Propuesta de Uso del Paradigma de Coreografía de Procesos Para Crear Sistemas de eSalud en Entornos Heterogéneos

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tesis doctoral

Autor
Jose Luis Bayo Montón

Directores
Vicente Traver Salcedo, Carlos Fernández Llatas y Antonio Martínez Millana

10 de enero de 2023, Valencia (España)

Agradecimientos

La realización de esta tesis ha sido un camino largo y complejo, que no habría sido posible sin el apoyo de compañeros, amigos y familiares.

Quiero agradecer a mis directores Vicente Traver Salcedo, Carlos Fernández Llatas y Antonio Martínez Millana su apoyo incondicional en este largo periodo y sus consejos.

Agradecer a mis compañeros y amigos de trabajo por sus ánimos y apoyo durante todos estos años.

A mis amigos, por estar ahí cuando los he necesitado, en especial a Jose Vicente, que me ha apoyado en momentos complicados.

Pero, sobre todo, a mi familia, que me ha apoyado, animado y proporcionado el tiempo sin el que esto no habría sido posible. Os quiero.

III

Índice

Índice de figuras

Índice de tablas

Resumen

Los sistemas de salud actuales están viendo cómo la cantidad de pacientes que tienen que atender, así como el número de servicios diferentes que han de prestar, es cada vez mayor, por lo que se cuestiona si serán sostenibles a largo plazo. Uno de los factores importantes de este aumento es el envejecimiento de la población, lo que se traduce en un mayor número de pacientes crónicos y personas dependientes. Al mismo tiempo, este cambio poblacional crea la necesidad en los sistemas de salud de abordar soluciones de prevención sobre la población general.

Para hacer frente a estos problemas, se está recurriendo a la aplicación de las Tecnologías de la Información y la Comunicación (TIC) en el ámbito de la salud, son los llamados sistemas de eSalud. Se está produciendo la transición de tecnologías o paradigmas aplicados en el ámbito de la empresa al campo de la salud, buscando mejorar sus sistemas y procesos.

En el desarrollo de sistemas de eSalud es necesario tener en cuenta que deben trabajar en entornos altamente heterogéneos y cambiantes. Además, han de ser capaces de adaptarse a las nuevas necesidades demandadas por la población, todo ello sin reducir la calidad de los servicios ya prestados y sin disparar los costes del sistema.

A la hora de desarrollar un sistema software, el modelo de arquitectura que se elija marcará qué características del sistema resultante se van a potenciar. El paradigma de Arquitectura Orientada a Servicios (SOA) presenta entre sus principales beneficios una alta flexibilidad, reducción de costes y desarrollo rápido, así como permite la escalabilidad de sistemas. Estas características apuntan a este tipo de paradigmas como un buen candidato a la hora de crear sistemas de eSalud.

Existen diferentes formas de construir una arquitectura SOA, atendiendo a cómo se componen sus servicios y a cómo se aplican las directrices SOA. En esta tesis se propone el uso del paradigma de coreografía de procesos, siendo éste un modelo en el que se busca construir la arquitectura del sistema desde el punto de vista de los procesos de negocio de la organización. Este paradigma trabaja desde el concepto de un proceso único definido de forma distribuida, empleando coreografía de servicios para la composición. Es decir, no existe un elemento que centralice la toma de decisiones, donde cada servicio es consciente de qué debe realizar y cómo interactuar con el resto.

En la coreografía de procesos se prioriza la eficiencia del sistema, aumentando el acoplamiento entre servicios, lo que puede reducir la flexibilidad del sistema SOA. Pero genera un sistema con mayor rendimiento y con una mejor alineación con los procesos de negocio. Además, la coreografía es más robusta que otros mecanismos de composición y ayuda con la integración de sistemas entre empresas.

El objetivo de esta tesis es validar si el paradigma de coreografía de procesos es aplicable al desarrollo de sistemas de eSalud, dadas sus características altamente heterogéneas y cambiantes, así como sus requisitos de rapidez de desarrollo, eficiencia, flexibilidad, escalabilidad o reducción de costes. Para ello se ha realizado la aplicación de este paradigma en tres escenarios de eSalud diferentes.

En el primero de los escenarios, se creó un sistema para integrar y evaluar un dispositivo tecnológico puntero, como son las Google Glass. Esto demostró que se puede crear un sistema de eSalud basado en coreografía de procesos que integre dispositivos tecnológicos complejos.

En el segundo de los escenarios, se validó el paradigma en un entorno real, creando un sistema distribuido de ejecución de modelos híbridos para la predicción y detección de diabetes tipo 2. El sistema permitió construir servicios para la ejecución de los modelos híbridos (integrando motores estadísticos comerciales como R y Matlab) y la integración con servicios de terceros para acceso a los datos clínicos (I2B2), así como la integración con las aplicaciones de cliente. De esta forma se validó que la coreografía de procesos ayuda a la integración con sistemas externos, permitiendo un desarrollo rápido y la creación de sistemas distribuidos.

En el tercero de los escenarios, se aplica el paradigma de coreografía de procesos orientado al modelo de Internet de las Cosas (IoT) integrando sensores portables para crear un sistema de eSalud. El sistema desarrollado integra un kit de sensores de eSalud (de bajo coste y orientados a pruebas de laboratorio) para permitir el seguimiento y monitorización remota de pacientes, comparando su rendimiento sobre un ordenador contra el rendimiento en una Raspberry Pi. El resultado refrenda la hipótesis de que la coreografía de procesos permite aplicarse para crear fácilmente sistemas de eSalud orientados a IoT e integrar sensores portables de este ámbito.

Los resultados obtenidos en los tres escenarios descritos muestran que diseñar sistemas de eSalud, aplicando el paradigma de coreografía de procesos, permite crear sistemas de eSalud de forma rápida, apoyando la integración de sistemas de terceros, la escalabilidad y la reducción de costes.

