQL en el tratamiento de *Dataframes*. El *dataframe* generado para el KPI.1 es agrupado dentro de una ventana de una semana, considerando la fecha de fin de operaciones como argumento de agrupación (línea de código 1). Posteriormente, se cuenta el número de buques que han terminado las operaciones en el terminal de contenedores (línea código 2). Luego, el algoritmo extrae información sobre valores máximo, mínimo y promedio de la ocupación de los buques en la terminal de operaciones del *Dataframe* agregado, por medio de las funciones de estadística descriptiva provistas por SparkSQL (*describe()*) (línea código 3). Finalmente, los datos contenidos en el *dataframe* son almacenados en la instancia de base de datos HBase a través de la función (*save_to_HBase*) (línea código 4).

```
1 df_KPI2 = df_KPI1.groupBy(window(col("finOperaciones"), "1 week"))
2 df_KPI2 = df_KPI2.agg(count("delta") as "ocupacion_semanal")
3 df_KPI2 = df_KPI2.describe()
4 df_KPI2.save_to_HBase("db.KPI2")
```

Figura 6.15: Pseudocódigo extracción KPI.2 usando *dataframes* de SparkSQL

KPI.3

El tiempo promedio de espera de un buque de la naviera al ancla en el puerto es calculado empleando los datos compartidos desde el conector IDS de la autoridad portuaria. Estos datos requieren ser procesados en tiempo real, por tanto los datos del conector IDS son enviados a la plataforma de procesamiento en tiempo real Apache Spark Streaming por medio de la plataforma de comunicación Apache Kafka. SparkSQL de Apache Spark es utilizado para realizar el análisis de los datos en tiempo real. Particularmente, SparkSQL tiene la capacidad de ejecutar consultas estilo SQL o estructurar los datos en *Dataframes* de forma similar a la librería Pandas de Python pero en tiempo real a través de su motor llamado Structured Streaming. Este motor toma ventaja del procesamiento en *micro-batch* para ejecutar consultas altamente escalables, rápidas y tolerantes a fallas. La Figura 6.16 muestra de manera conceptual el algoritmo empleado para calcular el tiempo de espera por buque.

El flujo constante de mensajes AIS es filtrado para seleccionar únicamente los mensajes enviados por buques de la naviera cercanos al puerto. Este filtro es definido a través de un polígono de 4 puntos geoespaciales (latitud, longitud) formando un cuadrado. Los mensajes cuya posición estén dentro del cuadrado son seleccionados para proseguir con el análisis. Los mensajes filtrados son acumulados en una ventana deslizante configurada con una duración de una hora.

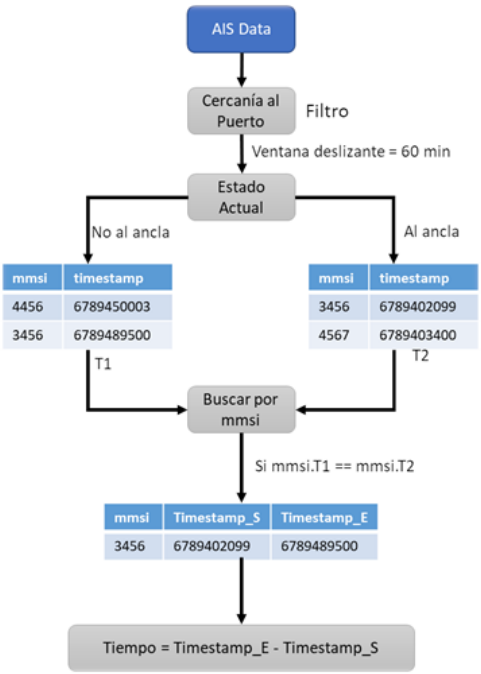


Figura 6.16: Algoritmo cálculo del tiempo espera por buques

Posteriormente, los mensajes dentro de la ventana son estructurados en una tabla (*dataframe*). A continuación, el *dataframe* se divide en dos *dataframes*, el primero con los mensajes que contiene como estado de navegación al ancla (tabla al ancla) y el segundo con los mensajes que contiene otros estados de navegación como atracado, maniobrando y crucero (tabla no al ancla). Las tablas únicamente contienen el código identificativo del buque MMSI y la marca de tiempo (segundos desde enero 01-1970) del reporte de su estado de navegación.

A continuación, el código identificativo de los buques MMSI del primer *dataframe* al ancla es buscado dentro del *dataframe* no al ancla. Esto es realizado de manera sencilla con la función *join()* provista por SparkSQL. De esta manera, los MMSI del *dataframe* al ancla y no al ancla son comparados y se seleccionan solo los registros donde el MMSI está presente en los dos *dataframes*. Esto indica que el buque ya no está al ancla y que su estado de navegación ha cambiado, por lo que se selecciona el primer mensaje con el estado de navegación nuevo y se almacena la marca de tiempo.

Posteriormente, un *dataframe* de resumen almacena el código identificativo del buque MMSI, la marca de tiempo de inicio del estado al ancla y la marca de tiempo de fin del estado al ancla. Los datos del *dataframe* son enviados a una tabla de una instancia en la base de datos PostGIS de resumen de resultados en tiempo real. Finalmente, una consulta es generada para extraer los datos de la tabla de la base de datos PostGIS. La consulta extrae el promedio, el valor máximo y el valor mínimo en minutos de la permanencia de los buques en estado de navegación al ancla. La información extraída es presentada en la GUI de visualización de datos en forma de KPI. Esta información indica a la naviera cuanto deben esperar sus buques en promedio por un lugar de atraque en el puerto. De esta manera, las navieras pueden ordenar a su flota reducir la velocidad de sus buques antes de la llegada al puerto para ahorrar combustible consumido durante la espera en la zona de anclaje del puerto.

KPI.4

Al igual que con el KPI del tiempo de espera de un buque en la zona de anclaje, el consumo de combustible promedio de los buques de la naviera durante el tiempo de espera es calculado empleando los datos AIS compartidos por la autoridad portuaria. La Figura 6.17 muestra de manera conceptual el algoritmo empleado para el cálculo de la estimación del consumo de combustible basado las ecuaciones del estado del arte para esta estimación en la literatura especializada [200] [192].

El tiempo al ancla calculado por buque en el KPI.3 es utilizado para la estimación de combustible de los buques durante el modo de operación al ancla. Además, el algoritmo emplea un *dataframe* que contiene la información sobre la potencia del motor principal (ME, del inglés *Main Engine*), la potencia del motor auxiliar (AE, del inglés *Auxiliary Engine*) y el consumo especifico de combustible base (SFC, del inglés *Specific Fuel Consumption*) por buque de la naviera. La información sobre los ME y AE utilizado por los buques es cargada desde la base de datos de la naviera donde se almacena las características de sus buques. Por otro lado, de acuerdo con la organización internacional marítima (IMO, del inglés *International Maritime Organization*), el valor SFC ha sido estimado en 195 g/kWh para buques construidos a partir del 2001 y en 205 g/kWh para buques construidos entre 1984 y 2000 [192].

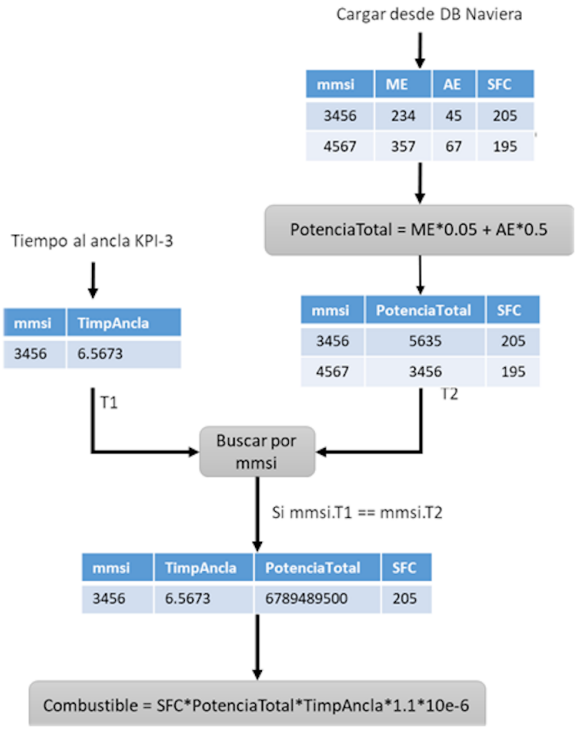


Figura 6.17: Algoritmo estimación del combustible consumido durante tiempo de espera al ancla

Utilizando el *dataframe* con información de los buques de la naviera se estima la potencia total que es capaz de alcanzar el buque en el modo de operación al ancla. Para el modo de operación al ancla, el motor auxiliar se activa, mientras el motor principal se mantiene en reposo. Por ello, el

factor de carga (LF) para este modo de operación se estima en 5 % de la potencia total para el motor ME y 50 % de la potencia total para el motor AE [199]. La potencia actual resulta de la suma total de la potencias por el factor de carga (kW). La ecuación 6.1 muestra esta operación.

$$P = ME \times LF + AE \times LF \tag{6.1}$$

A continuación, el código identificativo de los buques MMSI de la primera tabla (*dataframe* al ancla KPI.3) es buscado dentro de la tabla de las potencias totales por buque. Esto es realizado de manera sencilla con la función *join()* provisto por SparkSQL. De esta manera, los MMSI de la tabla al ancla KPI.3 y de las potencias totales son comparados y se seleccionan solo los registros donde el MMSI está presente en las dos tablas.

Finalmente, la estimación del combustible en toneladas (F) es resultado de la ecuación 6.2; donde P es la potencia total en kW, T es el tiempo total de espera en horas, y SFC es el consumo específico de combustible base en g/kWh [200].

$$F = P \times T \times SFC \times 1,1 \times 10^{-6} \tag{6.2}$$

La estimación del consumo de combustible cuando el buque está en modo de operación al ancla es almacenada en la base de datos PostGIS. Posteriormente, los valores máximos, mínimos y promedio de la estimación del consumo de combustible son obtenidos mediante una consulta a la tabla de la instancia de base de datos en el proceso de visualización de los KPI.

6.6.4 Visualización de los KPI

El servicio de visualización de los KPI relevantes para la naviera tiene el objetivo de mostrar gráficamente el resultado del análisis descriptivo del Big Data realizado con el proceso de extracción de información. El servicio de visualización proporcionado por el sistema de monitorización utiliza una instancia del proceso de visualización de analíticas y KPIs definido por la arquitectura. La Figura 6.18 muestra el diagrama del proceso implementado para este caso.

El proceso de visualización de analíticas y KPI tiene como eje principal una GUI web. Esta GUI compila los KPI que son representados en forma gráfica para una fácil interpretación por parte de los usuarios finales. La aplicación GUI web es implementada usando el modelo de diseño de aplicaciones web basado en *backend-frontend*.