Resum

Els sistemes de salut actuals estan veient com la quantitat de pacients que han d'atendre, així com el nombre de servicis diferents que han de prestar, és cada vegada major, per la qual cosa es qüestiona si seran sostenibles a llarg termini. Un dels factors importants d'este augment és l'envelliment de la població, la qual cosa es traduïx en un nombre més gran de pacients crònics i persones dependents. Al mateix temps, este canvi poblacional crea la necessitat en els sistemes de salut d'abordar solucions de prevenció sobre la població general.

Per a fer front a estos problemes, s'està recorrent a l'aplicació de les Tecnologies de la Informació i la Comunicació (TIC) en l'àmbit de la salut, són els cridats sistemes d'eSalut. S'està produint la transició de tecnologies o paradigmes aplicats en l'àmbit de l'empresa al camp de la salut, buscant millorar els seus sistemes i processos.

En el desenrotllament de sistemes d'eSalut és necessari tindre en compte que han de treballar en entorns altament heterogenis i canviants. A més, han de ser capaços d'adaptar-se a les noves necessitats demandades per la població, tot això sense reduir la qualitat dels servicis ja prestats i sense disparar els costos del sistema.

A l'hora de desenrotllar un sistema programari, el model d'arquitectura que es trie marcarà quines característiques del sistema resultant es van a potenciar. El paradigma d'Arquitectura Orientada a Servicis (SOA) presenta entre els seus principals beneficis una alta flexibilitat, reducció de costos i desenrotllament ràpid, així com permet l'escalabilitat de sistemes. Estes característiques apunten a este tipus de paradigmes com un bon candidat a l'hora de crear sistemes d'eSalut.

Hi ha diferents formes de construir una arquitectura SOA, atenent a com es componen els seus servicis i a com s'apliquen les directrius SOA. En esta tesi es proposa l'ús del paradigma de coreografia de processos, sent este un model en què es busca construir l'arquitectura del sistema des del punt de vista dels processos de negoci de l'organització. Este paradigma treballa des del concepte d'un procés únic definit de forma distribuïda, emprant coreografia de servicis per a la composició. És a dir, no hi ha un element que centralitze la presa de decisions, on cada servici és conscient de què ha de realitzar i com interactuar amb la resta.

En la coreografia de processos es prioritza l'eficiència del sistema, augmentant l'adaptament entre servicis, la qual cosa pot reduir la flexibilitat del sistema SOA. Però genera un sistema amb major rendiment i amb una millor alineació amb els processos de negoci. A més, la coreografia és més robusta que altres mecanismes de composició i ajuda a la integració de sistemes entre empreses.

L'objectiu d'esta tesi és validar si el paradigma de coreografia de processos és aplicable al desenrotllament de sistemes d'eSalut, donades les seues característiques altament heterogènies i canviants, així com els seus requisits de rapidesa de desenrotllament, eficiència, flexibilitat, escalabilitat o reducció de costos. Per a això s'ha realitzat l'aplicació d'este paradigma en tres escenaris d'eSalut diferents.

En el primer dels escenaris, es va crear un sistema per a integrar i avaluar un dispositiu tecnològic punter, com són les Google Glass. Açò va demostrar que es pot crear un sistema d'eSalut basat en coreografia de processos que integre dispositius complexos de forma ràpida i eficient.

En el segon dels escenaris, es va validar el paradigma en un entorn real, creant un sistema distribuït d'execució de models híbrids per a la predicció i detecció de diabetis tipus 2. El sistema va permetre construir servicis per a l'execució dels models híbrids (integrant motors estadístics comercials com a R i Matlab) i la integració amb servicis de tercers per a accés a les dades clíniques (I2B2) , així com la integració amb les aplicacions de client. D'esta manera es va validar que la coreografia de processos ajuda amb la integració amb sistemes externs, permetent un desenrotllament ràpid i la creació de sistemes distribuïts.

En el tercer dels escenaris, s'aplica el paradigma de coreografia de processos orientat al model d'Internet de les Coses (IoT) integrant sensors portables per a crear un sistema d'eSalut. El sistema desenrotllat integra un kit de sensors d'eSalut (de baix cost i orientats a proves de laboratori) per a permetre el seguiment i monitorització remota de pacients, comparant el seu rendiment sobre un ordinador contra el rendiment en una Raspberry Pi. El resultat referenda la hipòtesi que la coreografia de processos permet aplicar-se per a crear fàcilment sistemes d'eSalut orientats a IoT i integrar sensors portables d'este àmbit.

Els resultats obtinguts en els tres escenaris descrits mostren que dissenyar sistemes d'eSalut, aplicant el paradigma de coreografia de processos, permet ajudar a crear sistemes d'eSalut de forma ràpida i eficient, ajudant amb la integració de sistemes de tercers, l’escalabilitat i la reducció de costos.

Abstract

Today's health systems are seeing an increasing number of patients to be cared for, as well as a growing number of different services to be provided, raising questions as to whether they will be sustainable in the long term. One of the critical factors for this increase is the ageing of the population, which translates into a higher number of chronic patients and dependents. At the same time, this population shift creates a need for health systems to address prevention solutions in the general population.

To address these problems, the application of Information and Communication Technologies (ICT) in the field of health, the so-called eHealth systems, is being used. The transition of technologies or paradigms applied in the area of business to the field of health is taking place, seeking to improve its systems and processes.

In the development of eHealth systems, it is necessary to consider that they must work in highly heterogeneous and changing environments. Moreover, they must be able to adapt to the new needs demanded by the population, without reducing the quality of the services already provided and without increasing the costs of the system.

When developing a software system, the architecture model chosen will determine which characteristics of the resulting system will be enhanced. The Service Oriented Architecture (SOA) paradigm presents among its main benefits high flexibility, cost reduction and rapid development, as well as allowing the scalability of systems. These characteristics point to this paradigm as a good candidate when creating eHealth systems.

There are different ways to build an SOA architecture, depending on how its services are composed and how the SOA guidelines are applied. This thesis proposes the use of the process choreography paradigm, which is a model that seeks to build the architecture of the system from the point of view of the organization’s business processes. This paradigm works from the concept of a single process defined in a distributed way, using service choreography for the composition. In other words, there is no centralized decision-making element, where each service is aware of what it must do and how to interact with the rest.

Process choreography prioritizes system efficiency by increasing coupling between services, which can reduce the flexibility of the SOA system. But it generates a system with higher performance and better alignment with business processes. In addition, choreography is more robust than other composition mechanisms and helps to cross-company system integration.

The aim of this thesis is to validate whether the process choreography paradigm is applicable to the development of eHealth systems, given their highly heterogeneous and changing characteristics, as well as their requirements for speed of development,

efficiency, flexibility, scalability, and cost reduction. To this end, this paradigm has been applied in three different eHealth scenarios.

In the first scenario, a system was created to integrate and evaluate a cutting-edge technological device such as Google Glass. This demonstrated that it is possible to create an eHealth system based on process choreography that integrates complex devices quickly and efficiently.

In the second scenario, the paradigm was validated in a real environment, creating a distributed hybrid model execution system for the prediction and detection of type 2 diabetes. The system allowed the construction of services for the execution of hybrid models (integrating commercial statistical engines such as R and Matlab) and integration with third-party services for access to clinical data (I2B2), as well as integration with client applications. In this way, it was validated that the process choreography helps with the integration with external systems, enabling rapid development and the creation of distributed systems.

In the third scenario, the process choreography paradigm is applied to the Internet of Things (IoT) model by integrating wearable sensors to create an eHealth system. The developed system integrates a kit of eHealth sensors (low-cost and oriented to laboratory tests) to allow remote monitoring and tracking of patients, comparing their performance on a computer against the performance on a Raspberry Pi. The result supports the hypothesis that process choreography can be applied to easily create IoT-oriented eHealth systems and integrate IoT wearable sensors.

The results obtained in the three scenarios described above show that designing eHealth systems by applying the process choreography paradigm can help to create eHealth systems quickly and efficiently, helping with the integration of third-party systems, scalability, and cost reduction.

Glosario

Números

- **2h-OGTT:** Prueba de tolerancia a la glucosa a las 2 horas.

A

- **AHT:** Medicación antihipertensiva (Anti-Hypertensiver medication).
- **API:** Interfaz de programación de aplicaciones (Application Programming Interface)

B

- **B2B:** Comunicación módulo a módulo (Bussiness-to-Bussiness).
- **B2C:** Comunicación módulo a cliente (Bussiness-to-Client).
- **BLE:** Bluetooth de baja energía (Bluetooth Low Energy).

C

- **CPU:** Unidad central de procesamiento (Central Process Unit).
- **CRC:** Celda de I2B2 denominda Clinical Research Chart.

D

- **DM2:** Diabetes Mellitus Tipo 2.
- **DTO:** Objeto de transferencia de datos (Data Transfer Object).
- **DW:** Almacén de datos (Data Warehouse).

E

- **EAV:** Modelo Entidad-Atributo-Valor.
- **ECG:** Electrocardiograma.
- **ECLAP:** Early Lung Cancer Action Program.
- **EHR:** Historia clínica electrónica (Electronic Health Record).
- **EMG:** Electromiograma.
- **ETL:** Operaciones de extracción, transformación y carga de datos (Extract, Transform, Load).

F

- **FG:** Glucosa en ayunas (Fasting Glucose).
- **FHD:** Antecedentes familiares de diabetes (Family History of Diabetes).
- **FHIR:** Fast Health Interoperability Resources

- **FIPA:** Foundation for Intelligent Physical Agents.

G

- **GB:** Gigabyte.
- **GSR:** Respuesta galvánica de la piel (Galvanic Skin Response).
- **GDK:** Glass Development Kit.
- **GHz:** Gigahercio.
- **GSON:** Google Gson. Librería para trabajar con JSON.
- **GUI:** Interfaz gráfico de usuario (Graphical User Interface).
-

H

- **HbA1C:** Hemoglobina glicosilada.
- **HDL:** Lipoproteína de alta densidad (High-Density Lipoprotein).
- **HIS:** Sistema de información hospitalario (Hospital Information System).
- **HL7:** Health Level 7 Association.
- **HTTP:** Protocolo de transferencia de hipertexto (Hypertext Transfer Protocol).

I

- **I2B2:** Informatics for Integrating Biology and the Bedside.
- **I.C.**: Intervalo de Confianza.
- **IEEE:** Institute of Electrical and Electronics Engineers.
- **IoT:** Internet de las cosas (Internet of Things).
- **IMC:** Índice de masa corporal.
- **ISO:** International Organization for Standardization.

J

- **JPG:** Formato de imagen creado por el Joint Photographic Experts Group.
- **JSON:** Notación de objeto JavaScript (JavaScript Object Notation).

K

- **KPI:** Indicador clave de rendimiento (Key Performance Indicator).

L

- **LED:** Diodo emisor de luz (Light-Emitting Diode).
- **LL:** Medicación para la reducción de grasas o hipolepimiante (Lipid-Lowering medication).

M

- **MB:** Megabyte.
- **MDI:** Inputación de datos faltantes (Missing Data Imputation).
- **MIT:** Massachusetts Institute of Technology.
- **Mpx:** Megapixel.
- **MP4:** Formato de vídeo MPEG-4.

N

- **NIH:** Instituto Nacional de Salud (National Institute of Health).

O

- **OASIS:** Organization for the Advancement of Structured Information Standards.
- **OMS:** Organización Mundial de la Salud.
- **ONT:** Celda de I2B2 denominada Ontology Management.
- **OSGi:** Open Services Gateway initiative.

P

- **P2P:** Red de pares (Peer-to-Peer).
- **PCHA:** Personal Connected Health Alliance.
- **PM:** Celda de I2B2 denominada Project Management.

Q

- **QE:** Motor de consultas (Query Engine).

R

- **RAM:** Memoria de acceso aleatorio (Random-Access Memory).
- **REST:** Transferencia de estado representacional (REpresentational State Transfer).
- **RESTful:** Que soporta REST.
- **ROI:** Retorno sobre la inversión (Return On Investment).
- **RSM:** Módulo de escalas de riesgo (Risk Score Module)

S

- **SOA:** Arquitectura orientada a servicios (Service Oriented Architecture).
- **SOAP:** Protocolo simple de acceso a objetos (Simple Object Access Protocol).
- **SQL:** Lenguaje de consulta estructurada (Structured Query Language).
- **SSL:** Protocolo de seguridad de la capa de transporte (Secure Socket Layer).