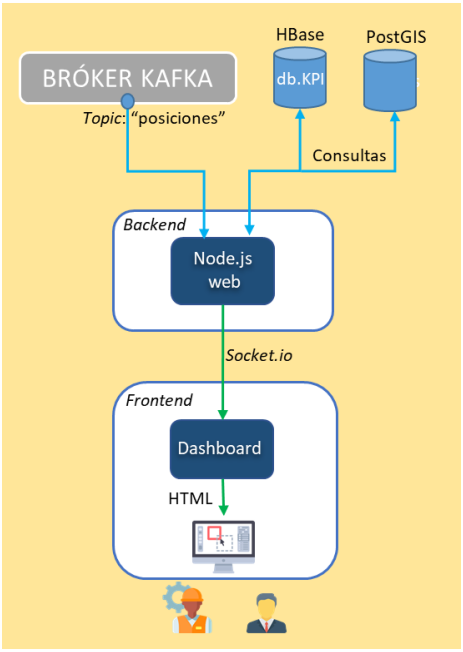


Figura 6.18: Diagrama de la instancia del proceso de visualización de KPI

El *backend* cumple como intermediario entre la aplicación de visualización y los repositorios de datos del sistema. Los repositorios de los resultados están compuestos por la instancia de base de datos HBase donde se almacena los resultados del procesamiento tipo *batch*, por la instancia de base de datos PostGIS donde se almacena los resultados del procesamiento en tiempo real, y por la plataforma de comunicación Apache Kafka donde llegan la posición de los buques en tiempo real. El *backend* es un programa servidor implementado utilizando las librerías Nodejs. El programa servidor de la aplicación GUI realiza las conexiones con las instancias de base de datos y con el bróker de comunicación de Apache Kafka. Además, el programa servidor tiene previamente configuradas las consultas a las instancias de base de datos que sirven para generar las gráficas. Los datos recolectados por el *backend* son enviados al *frontend* utilizando la librería de Javascript Socket.io.

Por otro lado, el *frontend* es implementado utilizando el *framework* Bootstrap de HTML, la librería Javascript Angular, Chat.js y la librería Leaflet para el diseño de mapas interactivos. Los datos enviados desde el *backend* a través de Socket.io son convertidos a gráficos donde se muestra los KPI. El mapa interactivo muestra la posición de los buques de la naviera cercanos

al puerto de Valencia a través de un icono de color verde para representar su posición en el mapa. El color del icono puede cambiar a color rojo para indicar que el buque de la naviera ha cambiado el modo de operación al modo al ancla. De esta manera, la empresa naviera podrá conocer cuando sus buques son puestos a la espera por un muelle para su atraque y posterior carga y/o descarga de contenedores.

6.7 Evaluación del sistema

La evaluación del sistema de monitorización de la flota de buques de la naviera tiene el objetivo de demostrar el funcionamiento del EDP y mostrar como la compartición de datos a través de EDP y el uso de la arquitectura de Big Data permiten extraer información útil para mejorar las operaciones de una empresa naviera. Además, la evaluación muestra una comparativa de la instancia de la arquitectura propuesta en relación con arquitecturas de Big Data utilizadas en la implementación de sistemas en el área marítima.

6.7.1 Escenario de pruebas

El sistema fue implementado bajo un entorno de pruebas controlado. El sistema es desplegado sobre varias máquinas virtuales en un servidor con las siguientes características: servidor FUJITSU Intel Xeon E3-1220 v5 3,00 GHz CPU, memoria RAM 64GB, y plataforma VMware vSphere. La Figura 6.19 muestra un diagrama del despliegue del sistema para las pruebas sobre las diferentes máquinas virtuales creadas. La máquina virtual 1 corresponde a la autoridad portuaria, la máquina virtual 2 al operador del terminal de contenedores y las máquinas virtuales 3, 4 y 5 a la empresa naviera.

La validación de la arquitectura es realizada a través del uso de dos conjuntos de datos. La Tabla 6.2 describe los conjuntos de datos (*dataset*) utilizados en la experimentación. Estos conjuntos de datos son empleados para verificar el funcionamiento del procesamiento tipo *batch* y el almacenamiento. Adicionalmente, el procesamiento en tiempo real es probado mediante el uso de los datos desde la web de AIS-HUB [218]. AIS-HUB provee un servicio de compartición de datos AIS gratuitos para el seguimiento de buques.

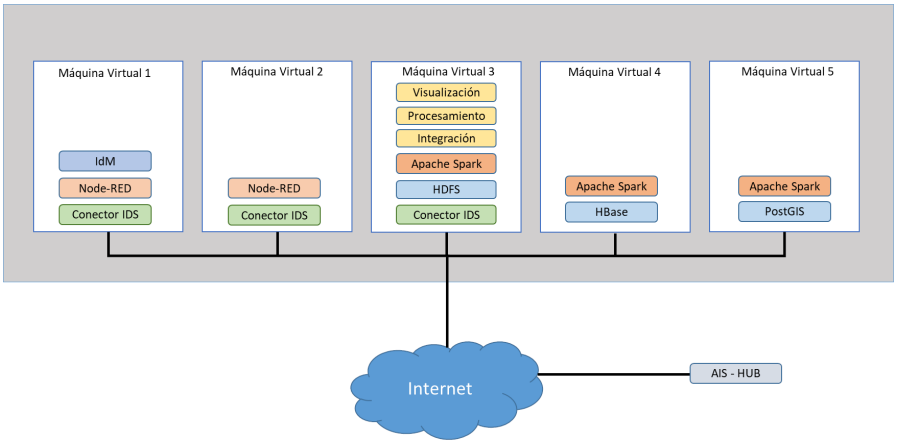


Figura 6.19: Esquema del despliegue del sistema para pruebas

Tabla 6.2: Detalle conjuntos de datos portuarios

Dataset	Tamaño	Período
AIS	10 GB	2016/01/01 - 2016/03/31
Operaciones Terminal Contenedores	520 MB	2014/01/01 - 2019/01/31

6.7.2 Ejecución de prueba

Los conjuntos de datos indicados en la Tabla 6.2 son usados durante la compartición en el EDP para extraer los KPI de interés para la naviera. El resultado del análisis de los datos es mostrado a continuación:

Tiempo promedio ocupado por buque en el terminal de contenedores a la semana

El conjunto de datos de las operaciones del terminal de contenedores es utilizado para generar este KPI. Este conjunto de datos corresponde a datos del tipo serie de tiempo. El análisis de este tipo de datos es realizado a través de una descomposición en sus componentes de tendencia y temporada. Este análisis es realizado empleando la librería Python Statsmodels. Como parte inicial, las funciones proporcionadas por la librería Statsmodels extrae los componentes desde los datos estructurados en *dataframes* con SparkSQL. La Figura 6.20 muestra la descomposición en componentes de los datos compartidos por el terminal portuario.

La tendencia es a la alza, es decir que el terminal portuario cada mes está soportando un incremento en sus operaciones de carga y descarga. Además,

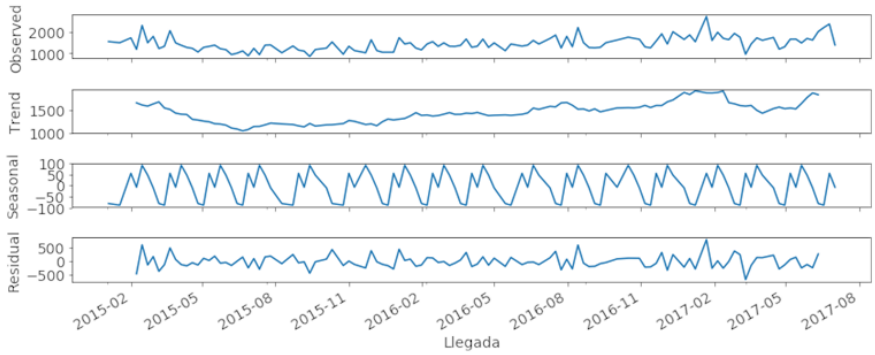


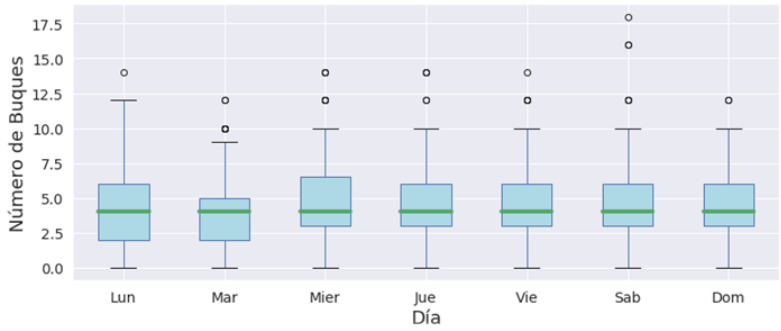
Figura 6.20: Descomposición en temporada y tendencia del tiempo de permanencia de buques en el terminal de contenedores a la semana.

la temporada indica un patrón que se repite cada dos meses con dos picos que representan la alta carga operativa en esas semanas. Estos componentes demuestran que el tiempo promedio de los buques no es el mismo en periodos cortos (temporada) y variará en los próximos años (tendencia). Al igual que el tiempo promedio varía en el tiempo, los valores máximos y mínimos también lo harán. Por ello, el cálculo de los valores máximos y mínimos es realizado usando el promedio de los valores mínimos por semana al igual que los valores máximos por semana. Como resultado se obtiene un valor máximo de 64,28 horas, un valor mínimo de 1,85 horas y un promedio de 23,31 horas. Estos valores promedio, máximo y mínimo tiempo de ocupación de un terminal son empleados como KPI para estimar los tiempos que un buque de la naviera permanecerá en el puerto. Así por ejemplo, la naviera estima que un buque de su flota permanecerá en promedio 23,31 horas cargando y descargando contenedores en la terminal de contenedores.

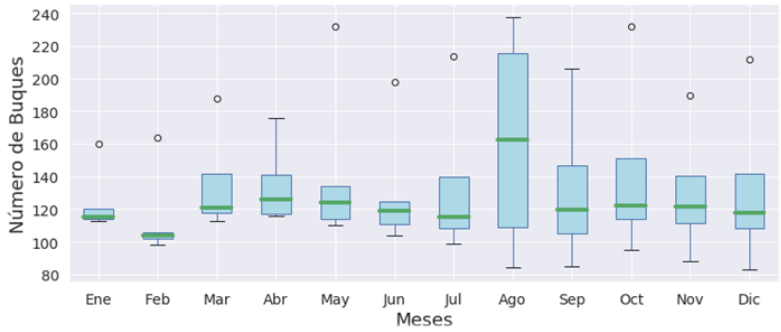
Ocupación de la terminal de contenedores a la semana

La ocupación de la terminal de contenedores permite conocer cuántos buques a la semana es capaz de soportar el terminal portuario. Similarmente al proceso anterior, una agregación de datos es realizada para contar el número de buques por días, semanas y meses. Los resultados son resumidos en gráficas de caja-bigotes para representar los días y meses con más carga operacional de la terminal de contenedores, y se muestran en la Figura 6.21.