T

- **TCP:** Protocolo de control de transmisión (Transmission Control Protocol).
- **TIC:** Tecnología de la Información y la Comunicación.

U

- **UDP:** Protocolo de datagramas de usuario (User Datagram Protocol).
- **UML:** Lenguaje unificado de modelado (Unified Modeling Language).
- **UUID:** Estándar de identificador único universal (Universally Unique IDentifier).

W

- **WLAN:** Red de área local (Wireless Local Area Network).
- **WSD:** Whole System Demonstrator Programme.

X

- **XML:** Formato de lenguaje de marcado extensible (eXtensible Markup Language).
- **XMSG:** Formato de mensaje extensible de corografía (eXtensible MeSsaGe).

1. Introducción y estado del arte

1.1. Prólogo

La sociedad actual se encuentra ante el reto de hacer frente a las necesidades derivadas de una inversión de la pirámide poblacional [1]. El envejecimiento de la población es cada vez mayor, lo que implica un mayor número de pacientes crónicos y personas dependientes. Los sistemas de salud son uno de los grandes afectados por esta realidad, ya que se están viendo obligados a adaptarse a las nuevas necesidades provocadas por este cambio en la edad de la población [2]. Al mismo tiempo, y propiciado por este envejecimiento de la población, los sistemas de salud están ampliando sus contextos de aplicación y ya no sólo actúan sobre pacientes, sino que se les pide que realicen acciones de prevención en población sana [3].

La cada vez mayor aplicación de las Tecnologías de la Información y las Comunicaciones (TIC) en el ámbito de la salud, está generando nuevas herramientas o sistemas denominados de eSalud que tratan de ayudar a los sistemas sanitarios a hacer frente a estos nuevos retos [4].

Estos nuevos sistemas deben de abordar los requisitos de un ámbito altamente heterogéneo, ya que el sistema sanitario debe soportar todo tipo de patologías y contextos de aplicación: hospitales, centros de atención primaria, unidades móviles, hogar del paciente, etc. Necesitan tener la capacidad de adaptarse rápidamente a las nuevas necesidades que está generando el cambio poblacional, pero sin perder de vista que los actuales niveles de servicios prestados no deben reducirse, ni en cantidad, ni en calidad. Esto implica la necesidad de trabajar en la reducción de los costes para que el sistema sea sostenible.

Cuando se aborda el desarrollo de un sistema TIC, los requisitos del sistema donde se va a utilizar condicionan el diseño de su arquitectura y qué paradigma de diseño se adecua más a sus necesidades.

En esta tesis se plantea el uso del paradigma de coreografía de procesos como una solución a los retos de flexibilidad para adaptarse a sistemas altamente heterogéneos, capacidad de integrar sistemas y capacidad de adaptación al cambio.

Los siguientes puntos de este capítulo abordarán diversos aspectos que ponen en contexto el trabajo realizado. Lo primero que se contextualiza es la definición de eSalud, cuando hablamos de un sistema para eSalud, no consiste simplemente en aplicar algún elemento TIC a una enfermedad. La eSalud es un término mucho más amplio, que hace referencia al uso coste-efectivo de las TIC en todas las facetas sanitarias, en servicios de

salud, en investigación, en educación, en salud pública (ayuda a la toma de decisiones, prevención), etc.

Para crear un sistema de eSalud es necesario disponer de varios elementos. Primero necesitaremos obtener datos de las personas de la mejor forma posible, una de las formas que más auge está teniendo en los últimos años es el uso de dispositivos que la personas podemos llevar encima de forma fácil, los denominados dispositivos *wearable* o portables. Estos dispositivos, como pulseras de actividad o dispositivos portátiles de monitorización, permiten capturar los datos a diario y directamente de la persona, sin que ésta tenga que estar en un centro sanitario.

No obstante, los dispositivos portables por sí solos no son un sistema de eSalud, es necesario describir cómo estos dispositivos van a interactuar entre ellos y comunicar dichos datos. El concepto de Internet de las Cosas (IoT) define sistemas en los que cada elemento (sensor, actuador, etc.) conforma un bloque de una red IoT, en la que dichos bloques interactúan e intercambian datos. Para construir sistemas IoT es necesario definir la arquitectura de los mismos, es aquí donde aparece el paradigma de Arquitectura Orientada a Servicios (SOA).

El paradigma SOA se basa en la existencia de diferentes servicios, cada uno de los cuales ofrece una funcionalidad autónoma. La interacción conjunta de estos servicios proporciona una funcionalidad completa que conforma el sistema completo. Este tipo de arquitectura tiene entre sus principales ventajas el desarrollo eficiente, la flexibilidad, escalabilidad o el aumento del ROI.

La manera en que se lleva a la práctica una arquitectura SOA es también diferente en función de cómo se aplican sus directrices, pero la principal de todas es decidir cómo se coordinan y componen los servicios. Tradicionalmente, se distingue la orquestación y la coreografía de servicios, la primera de ellas basada en un elemento central que organiza la comunicación y, la segunda, basada en una colaboración distribuida.

El paradigma de coreografía de procesos aplica las directrices SOA desde el punto de vista de los procesos de negocio. La funcionalidad del sistema se interpreta como un proceso único y distribuido, se deberá analizar el proceso hasta llegar a sus funcionalidades base y cómo se organizan. Cada uno de los servicios del sistema implementará una funcionalidad base, y la coreografía del proceso proporcionará la funcionalidad completa del sistema. Entre las ventajas de este paradigma encontraríamos una mayor robustez y rendimiento, además de facilitar la colaboración con servicios de terceras empresas.

Por último, se describe la herramienta CHOREMED, la cual permite aplicar el paradigma de coreografía de procesos y ha sido empleada para llevar a cabo los diferentes estudios realizados para validar la hipótesis planteada en esta tesis.

1.2. Sistemas de eSalud

El termino eSalud (o e-Salud) se origina como traducción del término inglés e-Health, el cual es habitual encontrar en definiciones de numerosos artículos científicos y páginas web. Definiciones que, generalmente, siempre coinciden en incluir términos como "salud" o "prestación de servicios de salud" relacionados con el término "tecnología", pero que no siempre tienen la misma orientación (centrada en la salud de los pacientes o en la mejora de las organizaciones), o abordan la salud desde el mismo punto de vista (gestión de la enfermedad, mejora general de la salud, prevención) [5].

La Organización Mundial de la Salud (OMS) en mayo del 2005 adopta una resolución, en la que reconoce el potencial impacto que los avances de las TIC pueden tener en la prestación de servicios de salud, la salud pública, la investigación y las actividades relacionadas con la salud en beneficio, tanto de los países con bajos ingresos, como en los de altos ingresos. Además, realiza la siguiente definición: "*Se define la eSalud como el uso coste-efectivo y seguro de las Tecnologías de la Información y las Comunicaciones en apoyo de la salud y los campos relacionados con la salud, incluidos los servicios de atención médica, la vigilancia de la salud, la literatura sobre salud y la educación, el conocimiento y la investigación en salud"* [6].

Esta definición más amplia abarca, no sólo la prestación de servicios médicos, sino que hace hincapié en términos de ser proactivos en las intervenciones de salud pública preventiva, en el acceso a la educación y en la investigación. Todo ello bajo la premisa de que sea un uso coste-efectivo, que no se introduzca tecnología por el simple hecho de seguir una moda, y, por descontado, proporcionando seguridad, pues se trata con datos de alto riesgo.

La eSalud, por tanto, engloba múltiples términos que suelen nombrarse en torno a las TIC y la salud. Por ejemplo, en 2010 la OMS realizó su tercera encuesta global sobre eSalud y organizó la información recogida sobre ocho temáticas: fundaciones de eSalud, salud móvil (mHealth), telemedicina, aprendizaje electrónico (e-learning) en ciencias de la salud, historia clínica electrónica, marcos legales para eSalud, redes sociales y big data [7].

Tal como indica la Comisión Europea en su plan de acción sobre salud electrónica 2012-2020, "*La salud electrónica puede redundar en beneficio de los ciudadanos, los pacientes y los profesionales de la salud y la asistencia, además de las organizaciones sanitarias y los poderes públicos. De aplicarse eficazmente, la salud electrónica facilita una atención sanitaria más personalizada y centrada en los ciudadanos, más específica, efectiva y eficaz, lo que contribuye a disminuir los errores médicos y la duración de la hospitalización. También, favorece la igualdad y la integración socioeconómica, la calidad de vida y la capacitación de los pacientes gracias a una mayor transparencia, al acceso a los servicios y la información y al empleo de medios sociales para la salud*" [4].

Dicho plan divide los retos a afrontar por los sistemas de eSalud en 4 puntos:

1. Mejorar el tratamiento de enfermedades crónicas, así como el fomento y prevención de la salud.
2. Aumentar la sostenibilidad y eficacia de los sistemas sanitarios.
3. Fomentar la universalidad y equidad transfronteriza.
4. Mejorar las condiciones jurídicas.

Y, para ello, se centra en superar los siguientes obstáculos:

- Conseguir una mayor interoperabilidad de los servicios de eSalud.
- Apoyar la investigación, desarrollo e innovación.
- Facilitar la aceptación e implantación de los sistemas de eSalud.
- Promover la cooperación internacional en materia de eSalud.

1.3. Wearables

Uno de los pilares de la eSalud es la obtención y manejo eficiente de datos del paciente de la mano de las TICs. En este sentido, una de las tecnologías más disruptivas de los últimos años son los sensores denominados "*wearables*". Éstos, permiten la obtención directa de datos de salud del paciente, permitiendo el seguimiento del estado de los pacientes de forma eficiente y sencilla (por ejemplo, el seguimiento de crónicos).

El término inglés "*wearable technology*" o "*wearables*", suele traducirse al castellano con diferentes términos como "tecnología portable, ponible, llevable o vestible".

Una de las definiciones más referenciadas (de uno de los precursores de este tipo de dispositivo) es la realizada por Steve Mann en 1996: *"La computación portable es el estudio o la práctica de inventar, diseñar, construir o usar dispositivos sensoriales y computacionales en miniatura llevados por el cuerpo. Las computadoras portables se pueden usar debajo, sobre o dentro de la ropa, o también pueden ser ropa"* [8]. En esta publicación, Mann nos muestra su dispositivo "*EyeTap goggles*" el cual podía proyectar una imagen en un ojo del usuario para reforzar su percepción e información contextual.

Desde entonces, numerosos autores definen estos dispositivos como elementos tecnológicos que se incorporan al usuario o su ropa (ya sea como textiles o complementos). Estos dispositivos miden características del usuario o del ambiente y ayudan a la persona. Generalmente, se incluyen también términos relacionados con la comunicación inalámbrica [9] [10] [11] [12].

Otras definiciones más modernas, amplían un poco más este concepto indicando que "*Los wearables incluyen todas las formas de dispositivos electrónicos computacionales o*

sensoriales que se pueden usar con la ropa o en el cuerpo. En el sentido más amplio, cualquier dispositivo informático que lleva una persona para ayudarla podría llamarse dispositivo portátil" [13].

En la actualidad, es habitual encontrar dispositivos de este tipo al alcance del usuario, como relojes inteligentes, monitores de actividad deportiva o gafas de realidad aumentada (Figura 1).