Los resultados muestran que agosto es el mes con más carga operacional de la terminal y que el sábado es el día de la semana con más carga operacional de la terminal, soportando 35 buques y 6 buques en promedio, respectivamente. Esta información proporciona a la naviera detalles importantes para



(a) Días de la semana



(b) Meses del año

Figura 6.21: Ocupación del terminal de contenedores

la planificación de sus operaciones. En este caso, la naviera puede planificar el arribo de sus buques en los días con menos carga operacional para evitar que sus buques permanezcan esperando mucho tiempo en las inmediaciones del puerto. Esta espera conlleva un gasto importante para la naviera en combustible y una baja eficiencia de sus buques de carga.

Tiempo promedio de espera de un buque al ancla

El tiempo promedio de espera de un buque de la naviera al ancla en el puerto es calculado empleando los datos compartidos del conector IDS de la autoridad portuaria. La Figura 6.22 muestra en promedio el tiempo de espera (horas) a la semana que un buque de la naviera espera al ancla a por un muelle libre para iniciar las operaciones de carga y descarga. En ella se observa que los días martes y viernes son días cuando los buques de la naviera deben esperar en aproximadamente 11 horas.

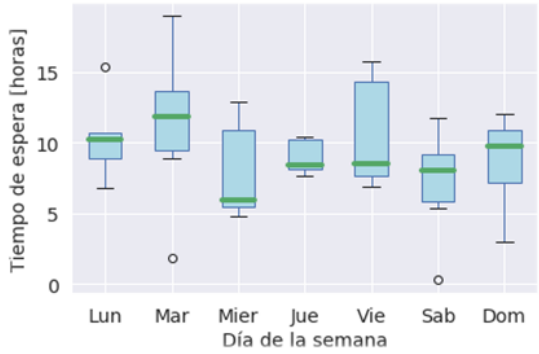


Figura 6.22: Tiempo de espera de un buque al ancla por día de la semana

El análisis descriptivo del tiempo de espera de un buque al ancla muestra que en promedio se espera 9,59 horas y que en el peor de los casos el buque de la naviera permanecerá anclado por 23,77 horas.

Consumo de Combustible al ancla

El consumo de combustible al ancla de un buque de la naviera es calculado empleando los datos previamente procesados en el KPI-3. El resultado del procesamiento de los datos siguiendo el algoritmo presentado en la subsección 6.6.3 presenta en promedio un consumo de 4,01 toneladas durante el tiempo de espera al ancla. La Figura 6.23 muestra la distribución del consumo de combustible en toneladas. En ella se observa que sigue una distribución en forma de campana y que concentra la mayor cantidad de

valores entre las 0 - 2,5 toneladas. Esto indica que la mayoría de los buques de la naviera que están en la zona de anclaje gastan entre 0 y 2,5 toneladas.

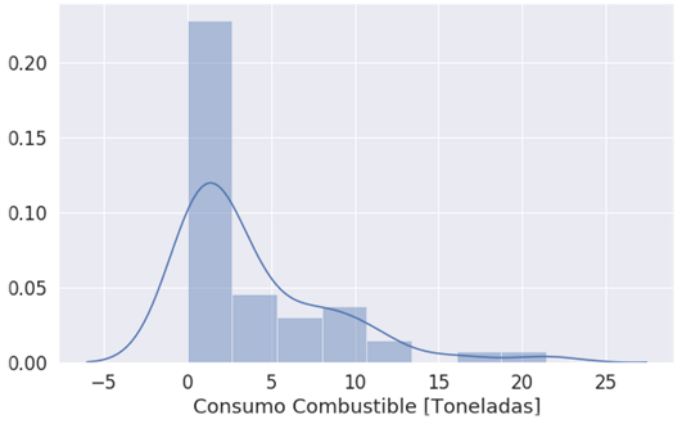


Figura 6.23: Distribución del consumo de combustible (toneladas).

Visualización de los KPI

Los KPI son compilados en el *dashboard* desarrollado para la visualización de los resultados. El *dashboard* de la naviera muestra los KPI en 4 medidores y muestra la ubicación de los buques de la naviera cerca del puerto de Valencia en tiempo real. La Figura 6.24 muestra la GUI web para la visualización de los datos y mejorar la interpretación del Big Data para la toma de decisiones en la planificación de las operaciones de la naviera. Así por ejemplo, el *dashboard* permite concluir que actualmente las operaciones de la naviera no son eficientes, ya que se observa que el color predominante es el rojo en todos los KPI. El escenario ideal mostraría una gran porción de los medidores de color verde y muy poco color rojo.

6.7.3 Análisis de mejoras en la operación de la naviera

El problema del tiempo de espera de la flota de los buques de la naviera en la zona de anclaje conlleva un gasto innecesario en el combustible. Además, el consumo de combustible provoca la liberación de gases contaminantes peligrosos para la salud. Estos efectos serían minimizados con el uso del sistema de monitorización basado en la arquitectura de Big Data para IoT y explotando el uso de un entorno IDS para mejorar la planificación de las operaciones. Una de las estrategias en la planificación es la reducción de la velocidad conociendo de antemano este posible tiempo estimado de espera en la zona de anclaje. De esta manera, el gasto de combustible se vería

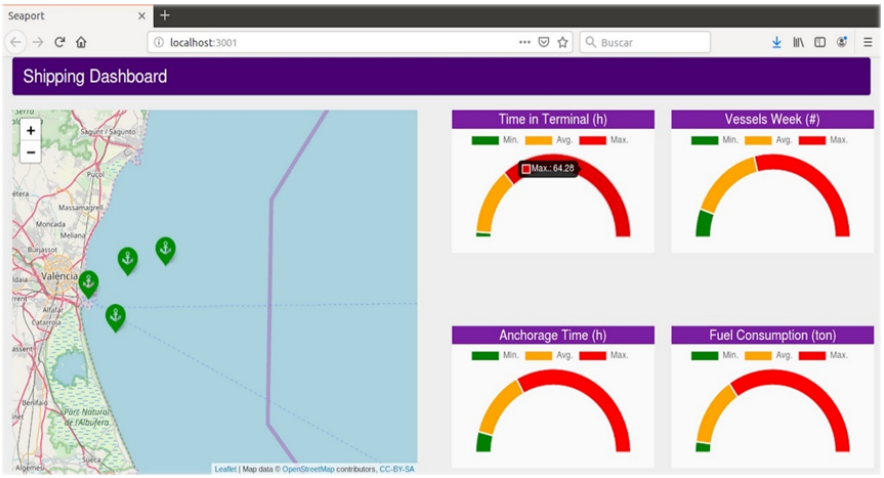


Figura 6.24: GUI visualización de datos de los KPI naviera

reducido. Es por ello que la implementación del sistema propuesto presenta beneficios económicos y medio ambientales.

En cuanto a los beneficios económicos se prevé una reducción en los costos de combustible para la empresa naviera. De lo analizado anteriormente, la empresa naviera consume 4,01 toneladas de combustible en promedio por buque durante su tiempo de espera. Si consideramos que el costo del combustible a fecha 9 Octubre 2019 es de $ 697,00 por tonelada de combustible con una tasa de cambio de 1 euro = 1,096 dólares, el gasto de combustible promedio por buque es de €2550,15 durante su tiempo de espera. La Figura 6.25 muestra semanalmente el gasto total en combustible durante la evaluación del sistema para las semanas del 6 al 10 del año.

Por otro lado, en cuanto al impacto medio ambiental se prevé una reducción de la cantidad de gases contaminantes que se liberan al ambiente producto del consumo de combustible durante el tiempo de estancia de los buques de la naviera en la zona de anclaje. El cálculo de los contaminantes liberados en promedio por buque durante el tiempo de espera depende del factor de cada contaminante. De acuerdo con el estado de la técnica especializada, el factor de contaminante (fc) para el CO_2, SO_2, y Óxidos de Nitrógeno (NO_x) es de 690,71 g/kWh, 2,12 g/kWh, y 13,90 g/kWh, respectivamente [219]. Siguiendo la ecuación 6.3, donde el consumo de combustible CC se estimó en 4,01 toneladas en promedio por buque durante el tiempo al ancla, y el SFC está estimado en 195 g/kWh [192]; como resultado se obtiene que un buque durante el tiempo al ancla en promedio libera 14,2 toneladas de CO_2, 0,043 toneladas de SO_2 y 0,288 toneladas de NO_x. Estos valores se

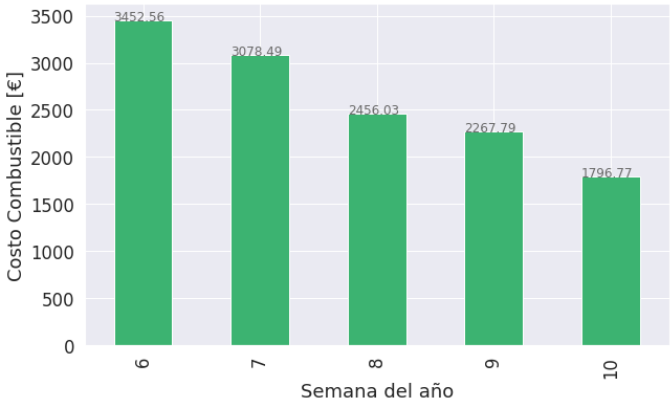


Figura 6.25: Gasto total de combustible durante la evaluación del sistema

verían seriamente reducidos aplicando una adecuada planificación de las operaciones de los buques a través de la información que proporciona el sistema de monitorización. Como consecuencia, los contaminantes en el medio ambiente y sobre todo en la zona cercana al puerto se reducirían.

$$Contaminate = \frac{CC \times fc}{SFC} \tag{6.3}$$

6.7.4 Comparativa de la arquitectura general con arquitecturas similares aplicadas al transporte

En el ámbito del transporte marítimo inteligente se han realizado varias propuestas de arquitecturas para la gestión del gran volumen de datos y extraer información útil. En esta evaluación consideramos las siguientes características importantes relacionadas con las 3Vs del Big Data y de acuerdo a los requerimientos en este tipo de sistemas:

- Integración con múltiples fuentes: muestra la capacidad de la arquitectura para soportar varias fuentes de datos IoT.
- Procesamiento de datos en tiempo real: muestra la capacidad de la arquitectura para soportar la velocidad de los datos con el fin de generar un tratamiento eficiente de los datos.
- Procesamiento de datos *batch*: muestra la capacidad de la arquitectura para soportar el tratamiento de un gran volumen de datos eficientemente.