Figura 1: Ejemplo de dispositivos wearables: Apple Watch[1], Fitbit[2] y Google Glass[3]

En general, los dispositivos de este tipo se podrían clasificar en varias categorías en función de su uso [14]:

- **Entretenimiento:** relojes, gafas o dispositivos para llamadas de voz o vídeo, y control de los gestos, etc.
- **Estilo de vida:** dispositivos de realidad aumentada, guantes inteligentes, gestos, control de dispositivos, etc.
- **Entrenamiento:** dispositivos para medir pulsaciones, distancia, temperatura, etc.
- **Médicos:** dispositivos para monitorizar señales biomédicas tales como el ECG, pulsaciones, EMG, temperatura, glucosa, oxígeno en sangre, etc.; que permiten la monitorización remota de pacientes.
- **Industriales:** incluye dispositivos tipo manos libres que apoyan en las operaciones de negocio para propósitos industriales.
- **Orientados al juego:** aquellos dispositivos que emplean la realidad aumentada con el objetivo de mejorar la jugabilidad.

Esta tesis se centrará exclusivamente en aquellos dispositivos portables de orientados a la salud (tanto certificados como no certificados y de uso exclusivo en laboratorio).

[1] https://www.apple.com/es/apple-watch-series-8/
[2] https://www.fitbit.com/global/es/products/trackers/luxe
[3] https://www.google.com/glass/tech-specs/

1.4. Internet de las Cosas

El uso de dispositivos portables permite el acceso a los datos, pero una vez se dispone de todo un ecosistema de dispositivos, se necesita plantear como éstos interactúan, se comunican y envían los datos para obtener una utilidad. Uno de los conceptos que se asocia con asiduidad al uso de dispositivos portables y que tiene gran impacto en el sector TIC actual es el de Internet de las Cosas.

El término Internet de las Cosas tiene su origen el en término inglés "Internet of Things" (IoT), el cual fue propuesto en 1999 por Kevin Ashton durante una presentación en el Instituto Tecnológico de Massachusetts (MIT) [15].

Desde su origen, han surgido diferentes definiciones que tienen en común la búsqueda de la integración del mundo físico con el virtual de Internet. La Comisión Europea realiza un compendio de diferentes definiciones para indicar que *"La IoT se puede definir en términos generales como una infraestructura de red global, que vincula objetos, cosas y dispositivos físicos y virtuales identificados de forma única a través de la explotación de capacidades de captura (detección), comunicación y actuación de datos. La infraestructura subyacente de "cosas" representadas virtualmente en una estructura similar a Internet incluye desarrollos de redes e Internet existentes y en evolución. Los servicios y aplicaciones emergentes se caracterizarán por un alto grado de captura de datos autónoma, transferencia de eventos, conectividad de red e interoperabilidad"* [16].

Por tanto, IoT se define como un concepto abstracto o innovación tecnológica. En la que los diferentes elementos (sensores, actuadores, etc.) son los bloques que conforman la red IoT. Los sensores detectan cambios físicos y generan un evento que un observador utiliza para actuar.

Las aplicaciones para IoT permiten dos tipos de interacciones, entre objetos o entre un objeto y el usuario, siendo estas últimas las que normalmente requieren de la visualización de datos. Las aplicaciones para IoT pueden ser englobadas en tres categorías [17]:

- **Monitorización y control:** Recolectan información de un equipo, elemento o persona. Realizando en algunos casos acciones de control en función de los datos. En esta categoría se incluye el denominado "Smart Home" u "Hogar Inteligente", que implica que el hogar actúe de forma inteligente (anticipando o respondiendo a las necesidades de la persona) empleando IoT.
- **Big Data y análisis de negocio:** Generación de un gran volumen de datos que se analizan posteriormente para obtener información útil. Esto involucra sobre todo grandes operaciones de negocio.

- **Intercambio de información y colaboración:** Entre personas, entre personas y cosas o entre cosas, como mantener informado a un comerciante del estado de su negocio o intercambiar información entre empresas para gestión de logística.

En esta tesis, se realiza la construcción de entornos basados en el concepto de IoT en diferentes casos de uso, aplicando conceptos como el mecanismo de comunicación e interacción entre objetos, y la abstracción de objetos físicos en forma de servicios software.

1.5. SOA

Para el desarrollo de sistemas software basados en IoT, es necesario definir la arquitectura del mismo. Uno de los paradigmas más utilizados para este fin, es el de Arquitectura Orientada a Servicios (Software Oriented Architecture, SOA).

La organización OASIS define el paradigma de Arquitectura Orientada a Servicios como un *"paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de propiedad"* [18]. La especificación SOA-RM basa su definición de SOA en torno al concepto de "necesidades y capacidades", donde SOA proporciona un mecanismo para satisfacer las necesidades de los servicios consumidores con las capacidades proporcionadas por los servicios proveedores. El servicio se define como un *"mecanismo para permitir el acceso a una o más capacidades, donde el acceso se proporciona utilizando una interfaz definida y se ejerce de manera coherente con las restricciones y políticas especificadas por la descripción del servicio"* [18].

Otras organizaciones, como The Open Group, ofrecen una definición más concreta indicando que se trata de *"un estilo de arquitectura que da soporte a la orientación a servicios"* [19] y define el servicio como la *"representación lógica de una actividad comercial repetible que tiene un resultado específico. Es auto contenido, puede estar compuesto de otros servicios y es una caja negra para los consumidores del servicio"* [19].

Así pues, este tipo de paradigma persigue diseñar diferentes servicios que ofrecen funcionalidades de forma autónoma e interactúan entre sí para ofrecer una funcionalidad mayor [19] [20] [21].

En general, las ventajas de una arquitectura orientada a servicios se pueden a resumir en:

- **Desarrollo eficiente:** Cada servicio se especializa en una funcionalidad de forma eficiente.
- **Desarrollo incremental:** El sistema crece en funcionalidad con cada servicio que se añade.

- **Aumento de la reusabilidad:** Los servicios pueden utilizarse una y otra vez combinados de formas distintas.
- **Aumento de la flexibilidad:** Resulta sencillo mejorar o ampliar una funcionalidad simplemente cambiando que servicios interactúan para conseguirla.
- **Mayor interoperabilidad:** Los servicios se diseñan para ser compatibles.
- **Aumento del ROI y reducción de costes:** Los servicios se crean como parte del sistema de información de la organización, con vistas a ser reutilizados una y otra vez aumentando el valor de retorno y los costes.
- **Aumento de la escalabilidad:** Los servicios funcionan de forma distribuida y la arquitectura debe ser capaz de integrar diferentes proveedores.