- Análisis de los datos: muestra la capacidad de la arquitectura para generar análisis del tipo descriptivo, predictivo, de diagnóstico o prescriptivo del gran volumen de datos recolectado.
- Almacenamiento escalable: muestra la capacidad de la arquitectura para soportar el gran volumen de datos y que estos puedan ser reutilizados en los análisis.
- Visualización: muestra la capacidad de la arquitectura para desplegar los resultados de los datos de forma visual para una mejor interpretación de los datos.
- Seguridad: muestra la capacidad de la arquitectura para proteger los diferentes módulos de la arquitectura.
- Gestión: muestra la capacidad de la arquitectura para administrar los diferentes módulos de la arquitectura.
- Soberanía de Datos: muestra la capacidad de la arquitectura para que la fuente de datos controle los datos que serán utilizados para el procesamiento de datos.

La arquitectura DatAcron propuesta inicialmente en [208] está enfocada en el procesamiento de los datos en tiempo real generados por el sistema AIS. Esta arquitectura está basada en la arquitectura Lambda de Big Data para proveer un procesamiento del tipo *Batch* y en tiempo real. Además, la arquitectura considera un almacenamiento eficiente a través del almacenamiento de archivos RDF (*Resource Description Framework*). En está primera etapa de la arquitectura se enfoca en la integración de una fuente RDF y el procesamiento de los datos siguiendo la arquitectura Lambda. En una mejora realiza posteriormente propuesta en [207] se considera la integración de varias fuentes, el análisis de los datos y la visualización de los resultados. La arquitectura general propuesta en esta tesis brinda capacidades de seguridad, gestión y soberanía de datos, las mismas que no fueron consideradas en la arquitectura DatAcron. Además, la arquitectura general propuesta en esta tesis presenta un proceso para ejecutar los modelos desarrollados durante el análisis de datos con el fin de predecir situaciones.

Por otra parte, la arquitectura propuesta por el proyecto Big Data Ocean en [206] está destinada a la gestión eficiente de los datos generados por el sistema de transporte marítimo AIS, por lo que no considera la integración con otro tipo de fuentes de datos. A diferencia de la arquitectura general propuesta en está tesis que considera varias fuentes de datos, principalmente las fuentes IoT; y sobre todo se considera la soberanía de datos la cual es de

suma importancia para controlar los datos en entornos donde varias partes interesadas comparten sus datos como fuente del Big Data.

Finalmente, en comparación con otras propuestas las cuales se enfocan solo en la potencialización de una característica en especial. Por ejemplo, [209] propone una arquitectura de Big Data enfocado en la gestión de los datos para la visualización de los datos portuarios con tecnologías 3D. Por otro lado, [220] propone una arquitectura para el análisis de la gran cantidad de datos generados por AIS enfocado en el procesamiento *batch*. A diferencia de estas arquitecturas, la arquitectura de Big Data para IoT de esta tesis es una arquitectura integral donde se consideran todas las etapas del ciclo de vida del Big Data y bajo estándares y recomendaciones internacionales lo que garantiza su interoperabilidad.

La Tabla 6.3 muestra un cuadro comparativo de las características destacadas en relación con arquitecturas similares.

Tabla 6.3: Comparativa arquitectura general con arquitectura similares transporte marítimo

Característica	Arquitectura					
	[208]	[209]	[207]	[206]	[220]	General
Integración múltiples fuentes			•			•
Procesamiento Tiempo Real	•	•	•	•		•
Procesamiento *Batch*	•	•	•		•	•
Almacenamiento eficiente	•	•	•	•	•	•
Análisis		•	•	•	•	•
Visualización		•	•	•		•
Seguridad				•		•
Gestión				•		•
Soberanía de Datos						•

6.8 Conclusiones

La transformación de los puertos marítimos en puertos inteligentes presentan retos que aún deben ser resueltos para facilitar la integración de las tecnologías habilitadoras de Big Data, IoT, IA, *cloud computing*, entre otras. Uno de los retos que inhibe la explotación de los beneficios del Big Data es la falta de comunicación entre los actores involucrados en los procesos portuarios. La interoperabilidad entre aplicaciones IoT y la creciente preocupación de la pérdida de control de los datos ocasionan que el Big

Data no pueda ser liberado para extraer información útil en beneficio de las operaciones portuarias. Una de las soluciones en la industria es el establecimiento de un entorno seguro de compartición de datos a través de la arquitectura de referencia IDS. En este caso de uso se ha validado el uso de la arquitectura de Big Data para IoT en un entorno industrial marítimo con el fin de explotar los datos provenientes de un espacio virtual de datos basado en la arquitectura de referencia IDS.

El uso de la arquitectura de referencia IDS permitió implementar el espacio virtual de datos EDP, donde los actores portuarios compartieron sus datos (terminal de contenedores, autoridad portuaria y empresa naviera). Además, el uso de IDS facilitó la compartición del Big Data generado por aplicaciones de IoT solucionando los problemas de interoperabilidad y control sobre los datos. Esto a su vez permitió la explotación del Big Data compartido utilizando una instancia de la arquitectura de Big Data para IoT propuesta en el capítulo 3. La explotación del Big Data utilizando el proceso definido por la arquitectura generó información relacionada con el tiempo de espera de los buques en el puerto, el tiempo de la operación carga/descarga de los buques, y la estimación del consumo de combustible durante el tiempo de espera en forma de KPI. Esta información permitirá planificar adecuadamente las operaciones de la naviera con el propósito de optimizar los tiempos y consumo de combustible de su flota de buques. Además, la instancia de la arquitectura de Big Data facilitó el desarrollo de un servicio de visualización de los resultados del análisis de datos a través de un *dashboard*. El *dashboard* proporciona la ubicación de los buques en las inmediaciones del puerto y los KPI generados en el procesamiento de datos en tiempo real. De esta manera, los resultados son fáciles de interpretar para tomar las mejores decisiones en la planificación de la naviera.

Finalmente, el uso de la arquitectura de Big Data para IoT utilizando como fuente de datos al EDP demostró obtener beneficios económicos y medio ambientales. La planificación de estrategias para reducir los tiempos de espera en la zona de anclaje del puerto permitirá ahorra al menos €2550,15 por semana en el costo del combustible para los buques de la naviera. Además, los resultados preliminares indican que la liberación de contaminantes en la zona del puerto se reducirá en 14,2 toneladas de CO_2, 0,043 toneladas de SO_2 y 0,288 toneladas de NO_x. Sin el uso de la arquitectura de Big Data para IoT hubiera sido muy difícil detectar los posibles focos de mejora en las operaciones de la flota de buques de la empresa naviera y tomar las decisiones respaldadas con información oportuna que deriven en beneficios económicos y medio ambientales.

Capítulo 7

Conclusiones y futuras líneas de investigación

7.1 Conclusiones finales

En esta tesis se han identificado, analizado y evaluado los principales problemas presentes en la integración de las tecnologías de Big Data con IoT. Posteriormente, una arquitectura integral ha sido presentada basada en estándares de tecnologías y recomendaciones de telecomunicaciones para que se pueda adaptar fácilmente a los dominios de aplicación de IoT. La arquitectura se ha puesto a prueba en tres casos de uso en distintos dominios de aplicación de IoT.

A continuación se exponen las principales conclusiones extraídas de esta investigación.

- La transformación digital de todo lo que nos rodea está progresando de manera acelerada, en gran parte gracias a la materialización de la visión de IoT. Cada día, millones de dispositivos IoT son conectados a Internet con el fin de conectar el mundo físico con el virtual. Como resultado, grandes oleadas de datos son generados por los diferentes entornos de aplicación de IoT. En esta tesis doctoral se ha estudiado a la gestión del gran volumen de datos generados por entornos IoT para

explotar los potenciales beneficios económicos, sociales y tecnológicos de IoT.

- El problema de grandes volúmenes de datos es conocido actualmente como Big Data. Este problema ha sido estudiado ampliamente considerando como fuentes de datos a páginas web, redes sociales, transacciones y registros. IoT como fuente de Big Data es relativamente nueva por lo que aún no se disponen de los suficientes conocimientos para afrontar los problemas de Big Data generados por entornos IoT. IoT como fuente de datos presenta las características de velocidad, volumen y variedad; y las características particulares de heterogeneidad, correlación espacio-tiempo, alto ruido y datos en gran escala.

- En el extenso estudio de la bibliografía relacionada con la temática se ha identificado una amplia variedad de conceptos, definiciones, términos, metodologías y plataformas empleados en el entorno de Big Data. Además, se han analizado las arquitecturas de Big Data consideradas como referencia y sus características. De la misma forma, se ha explicado la visión de IoT, sus características y arquitecturas de referencia para la implementación de sistemas IoT. Finalmente, se han estudiado las tecnologías de *cloud computing* y *fog computing* como tecnologías habilitadoras para la gestión de los grandes volúmenes de datos generados por los entornos IoT; remarcando la importancia del uso de *fog computing* para reducir las limitaciones de *cloud computing*.

- En esta tesis doctoral se ha propuesto una arquitectura para la gestión eficiente de los grandes volúmenes de datos generados por los entornos IoT. El diseño de la arquitectura consideró las recomendaciones propuestas por la ITU-T y el NIST para la implementación de sistemas IoT y para la interoperabilidad de *frameworks* de Big Data; así como, se consideraron las características particulares de los datos generados por IoT. Como resultado, la arquitectura es genérica y flexible para adaptarse a los intereses de los diversos entornos de aplicación de IoT.

- Se ha realizado la descripción de la arquitectura siguiendo la normativa ISO/IEC/IEEE 42010:2011 desde los puntos de vista conceptual, funcional, de procesos y de implementación. Estos puntos de vista cubrieron todos los intereses requeridos para que la arquitectura pueda ser entendida por los arquitectos, ingenieros y técnicos de aplicaciones IoT y sobre todo por el público en general.

- Desde el punto de vista conceptual, la arquitectura describió los roles que intervienen en la gestión del Big Data generado por IoT y sus interacciones. Se identificaron los roles de propietario de datos, proveedor de datos, proveedor de habilitadores tecnológicos de Big Data,

proveedor de servicios y consumidor de servicios. Estos roles están acorde a las recomendaciones del NIST para que la arquitectura pueda ser interoperable con los *frameworks* de Big Data. Además, se hizo una distinción entre proveedor de datos y propietario de datos con el fin de cubrir los intereses sobre privacidad y soberanía de datos.

- Desde el punto de vista funcional, la arquitectura describió los componentes funcionales requeridas y sus características para una eficiente gestión de los datos. Los componentes funcionales fueron agrupados en los dominios de IoT, de información, de aplicaciones y de operación. La implementación de los componentes dependerá del grado de complejidad del sistema, por lo que en algunos casos se pueden omitir algunos componentes funcionales.

- Desde el punto de vista de procesos, la arquitectura describió las interacciones entre los componentes funcionales y las actividades que se realizan para la gestión del Big Data generado por IoT. Los procesos descritos fueron los de recolección de datos, extracción de información, predicción en tiempo real, distribución de modelos predictivos como servicios, y la visualización de analíticas y KPIs. Uno de los procesos novedosos que introduce la arquitectura en relación con el estado del arte es el proceso de distribución de modelos predictivos como servicios, como mecanismo de despliegue de modelos basados en algoritmos IA lo más cercana a la fuente de generación de datos de IoT.

- Desde el punto de vista de implementación, la arquitectura describió los patrones que pueden utilizarse para implementar la arquitectura basados en las tecnologías de *fog computing* y *cloud computing*. La arquitectura propone el patrón *fog-cloud computing* acorde a las nuevas tendencias de extender las capacidades del *cloud*, también llamado *Cloud Continuum*, con el fin de reducir la cantidad de datos que se envían al *cloud*.

- En el primer caso de uso se ha validado la integración de arquitectura de Big Data para IoT con una aplicación de monitorización remota de enfermedades crónicas en el ámbito de la salud. La instancia de la arquitectura ha permitido la gestión eficiente de los datos generados por un sistema para la monitorización de la apnea del sueño en adultos mayores. Como resultado, se han generado servicios innovadores para controlar y guiar a los pacientes durante el tratamiento contra la apnea del sueño.

- La pruebas y los experimentos realizados a través de varios conjuntos de datos generados por el sistema de monitorización de IoT han sido

favorables y satisfactorios, demostrando la eficiencia para gestionar el gran volumen de datos generados por el sistema de IoT. Como consecuencia, se confirma la validez de la instancia de la arquitectura para integrarse con sistemas IoT que se encuentran operando actualmente.

- En el segundo caso de uso se ha validado el uso de la arquitectura de Big Data para IoT en una aplicación para el cuidado de la salud. La instancia de la arquitectura ha permitido la gestión de los datos generados por dispositivos *wearables* IoT utilizando el patrón de conexión *fog-cloud computing*. Como resultado, el uso de la instancia de la arquitectura permitió la generación de modelos de *deep learning* para la detección de caídas en adultos mayores y que estos sean desplegados en nodos *fog*.

- Se han diseñado escenarios de pruebas y experimentos para validar las prestaciones del sistema de detección de caídas basado en la arquitectura de Big Data para IoT. Los resultados mostraron que el sistema de detección propuesto alcanzó la mayor precisión en comparación con sistemas similares. Además, el tiempo de detección de caídas fue significativamente menor al resto de sistema, tardando en detectar una caída en 1,14 segundos. Finalmente, se ha verificado que el uso de la virtualización de contenedores utilizada en los nodos *fog* permite soportar eficientemente el despliegue de los modelos de *deep learning* en dispositivos de limitadas capacidades computacionales como *gateways* IoT.

- En el tercer caso de uso se ha validado la explotación de los datos compartidos en un entorno seguro basado en la arquitectura de referencia IDS en el ámbito industrial del transporte marítimo. La instancia de la arquitectura ha permitido la gestión de los datos compartidos en el IDS portuario y ha permitido el diseño de un sistema de monitorización de la flota de buques de una naviera. Fruto del uso de la instancia de la arquitectura se han generado KPIs relevantes para la planificación de las operaciones de transporte de la naviera.

- Se han realizado pruebas y experimentos en un entorno controlado utilizando varios conjuntos de datos provenientes de la autoridad portuaria y del terminal de contenedores para validar el funcionamiento de la arquitectura. El análisis descriptivo realizado a estos conjuntos de datos permitió generar conocimiento útil para la naviera. Los resultados mostraron que la flota de los buques de la naviera esperan en la zona de anclaje del puerto en promedio 11 horas. Esto ocasiona un consumo de 4,01 toneladas de combustible durante este tiempo de espera. Como consecuencia, la empresa naviera gasta €2550,15 promedio por buque durante el tiempo de espera en la zona de anclaje

y se libera a la atmósfera 14,2 toneladas de CO_2, 0,043 toneladas de SO_2 y 0,288 toneladas de NO_x.

7.2 Líneas futuras de investigación

La investigación presentada en esta tesis doctoral puede ser ampliada y mejorada con el estudio de múltiples líneas de investigación tanto en el dominio de Big Data IoT como en los casos de uso estudiados.

A continuación, las posibles líneas futuras de investigación se señalan para ser consideradas en un futuro.

- La arquitectura integral planteada en esta tesis doctoral fue diseñada lo más genérica posible para que se pueda adaptar fácilmente a los dominios de aplicación de IoT. La arquitectura podría ser analiza en el estudio de otros dominios de aplicación de IoT, como por ejemplo la agricultura inteligente, medición inteligente (consumo de electricidad, agua, y gas), o gestión de tráfico inteligente.
- Una importante mejora a la arquitectura sería la integración con la tecnología *Blockchain* con el fin de solucionar problemas de veracidad de los datos; así como, brindar transparencia y trazabilidad a los proveedores de datos.
- La visualización de los resultados de análisis del Big Data podría incluir diagramas interactivos donde el usuario pueda generar reportes de una manera fácil e intuitiva. Además, otras perspectivas de visualización deben ser evaluadas como por ejemplo el uso de realidad virtual o realidad aumentada para visualización de los KPI.
- Una de las limitaciones del proceso de distribución de los servicios de predicción en nodos *fog* fue la escalabilidad. La colaboración de varios nodos *fog* para brindar uno o varios servicios de predicción permitiría mejorar los tiempos de inferencia para reducir esta brecha de escalabilidad.
- Si bien la tecnología de virtualización de contenedores demostró ser adecuada para ser empleada en el borde de la red, esta ha sido diseñada para entornos *Cloud Computing*. Bajo esta consideración, la plataforma Balena[1] realiza una optimización de contenedores para que sean adecuados en el despliegue en *Fog Computing*. El uso de Balena podría brindar mejores prestaciones permitiendo distribuir modelos

[1] https://www.balena.io/

de predicción más complejos, como por ejemplo modelos de visión artificial en el borde de la red.

- En el área de la salud, la arquitectura IDS puede ser estudiada con el fin de generar un mercado de datos donde los usuarios puedan comercializar los datos recolectados por sus dispositivos *wearables*. A la misma vez, los usuarios controlarían en todo momento el uso que se le den a sus datos.

- Nuevos escenarios de aplicación en la salud resultarían interesantes y retadores estudiarlos. Por ejemplo, la arquitectura puede ser aplicada para la detección temprana de arritmias cardíacas, empleando el patrón de implementación basado en *Fog-Cloud Computing*. Al igual que, la consideración del uso de los sistemas de salud basados en el estándar HR7.

- En el área de transporte, otros agentes portuarios pueden ser añadidos al espacio de datos portuario para compartir datos. Por ejemplo, la compartición de datos de un agente de transporte terrestre permitiría extraer los KPI relacionados con el tráfico de camiones tanto de salida como de entrada al puerto o predecir el tiempo de permanencia de un contenedor en la terminal de contenedores.

- Otra oportunidad de mejora es aplicar el patrón de implementación basado en *Fog-Cloud Computing* en el entorno portuario con el fin de desplegar servicios de predicción de fallas de los equipos portuarios para elaborar un mantenimiento preventivo.

- Una oportunidad de mejora en el sistema de monitorización de la flota de buques de la naviera es el uso de análisis predictivo basados en los datos AIS. De la misma forma, es interesante la utilización de imágenes de satélites para la detección de buques cercanos al puerto y predecir situaciones anómalas aprovechando la instancia de arquitectura ya implementada en el sistema de monitorización de la flota de buques de la naviera.

Bibliografía

[1] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the internet of things," *Cluster of European Research Projects on the Internet of Things, European Commision*, vol. 3, no. 3, pp. 34–36, 2010.

[2] J. Manyika, *The Internet of Things: Mapping the value beyond the hype*. McKinsey Global Institute, 2015.

[3] Cisco Networking Visual, "Cisco Visual Networking Index: Forecast and Trend, 2017-2022 White Paper," Tech. Rep., 2019.

[4] Cisco Cloud, "Cisco global cloud index: Forecast and methodology, 2017-2022 white paper," Tech. Rep., 2018.

[5] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, jan 2014.

[6] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, jan 2015.

[7] K. Papadokostaki, G. Mastorakis, S. Panagiotakis, C. X. Mavromoustakis, C. Dobre, and J. M. Batalla, "Handling big data in the era of internet of things (iot)," in *Studies in Big Data*. Springer International Publishing, nov 2017, pp. 3–22.

[8] A. Cravero, "Big data architectures and the internet of things: A systematic mapping study," *IEEE Latin America Transactions*, vol. 16, no. 4, pp. 1219–1226, April 2018.

[9] D. Sarabia-Jácome, R. Usach, C. Palau, and M. Esteve, "Highly-efficient fog-based deep learning aal fall detection system," *Internet of Things*, p. 100185, feb 2020. [Online]. Available: https://doi.org/10.1016%2Fj.iot.2020.100185

[10] 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.

[11] D. Yacchirema, D. Sarabia-Jácome, C. E. Palau, and M. Esteve, "System for monitoring and supporting the treatment of sleep apnea using iot and big data," *Pervasive and Mobile Computing*, vol. 50, pp. 25–40, 2018.

[12] D. C. Yacchirema, D. Sarabia-Jácome, C. E. Palau, and M. Esteve, "A smart system for sleep monitoring by integrating iot with big data analytics," *IEEE Access*, vol. 6, pp. 35 988–36 001, 2018.

[13] 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," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 351–356.

[14] A. Belsa, D. Sarabia-Jacome, C. E. Palau, and M. Esteve, "Flow-based programming interoperability solution for iot platform applications," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 304–309.

[15] D. Sarabia-Jácome, I. Lacalle, C. E. Palau, and M. Esteve, "Efficient deployment of predictive analytics in edge gateways: Fall detection scenario," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, April 2019, pp. 41–46.

[16] D. Sarabia-Jacome, I. Lacalle, C. E. Palau, and M. Esteve, "Enabling industrial data space architecture for seaport scenario," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, April 2019, pp. 101–106.

[17] D. Sarabia-Jácome, R. Gonzalez-Usach, and C. E. Palau, "IoT Big Data Architectures, Approaches, and Challenges: A Fog-Cloud Approach," in *Handbook of Research on Big Data and the IoT*. IGI Global, 2019, pp. 125–148.

[18] D. Sarabia-Jácome and C. E. Palau, "Data from vessels transportation around the Spain mediterranean area," 2020. [Online]. Available: http://dx.doi.org/10.21227/2580-e466

[19] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[20] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[21] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, may 2016. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2016.01.010

[22] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of internet of things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, vol. 5. IEEE, aug 2010, pp. V5–484–V5–487.