1.6. Coreografía de procesos

Uno de los aspectos más importantes en una arquitectura SOA es definir cómo se coordinan los servicios para interactuar entre ellos y componer una funcionalidad. En SOA, esto se denomina composición de servicios y consiste en combinar un número de servicios simples para generar uno más complejo [19].

Habitualmente se distinguen dos estilos de composición, la orquestación de servicios y la coreografía de servicios [18] [19] [22] [23] (Figura 2):

- **La orquestación de servicios:** se caracteriza por tener un elemento central que gestiona y controla la interacción de los servicios.
- **La coreografía de servicios:** se caracteriza por que todos los servicios se encuentran al mismo nivel, los servicios interaccionan entre sí, sin un elemento que los coordine. Cada elemento es responsable de ejecutar su parte de la interacción y conoce con quién debe interactuar.

Es habitual utilizar el símil de la orquesta de música y de un cuerpo de baile para entender mejor estos modelos. En una orquesta existe un director, que indica a cada músico cuándo realizar su parte de la partitura, lo cual podría asimilarse a la orquestación de servicios. Por otro lado, asimilando la coreografía de servicios con un cuerpo de baile, el coreógrafo prepara y define qué debe hacer cada bailarín, y cuando se realiza la representación cada bailarín sabe qué debe hacer, cuándo y con quién interactuar.

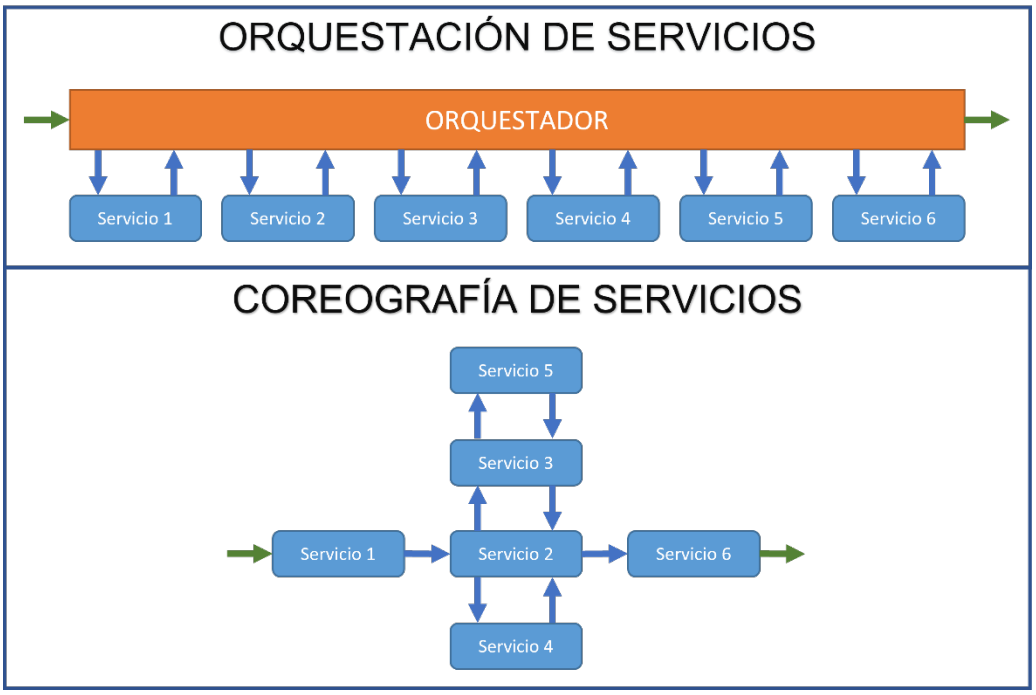

Figura 2: Comparación de Orquestación de servicios y Coreografía de servicios

La definición de una orquestación describe la secuencia de servicios a ejecutar y en qué orden, enlazando el resultado de un servicio con otro. La orquestación puede llevarse a la práctica codificando un servicio que realice la orquestación definida, o puede existir un elemento central capaz de entender esta definición y ejecutarla como un servicio más. Este elemento central orquestador es el que le proporciona a la orquestación una de sus ventajas más importantes en comparación con la coreografía. Este elemento permite que resulte más fácil definir lenguajes y herramientas de ejecución de orquestaciones. Además, estas orquestaciones suelen ser más fáciles de entender y depurar en caso de errores.

Por otra parte, el hecho de que la orquestación se base en un elemento central que coordina todo y centraliza el flujo de datos, es a su vez una de sus mayores desventajas. Al ser las orquestaciones ejecutadas por un elemento central, cuando éste cae, todas fallarán. Aún en el caso de que cada orquestación se haya codificado de forma independiente o se ejecute como un servicio independiente, una vez puesta en marcha, el problema sigue existiendo, ya que si falla este servicio nada se ejecutará. Además, el centralizar el tráfico de datos en un elemento, genera una sobrecarga que reduce notablemente el rendimiento del sistema (en comparación con ejecutar el mismo proceso mediante coreografía) [24].

Independientemente de qué estilo de composición se emplee una arquitectura SOA debe ser capaz de dar soporte a los procesos de negocio de la organización. El proceso de negocio se define *"como una actividad del mundo real que consiste en un conjunto de*

tareas relacionadas lógicamente que, cuando se realizan en la secuencia adecuada y de acuerdo con las reglas empresariales correctas, producen un resultado empresarial" [21]. Por lo que, cuando se habla de servicios que interactúan en la arquitectura, no sólo se habla de servicios internos a la organización, sino que también involucra organizaciones externas. De hecho, este potencial integrador es una de las grandes ventajas de una arquitectura SOA.

El problema en los entornos reales es que las entidades corporativas, a menudo, no están dispuestas a delegar el control de sus procesos comerciales a sus socios de integración, por lo que la orquestación no sería una opción. En cambio, la coreografía de servicios ofrece un medio por el cual las reglas de participación dentro de una colaboración pueden definirse claramente y acordarse conjuntamente. Luego, cada entidad, puede implementar su parte de la coreografía según lo determine la vista común o global [25].