[23] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *2012 10th International Conference on Frontiers of Information Technology*. IEEE, dec 2012, pp. 257–260.

[24] FI-WARE Consortium, "FIWARE: The open source platraform for our smart digital future," Accessed: 2018-12-15. [Online]. Available: https://www.fiware.org/

[25] Standardisation AIOTI WG03–loT, "High Level Architecture," Tech. Rep., 2017.

[26] S.-W. Lin, B. Miller, J. Durand, G. Bleakley, A. Chigani, R. Martin, B. Murphy, and M. Crawford, "The industrial internet of things volume g1: reference architecture," *Industrial Internet Consortium*, pp. 10–46, 2017.

[27] M. A. Pisching, M. A. Pessoa, F. Junqueira, D. J. dos Santos Filho, and P. E. Miyagi, "An architecture based on rami 4.0 to discover equipment to process operations required by products," *Computers & Industrial Engineering*, vol. 125, pp. 574–591, 2018.

[28] M. Hankel and B. Rexroth, "The reference architectural model industrie 4.0 (rami 4.0)," *ZVEI, April*, vol. 410, 2015.

[29] K. Schweichhart, "Reference architectural model industrie 4.0 (rami 4.0)," *An Introduction. Available online: https://www. plattform-i40. de I*, vol. 40, 2016.

[30] Y. Wang, T. Towara, and R. Anderl, "Topological approach for mapping technologies in reference architectural model industrie 4.0 (rami 4.0)," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2017.

[31] K. Al-Gumaei, K. Schuba, A. Friesen, S. Heymann, C. Pieper, F. Pethig, and S. Schriegel, "A survey of internet of things and big data integrated solutions for industrie 4.0," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 1417–1424.

[32] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[33] M. Haghi, K. Thurow, and R. Stoll, "Wearable devices in medical internet of things: scientific research and commercially available devices," *Healthcare informatics research*, vol. 23, no. 1, pp. 4–15, 2017.

[34] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.

[35] D. V. Dimitrov, "Medical internet of things and big data in healthcare," *Healthcare informatics research*, vol. 22, no. 3, pp. 156–163, 2016.

[36] K. N. Mishra and C. Chakraborty, "A novel approach towards using big data and iot for improving the efficiency of m-health systems," in *Advanced Computational Intelligence Techniques for Virtual Reality in Healthcare*. Springer, 2020, pp. 123–139.

[37] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," *Health Information Science and Systems*, vol. 2, no. 1, pp. 1–10, feb 2014.

[38] M. Chen, J. Yang, J. Zhou, Y. Hao, J. Zhang, and C.-H. Youn, "5g-smart diabetes: Toward personalized diabetes diagnosis with healthcare big data clouds," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 16–23, apr 2018.

[39] "Transforming transport," Accessed: 2019-05-21. [Online]. Available: https://transformingtransport.eu/

[40] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions*

on Intelligent Transportation Systems, vol. 20, no. 1, pp. 383–398, 2018.

[41] D. Štepec, T. Martinčič, and D. Skočaj, "Automated system for ship detection from medium resolution satellite optical imagery," in *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE, 2019, pp. 1–10.

[42] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0 – an introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[43] "INTER-IoT - Interoperability Internet of Things," Accessed: 2018-02-16. [Online]. Available: https://inter-iot.eu/

[44] I. Schirmer, P. Drews, S. Saxe, U. Baldauf, and J. Tesse, "Extending enterprise architectures for adopting the internet of things–lessons learned from the smartport projects in hamburg," in *International Conference on Business Information Systems*. Springer, 2016, pp. 169–180.

[45] A. Belfkih, C. Duvallet, and B. Sadeg, "The internet of things for smart ports: Application to the port of le havre," in *International Conference on Intelligent Platform for Smart Port (IPaSPort 2017)*, 2017.

[46] P. Fernández, J. M. Santana, S. Ortega, A. Trujillo, J. P. Suárez, C. Domínguez, J. Santana, and A. Sánchez, "Smartport: A platform for sensor data monitoring in a seaport based on FIWARE," *Sensors (Switzerland)*, vol. 16, no. 3, pp. 1–24, 2016.

[47] "Pixel-ports," Accessed: 2017-04-19. [Online]. Available: https://pixel-ports.eu/

[48] ITU-T, "Big Data - Cloud computing based requirements and capabilities. T-REC-Y.3600," 2015, Accessed: 2018-05-18. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3600

[49] NIST Big Data Public Working Group, "NIST big data interoperability framework: volume 1, definitions, version 2," NIST, Tech. Rep., jun 2018.

[50] "What Is Big Data? - Gartner IT Glossary - Big Data," Accessed: 2018-04-10. [Online]. Available: https://www.gartner.com/it-glossary/big-data

[51] J. Gantz and D. Reinsel, "Extracting Value from Chaos State of the Universe: An Executive Summary," *IDC iView*, no. June, pp. 1–12, 2011. [Online]. Available: http://idcdocserv.com/1142

[52] Y. Demchenko, C. De Laat, and P. Membrey, "Defining architecture components of the big data ecosystem," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2014, pp. 104–112.

[53] P. Wongthongtham, J. Kaur, V. Potdar, and A. Das, "Big data challenges for the internet of things (IoT) paradigm," in *Connected Environments for the Internet of Things*. Springer International Publishing, 2017, pp. 41–62.

[54] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the internet of things: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[55] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: a survey," *IEEE Internet of things journal*, vol. 3, no. 1, pp. 70–95, 2015.

[56] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[57] M. Strohbach, H. Ziekow, V. Gazis, and N. Akiva, "Towards a big data analytics framework for iot and smart city applications," in *Modeling and processing for next-generation big-data technologies*. Springer, 2015, pp. 257–282.

[58] B. Gaunitz, M. Roth, and B. Franczyk, "Dynamic and scalable real-time analytics in logistics combining apache storm with complex event processing for enabling new business models in logistics," in *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. IEEE, 2015, pp. 289–294.

[59] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, sep 2013.

[60] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marín-Torder, "Towards a comprehensive data lifecycle model for big data environments," in *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*. ACM, 2016, pp. 100–106.

[61] C. Cecchinel, M. Jimenez, S. Mosser, and M. Riveill, "An Architecture to Support the Collection of Big Data in the Internet of Things," in

2014 IEEE World Congress on Services. IEEE, jun 2014, pp. 442–449.

[62] D. Tracey and C. Sreenan, "A holistic architecture for the internet of things, sensing services and big data," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. IEEE, 2013, pp. 546–553.

[63] B. Cheng, S. Longo, F. Cirillo, M. Bauer, and E. Kovacs, "Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander," *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015*, pp. 592–599, 2015.

[64] C. Costa and M. Y. Santos, "BASIS: A big data architecture for smart cities," in *2016 SAI Computing Conference (SAI)*. IEEE, jul 2016, pp. 1247–1256.

[65] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce," *Information Fusion*, vol. 42, pp. 51–61, 2018.

[66] M. D. de Assuncao, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications*, vol. 103, pp. 1–17, 2018.

[67] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed stream computing platform," in *2010 IEEE International Conference on Data Mining Workshops*, Dec 2010, pp. 170–177.

[68] "Apache Hive TM," accessed: 2017-04-19. [Online]. Available: http://hive.apache.org/

[69] "Ambari -," Accessed: 2017-04-19. [Online]. Available: http://ambari.apache.org/

[70] "Apache ZooKeeper - Home," Accessed: 2017-04-19. [Online]. Available: http://zookeeper.apache.org/

[71] "Apache Mahout: Scalable machine learning and data mining," Accessed: 2017-05-10. [Online]. Available: http://mahout.apache.org/

[72] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, 2010.

[73] "Apache Spark - Lightning-Fast Cluster Computing," Accessed: 2017-03-27. [Online]. Available: http://spark.apache.org/

[74] "Apache Flink: Introduction to Apache Flink," Accessed: 2017-04-28. [Online]. Available: https://flink.apache.org/introduction.html

[75] N. Marz and J. Warren, *Big data: principles and best practices of scalable realtime data systems.* Shelter Island, NY : Manning, cop. 2015, 2015. [Online]. Available: http://nathanmarz.com/about/

[76] H. Mohanty, "Big data: an introduction," in *Big Data.* Springer, 2015, pp. 1–28.

[77] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in internet of things," *Computer Networks*, vol. 129, pp. 459–471, 2017.

[78] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive iot data and open research issues," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.

[79] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[80] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[81] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. T. Yang, "Data mining for internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[82] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[83] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[84] "Caffe | Deep Learning Framework," Accessed: 2019-04-10. [Online]. Available: http://caffe.berkeleyvision.org/

[85] Yahoo, "Distributed deep learning on Hadoop and Spark clusters. ," 2017, Accessed: 2017-06-15. [Online]. Available: https://github.com/yahoo/CaffeOnSpark

[86] "TensorFlow," Accessed: 2017-05-09. [Online]. Available: https://www.tensorflow.org/

[87] "Open Sourcing TensorFlowOnSpark," Accessed: 2017-05-09. [Online]. Available: http://yahoohadoop.tumblr.com/post/157196317141/open-sourcing-tensorflowonspark-distributed-deep

[88] "About BigML.com," accessed: 2017-05-10. [Online]. Available: https://bigml.com/about

[89] "Home - Keras Documentation," Accessed: 2019-04-10. [Online]. Available: https://keras.io/

[90] "Apache Mahout: Scalable machine learning and data mining," Accessed: 2017-05-10. [Online]. Available: http://mahout.apache.org/

[91] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," Accessed: 2017-05-11. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/index.html

[92] "scikit-learn: machine learning in Python scikit-learn 0.20.3 documentation," Accessed: 2019-04-10. [Online]. Available: https://scikit-learn.org/stable/

[93] NIST Big Data Public Working Group, "NIST big data interoperability framework: volume 6, reference architecture, version 2," NIST, Tech. Rep., jun 2018.

[94] B. D. V. Association *et al.*, "Big Data Value Strategic Research and Innovation Agenda," 2017. [Online]. Available: http://www.bdva.eu/node/874

[95] J. Kreps, "Questioning the Lambda Architecture - O'Reilly Media," 2014. [Online]. Available: https://www.oreilly.com/ideas/questioning-the-lambda-architecture

[96] S. Uesugi, "Kappa Architecture - Where Every Thing Is A Stream," 2014, Accessed: 2018-05-04. [Online]. Available: http://milinda.pathirage.org/kappa-architecture.com/

[97] NIST Big Data Public Working Group, "NIST big data interoperability framework: Volume 5, architectures white paper survey," NIST, Tech. Rep., oct 2015.

[98] D. Chappette, "Oracle big data and analytics reference architecture," 2013.

[99] IBM, "IBM Big Data & Analytics Reference Architecture V1," 2014, Accessed: 2017-07-19. [Online]. Available: https://www.ibm.com

[100] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the Internet of Things using Big Data analytics," *Computer Networks*, vol. 101, pp. 63–80, jun 2016.

[101] P. Ta-Shma, A. Akbar, G. Gerson-Golan, G. Hadash, F. Carrez, and K. Moessner, "An ingestion and analytics architecture for iot applied to smart city use cases," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 765–774, April 2018.

[102] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of Things and Big Data Analytics for Smart and Connected Communities," *IEEE Access*, vol. 4, no. 4, pp. 766–773, 2016.

[103] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "Alljoyn lambda: An architecture for the management of smart environments in iot," in *2014 International Conference on Smart Computing Workshops*. IEEE, 2014, pp. 9–14.

[104] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[105] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2013, pp. 233–240.

[106] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, jan 2018.

[107] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a Service and Big Data," *Proceedings of the International Conference on Advances in Cloud Computing (ACC-2012)*, vol. 25, pp. 21–29, 2012.

[108] M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, "Cloud of things: Integrating internet of things and cloud computing and the issues involved," in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*, no. January. IEEE, jan 2014, pp. 414–419.

[109] Amazon, "AWS IoT Services Overview - Amazon Web Services," 2018, Accessed: 2018-05-15. [Online]. Available: https://aws.amazon.com/iot/

[110] Google, "Google Cloud IoT," 2017, accessed: 2018-05-15. [Online]. Available: https://cloud.google.com/iot-core/https://cloud.google.com/solutions/iot/?hl=es

[111] Microsoft, "Azure IoT Hub | Microsoft Azure," 2016, Accessed: 2018-05-15. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-hub/

[112] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, mar 2016.

[113] KAA, "Kaa IoT Development Platform overview," 2018, Accessed: 2018-05-18. [Online]. Available: https://www.kaaproject.org/overview/

[114] European Commission, "The Internet of Things | Digital Single Market," 2017, Accessed: 2018-05-12. [Online]. Available: https://ec.europa.eu/digital-single-market/en/research-innovation-iot

[115] S. Yang, "Iot stream processing and analytics in the fog," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 21–27, 2017.

[116] V. A. Memos, K. E. Psannis, Y. Ishibashi, B.-G. Kim, and B. Gupta, "An efficient algorithm for media-based surveillance system (EAM-SuS) in IoT smart city framework," *Future Generation Computer Systems*, vol. 83, pp. 619–628, jun 2018.

[117] G. Kokkonis, K. E. Psannis, M. Roumeliotis, and Y. Ishibashi, "Efficient algorithm for transferring a real-time HEVC stream with haptic data through the internet," *Journal of Real-Time Image Processing*, vol. 12, no. 2, pp. 343–355, may 2015.

[118] R. Buyya, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojicic, C. Varela, R. Bahsoon, M. D. D. Assuncao, S. N. Srirama, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, H. Shen, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, and L. M. Vaquero, "A manifesto for future generation cloud computing," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–38, nov 2018.

[119] European Telecommunications Standards Institute (ETSI), "Executive Briefing Mobile Edge Computing (MEC) Initiative," pp. 1–3, 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mec%20executive%20brief%20v1%2028-09-14.pdf

[120] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[121] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, no. 4, pp. 14–23, 2009.

[122] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. New York, New York, USA: ACM Press, 2012, p. 13.

[123] A. Carrega and M. Repetto, "A network-centric architecture for building the cloud continuum," in *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2017, pp. 701–705.

[124] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3, pp. 134–155, 2018.

[125] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–21, 2019.

[126] R. Silipo, I. Adae, A. Hart, and M. Berthold, "Seven Techniques for Dimensionality Reduction," pp. 1–21, 2014. [Online]. Available: https://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html

[127] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, apr 2015.

[128] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, jan 2018.

[129] The OpenFog Consortium, "OpenFog Architecture Overview," Tech. Rep. February, 2016. [Online]. Available: www.OpenFogConsortium.org

[130] J. Green, "Iot reference model june, 2014," Jun. 2014. [Online]. Available: https://www.iotwf.com/resources

[131] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*, pp. 171–172, mar 2015.

[132] R. Morabito, R. Petrolo, V. Loscrì, and N. Mitton, "LEGIoT: A lightweight edge gateway for the internet of things," *Future Generation Computer Systems*, vol. 81, pp. 1–15, apr 2018.

[133] Sphinx, "ParaDrop - Enabling Edge Computing at the Extreme Edge paradrop 0.11.2 documentation," 2018, Accessed: 2018-05-23. [Online]. Available: http://paradrop.readthedocs.io/en/latest/index.html

[134] P. Liu, L. Hartung, and S. Banerjee, "Lightweight multitenancy at the network's extreme edge," *Computer*, vol. 50, no. 10, pp. 50–57, 2017.

[135] "Apache edgent," Accessed: 2019-04-10. [Online]. Available: http://edgent.apache.org/

[136] F. Pisani, J. R. Brunetta, V. M. do Rosario, and E. Borin, "Beyond the fog: Bringing cross-platform code execution to constrained IoT devices," *Proceedings - 29th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2017*, pp. 17–24, oct 2017.

[137] ITU-T, "Next-Generation Networks - Common requirements of the Internet of things. T-REC-Y.2066," 2014, Accessed: 2018-05-18. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2066-201406-I

[138] ITU-T, "Internet of things and smart cities - Specific requirements and capabilities of the Internet of things for big data. T-REC-Y.4114," 2017, Accessed: 2018-05-18. [Online]. Available: https://www.itu.int/rec/T-REC-Y.4114/en

[139] ISO/IEC, "Calidad del producto software," Accessed: 2019-04-16. [Online]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010

[140] I. May, "Systems and software engineering–architecture description," Technical Report. ISO/IEC/IEEE 42010, Tech. Rep., 2011.

[141] "Chronic respiratory diseases," Accessed: 2017-09-29. [Online]. Available: http://www.who.int/respiratory/copd/en/

[142] M. Glasser, N. Bailey, A. McMillan, E. Goff, and M. Morrell, "Sleep apnoea in older people," *Breathe*, vol. 7, no. 3, pp. 248–256, 2011.

[143] A. S. Karunajeewa, U. R. Abeyratne, and C. Hukins, "Multi-feature snore sound analysis in obstructive sleep apnea–hypopnea syndrome," *Physiological measurement*, vol. 32, no. 1, p. 83, 2010.

[144] M. Bsoul, H. Minn, and L. Tamil, "Apnea medassist: real-time sleep apnea monitor using single-lead ecg," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 3, pp. 416–427, 2010.

[145] G. Sannino, I. De Falco, and G. De Pietro, "Monitoring obstructive sleep apnea by means of a real-time mobile system based on the automatic extraction of sets of rules through differential evolution," *Journal of biomedical informatics*, vol. 49, pp. 84–100, 2014.

[146] M. Rofouei, M. Sinclair, R. Bittner, T. Blank, N. Saw, G. DeJean, and J. Heffron, "A non-invasive wearable neck-cuff system for real-time sleep monitoring," in *2011 international conference on body sensor networks*. IEEE, 2011, pp. 156–161.

[147] Y. Nam, Y. Kim, and J. Lee, "Sleep monitoring based on a tri-axial accelerometer and a pressure sensor," *Sensors*, vol. 16, no. 5, p. 750, 2016.

[148] X. Zhu, X. Zhou, W. Chen, K.-I. Kitamura, and T. Nemoto, "Estimation of sleep quality of residents in nursing homes using an internet-based automatic monitoring system," in *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*. IEEE, 2014, pp. 659–665.

[149] H. Nakano, K. Hirayama, Y. Sadamitsu, A. Toshimitsu, H. Fujita, S. Shin, and T. Tanigawa, "Monitoring sound to quantify snoring and sleep apnea severity using a smartphone: proof of concept," *Journal of Clinical Sleep Medicine*, vol. 10, no. 01, pp. 73–78, 2014.

[150] F. M. F. Lobato, D. C. O. de Resende, R. P. do Nascimento, A. L. C. Siqueira, A. F. L. Jacob, and Á. L. de Santana, "Multimodal low-invasive system for sleep quality monitoring and improvement," in *Beyond the Internet of Things*. Springer, 2017, pp. 223–242.

[151] J. Kestelyn, "Benchmarking Apache Parquet: The Allstate Experience," 2016, Accessed: 2016-04-22. [Online]. Available: https://blog.cloudera.com/blog/2016/04/benchmarking-apache-parquet-the-allstate-experience/

[152] "Hdfs architecture guide," Accessed: 2017-04-19. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[153] "Apache cassandra," Accessed: 2017-04-19. [Online]. Available: http://cassandra.apache.org/

[154] Red Valenciana de Vigilancia y Control de la Contaminación Atmosférica, "Índices de la calidad del aire," 2017, Accessed: 2017-10-15. [Online]. Available: https://www.cma.gva.es/cidam/emedio/atmosfera/jsp/pde.jsp?PDE.CONT=1466

[155] S. Din, H. Ghayvat, A. Paul, A. Ahmad, M. M. Rathore, and I. Shafi, "An architecture to analyze big data in the Internet of Things," in *2015 9th International Conference on Sensing Technology (ICST)*, vol. 2016-March. IEEE, dec 2015, pp. 677–682.

[156] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, and O. Fratu, "Big data, internet of things and cloud convergence–an architecture for secure e-health applications," *Journal of medical systems*, vol. 39, no. 11, p. 141, 2015.

[157] S. Coban, M. O. Gokalp, E. Gokalp, P. E. Eren, and A. Kocyigit, "[WiP] Predictive Maintenance in Healthcare Services with Big Data Technologies," in *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, nov 2018, pp. 93–98.

[158] C. Liu, C. Zhang, F. Chen, and C. Zhao, "Towards IPv6-based Architecture for Big Data Processing of Community Medical Internet of Things," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, jun 2018, pp. 1333–1338.

[159] "Falls," Accessed: 2018-09-20. [Online]. Available: http://www.who.int/news-room/fact-sheets/detail/falls

[160] N. Zerrouki, F. Harrou, Y. Sun, and A. Houacine, "Vision-based human action classification using adaptive boosting algorithm," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5115–5121, 2018.

[161] L. Panahi and V. Ghods, "Human fall detection using machine vision techniques on rgb–d images," *Biomedical Signal Processing and Control*, vol. 44, pp. 146–153, 2018.

[162] N. Lu, Y. Wu, L. Feng, and J. Song, "Deep learning for fall detection: Three-dimensional cnn combined with lstm on video kinematic data," *IEEE journal of biomedical and health informatics*, vol. 23, no. 1, pp. 314–323, 2018.

[163] Y. Li, K. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 5, pp. 1291–1301, 2012.

[164] C. Taramasco, T. Rodenas, F. Martinez, P. Fuentes, R. Munoz, R. Olivares, V. H. C. De Albuquerque, and J. Demongeot, "A novel monitoring system for fall detection in older people," *IEEE Access*, vol. 6, pp. 43 563–43 574, 2018.

[165] X. Fan, H. Zhang, C. Leung, and Z. Shen, "Fall detection with unobtrusive infrared array sensors," in *Multisensor Fusion and Integration for Intelligent Systems*. Springer, 2017, pp. 253–267.

[166] C. Wang, W. Lu, M. R. Narayanan, D. C. W. Chang, S. R. Lord, S. J. Redmond, and N. H. Lovell, "Low-power fall detector using triaxial accelerometry and barometric pressure sensing," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2302–2311, 2016.

[167] T. Theodoridis, V. Solachidis, N. Vretos, and P. Daras, "Human fall detection from acceleration measurements using a recurrent neural network," in *Precision Medicine Powered by pHealth and Connected Health*. Springer, 2018, pp. 145–149.

[168] S. Khojasteh, J. Villar, C. Chira, V. González, and E. De La Cal, "Improving fall detection using an on-wrist wearable accelerometer," *Sensors*, vol. 18, no. 5, p. 1350, 2018.

[169] A. Ngu, Y. Wu, H. Zare, A. Polican, B. Yarbrough, and L. Yao, "Fall detection using smartwatch sensor data with accessor architecture," in *International Conference on Smart Health*. Springer, 2017, pp. 81–93.

[170] D. Yacchirema, J. S. de Puga, C. Palau, and M. Esteve, "Fall detection system for elderly people using IoT and big data," *Procedia Computer Science*, vol. 130, pp. 603–610, 2018.

[171] V. Carletti, A. Greco, A. Saggese, and M. Vento, "A smartphone-based system for detecting falls using anomaly detection," in *Image Analysis and Processing - ICIAP 2017*. Cham: Springer International Publishing, 2017, pp. 490–499.

[172] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[173] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition

system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2017.

[174] K. Jaiswal, S. Sobhanayak, A. K. Turuk, S. L. Bibhudatta, B. K. Mohanta, and D. Jena, "An iot-cloud based smart healthcare monitoring system using container based virtual environment in edge device," in *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*. IEEE, 2018, pp. 1–7.

[175] M. Al-Rakhami, M. Alsahli, M. M. Hassan, A. Alamri, A. Guerrieri, and G. Fortino, "Cost efficient edge intelligence framework using docker containers," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 800–807.

[176] "Eclipse mosquitto tm - an open source mqtt broker," Accessed: 2018-08-13. [Online]. Available: https://mosquitto.org/

[177] "Mongodb," Accessed: 2018-08-13. [Online]. Available: https://www.mongodb.com/es/

[178] "Portainer," Accessed: 2018-09-12. [Online]. Available: https://www.portainer.io/

[179] A. Sucerquia, J. López, and J. Vargas-Bonilla, "Sisfall: A fall and movement dataset," *Sensors*, vol. 17, no. 1, p. 198, 2017.

[180] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *arXiv preprint arXiv:1905.03554*, 2019.

[181] U. of Stanford, "CS231n Convolutional Neural Networks," 2019, Accessed: 2019-08-15. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[182] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[183] "Redes neuronales," Accessed: 2019-03-28. [Online]. Available: https://ml4a.github.io/ml4a/es/neural_networks/

[184] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[185] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[186] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[187] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[188] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[189] "Hivemq - blog posts with category "mqtt essentials"," accessed: 2018-10-13. [Online]. Available: https://www.hivemq.com/tags/mqtt-essentials/

[190] F. Kaup, P. Gottschling, and D. Hausheer, "Powerpi: Measuring and modeling the power consumption of the raspberry pi," in *39th Annual IEEE Conference on Local Computer Networks*. IEEE, 2014, pp. 236–243.

[191] D. Sarabia-Jácome, A. Rego, S. Sendra, and J. Lloret, "Energy consumption in software defined networks to provide service for mobile users," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1453–1458.

[192] Ø. Buhaug, J. J. Corbett, Ø. Endresen, V. Eyring, J. Faber, S. Hanayama, D. S. Lee, D. Lee, H. Lindstad, A. Markowska *et al.*, "Second imo ghg study 2009," *International Maritime Organization (IMO) London, UK*, vol. 20, 2009.

[193] "Maritime transport of goods - quarterly data - Statistics Explained," Accessed: 2019-06-04. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php/Maritime_transport_of_goods_-_quarterly_data#EU_ports_activity

[194] K. Douaioui, M. Fri, C. Mabrouki, and E. A. Semma, "Smart port: Design and perspectives," in *2018 4th International Conference on Logistics Operations Management (GOL)*. IEEE, apr 2018, pp. 1–6.

[195] "International data spaces association," Accessed: 2018-11-15. [Online]. Available: https://www.internationaldataspaces.org/

[196] B. Otto, S. Lohmann, S. Auer, J. Cirullies, A. Eitel, T. Ernst, C. Haas, M. Huber, J. Jurjens, C. Lange, C. Mader, N. Menz, R. Nagel, H. Pettenpohl, J. Pullmann, C. Quix, J. Schon, D. Schulz, J. Schutte, M. Spiekermann, and S. Wenzel, "Reference architecture model for the industrial data space," p. 91, 2017.

[197] Á. Alonso, A. Pozo, J. Cantera, F. de la Vega, and J. Hierro, "Industrial data space architecture implementation using FIWARE," *Sensors*, vol. 18, no. 7, p. 2226, 2018.

[198] S. Matwin, "Big data meets big water: Analytics of the ais ship tracking data," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sep. 2016, pp. 1–1.

[199] P. De Meyer, F. Maes, and A. Volckaert, "Emissions from international shipping in the belgian part of the north sea and the belgian seaports," *Atmospheric Environment*, vol. 42, no. 1, pp. 196–206, 2008.

[200] L. Schrooten, I. De Vlieger, L. I. Panis, K. Styns, and R. Torfs, "Inventory and forecasting of maritime emissions in the belgian sea territory, an activity-based emission model," *Atmospheric Environment*, vol. 42, no. 4, pp. 667–676, 2008.

[201] G. Pallotta, M. Vespe, and K. Bryan, "Vessel pattern knowledge discovery from ais data: A framework for anomaly detection and route prediction," *Entropy*, vol. 15, no. 6, pp. 2218–2245, 2013.

[202] G. Pallotta, M. Vespe, and K. Bryan, "Traffic route extraction and anomaly detection from ais data," in *International COST MOVE Workshop on Moving Objects at Sea, Brest, France*, 2013.

[203] H. Wang, O. L. Osen, G. Li, W. Li, H.-N. Dai, and W. Zeng, "Big data and industrial internet of things for the maritime industry in northwestern norway," in *TENCON 2015 - 2015 IEEE Region 10 Conference*, IEEE. IEEE, nov 2015, pp. 1–5.

[204] X. Sun, Z. Tian, R. Malekian, and Z. Li, "Estimation of vessel emissions inventory in qingdao port based on big data analysis," *Symmetry*, vol. 10, no. 10, p. 452, 2018.

[205] "Exploiting Oceans of Data for Maritime Apps," accessed: 2019-04-17. [Online]. Available: https://www.bigdataocean.eu/

[206] I. Lytra, M.-E. Vidal, F. Orlandi, and J. Attard, "A big data architecture for managing oceans of data and maritime applications," in *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2017, pp. 1216–1226.

[207] G. Vouros, A. Vlachou, G. Santipantakis, C. Doulkeridis, N. Pelekis, H. Georgiou, Y. Theodoridis, K. Patroumpas, E. Alevizos, A. Artikis, C. Claramunt, C. Ray, D. Scarlatti, G. Fuchs, G. Andrienko, N. Andrienko, M. Mock, E. Camossi, A.-L. Jousselme, and J. M. C. Garcia, "Big data analytics for time critical mobility forecasting: Recent progress and research challenges," *Published in Proceedings of the 21st International Conference on Extending Database Technology (EDBT)*, pp. 612–623, 2018.

[208] G. M. Santipantakis, A. Glenis, K. Patroumpas, A. Vlachou, C. Doulkeridis, G. A. Vouros, N. Pelekis, and Y. Theodoridis, "SPARTAN: Semantic integration of big spatio-temporal data from streaming and archival sources," *Future Generation Computer Systems*, jul 2018.

[209] P. Fernández, J. P. Suárez, A. Trujillo, C. Domínguez, and J. M. Santana, "3d-monitoring big geo data on a seaport infrastructure based on fiware," *Journal of Geographical Systems*, pp. 1–19, 2018.

[210] J. Muñuzuri, L. Onieva, P. Cortés, and J. Guadix, "Using iot data and applications to improve port-based intermodal supply chains," *Computers & Industrial Engineering*, 2019.

[211] V. Carlan, C. Sys, and T. Vanelslander, "How port community systems can contribute to port competitiveness: Developing a cost–benefit framework," *Research in Transportation Business & Management*, vol. 19, pp. 51–64, 2016.

[212] M. Lind, T. Andersen, M. Bergmann, R. T. Watson, S. Haraldson, M. Karlsson, M. Michaelides, J. Gimenez, R. Ward, N. Bjørn-Andersen *et al.*, "The maturity level framework for portcdm," 2018.

[213] A. Gharehgozli, H. de Vries, and S. Decrauw, "The role of standardisation in european intermodal transportation," *Maritime Business Review*, 2019.

[214] "International data spaces association," Accessed: 2018-12-16. [Online]. Available: https://www.internationaldataspaces.org/thyssenkrupp-implements-first-use-case-for-industrial-data-space

[215] "International data spaces association," accessed: 2018-12-16. [Online]. Available: https://www.internationaldataspaces.org/success-stories/{#}advaneo/

[216] Ø. J. Rødseth and A. J. Berre, "From digital twin to maritime data space: Transparent ownership and use of ship information."

[217] "Node-red low-code programming for event-driven applications," Accessed: 2018-03-27. [Online]. Available: https://nodered.org/

[218] "AIS data sharing and vessel tracking by AISHub," Accessed: 2019-03-19. [Online]. Available: https://www.aishub.net/

[219] D. Chen, X. Wang, Y. Li, J. Lang, Y. Zhou, X. Guo, and Y. Zhao, "High-spatiotemporal-resolution ship emission inventory of china based on ais data in 2014," *Science of the Total Environment*, vol. 609, pp. 776–787, 2017.

[220] L. Zhang, Q. Meng, and T. F. Fwa, "Big ais data based spatial-temporal analyses of ship traffic in singapore port waters," *Transportation Research Part E: Logistics and Transportation Review*, vol. 129, pp. 287–304, 2019.