Formalmente podemos decir que *"una coreografía define reglas comunes reutilizables que rigen el orden de los mensajes intercambiados y los patrones de provisión de comportamiento colaborativo, según lo acordado entre dos o más participantes que interactúan"* [25].

La coreografía, al contrario que la orquestación, no se basa en un elemento central que gestiona y organiza. Esto hace que la creación de herramientas para ejecutar definiciones de coreografía y la depuración sean más complejas. Pero, por el contrario, soluciona muchos de los problemas de la orquestación. La hace más robusta ante la caída de un servicio, ya que no toda la funcionalidad del sistema se pierde. También, como ya hemos comentado, es más eficiente en cuanto a rendimiento, ya que no todos los datos pasan por un elemento central. Además, es más flexible y se adapta rápidamente a los cambios, pues, cambiando con quién interactúa un servicio, podemos actualizar todas las coreografías en las que está involucrado, sin ser necesario modificar cada orquestación donde dicho servicio participa.

Por último, como ya se ha mencionado, uno de los puntos más importantes a favor de la coreografía, y por el que se propone su empleo en sistemas de eSalud, es el hecho de resultar más sencilla la integración con componentes de terceros. Las entidades externas no suelen estar dispuestas a ceder el control de sus herramientas, o a implementar sus interfaces en la tecnología de otra organización.

El paradigma de la coreografía de procesos va más allá del mecanismo de composición utilizado para componer los servicios, además, también involucra el cómo se aplica este mecanismo a la organización. Una organización no es un conjunto de funcionalidades dispersas, sino que tiene unos objetivos de negocio entorno a los cuales construye sus procesos. Es, por tanto, necesario evaluar los procesos de negocio de la organización y construir coreografías ajustadas a dichos procesos.

En la coreografía de procesos, los servicios involucrados colaboran entre sí de forma distribuida mediante el intercambio de mensajes y conforme a los protocolos acordados, aportando cada uno su parte de funcionalidad para logar la funcionalidad completa de un proceso de negocio. Al igual que en la vida real, los procesos dependen de que cada trabajador realice su cometido asignado. La lógica del proceso no se centraliza, sino que está distribuida entre todos los servicios que componen la coreografía. A su vez, los procesos en una organización no están aislados, por lo que los procesos definidos en forma de coreografía se organizan y coordinan nuevamente mediante coreografías de mayor nivel.

En la coreografía de procesos se define la funcionalidad del sistema como un proceso único distribuido, el cual se aborda aplicando un enfoque de diseño top-down [26], en el que se parte del proceso completo a abordar y se descompone en sus partes principales. Estas partes se dividen en más partes y así, sucesivamente, hasta llegar a elementos de tamaño suficientemente pequeño para que sean fáciles de diseñar y crear como servicios. Estos servicios serán los que luego realizarán la coreografía del proceso.

Si bien una arquitectura SOA tiene como objetivo facilitar la interoperabilidad mediante un bajo acoplamiento de los servicios, en un entorno empresarial esto reduce la eficiencia en la ejecución de los procesos. La coreografía de procesos se centra en maximizar esa eficiencia, aumentando el acoplamiento de los procesos cuando es necesario. Es decir, prioriza la integración sobre la interoperabilidad. Esto puede reducir la flexibilidad, pero, a cambio, aumenta el rendimiento. Se deja en manos del diseñador de la coreografía decidir cuál es el nivel de acoplamiento adecuado al proceso [27] [28]. Por último, también se consigue un aumento de la alineación del sistema con el proceso de negocio, pues las arquitecturas se diseñan teniendo en cuenta las necesidades de negocio, desde su punto de vista, por lo que tienden a alinearse mejor.

1.6.1. CHOREMED

Para aplicar el paradigma de coreografía de procesos y crear la arquitectura de un sistema de eSalud se requiere del apoyo de herramientas software y de desarrollo que soporten este paradigma. En esta tesis se ha utilizado el coreógrafo CHOREMED [29] el cual se ha desarrollado dentro del grupo SABIEN (del cual el doctorando es miembro activo desde su creación) y ha sido utilizado en diferentes proyectos de investigación. Este coreógrafo parte de una versión inicial basada en OSGi [27] y que, posteriormente, fue portada a la tecnología .NET Framework 2.0 para lograr una mayor portabilidad y eficiencia.

Dicho coreógrafo, cuenta con un motor para la ejecución y comunicación de los servicios, y proporciona la definición de un lenguaje común de intercambio de mensajes entre los servicios para ejecutar las coreografías. CHOREMED también incluye herramientas para

ayudar con la creación de servicios, así como un conjunto de servicios base reutilizables en distintos entornos.

Entre los elementos que proporciona CHOREMED se encuentra como elemento principal un motor de coreografía ligero, que permite su ejecución en dispositivos de bajas prestaciones, lógicamente siempre dependiendo de la carga de los servicios que luego se ejecuten en él. La versión del coreógrafo utilizada en esta tesis estaba desarrollada en .NET Framework 4.5.

Para que un servicio pueda ser ejecutado y se comunique con otros servicios a través del motor de coreografía, CHOREMED define una interfaz de programación (Application Programming Interface, API) que éstos deben cumplir. Para ello, se proporciona un interfaz que cualquier objeto que quiera ser instanciado como servicio debe cumplir, en ella se definen los métodos y propiedades necesarias para realizar las siguientes acciones:

- **Arranque e inicialización:** Se ejecuta al instanciar un servicio en el motor y permite que el servicio inicialice todos los recursos que pueda necesitar durante su funcionamiento. Como, por ejemplo, acceso a bases de datos o recursos hardware.
- **Parada y finalización:** Se ejecuta antes de parar un servicio y permite liberar todos los recursos que el servicio estuviese utilizando. Por ejemplo, liberar recursos hardware, conexiones a otro software o salvar su estado a un medio de almacenamiento persistente.
- **Definición del identificador o dirección lógica:** Permite leer o modificar el conjunto de direcciones lógicas que el servicio utiliza para ser identificado y localizado por otros servicios a la hora de intercambiar mensajes.
- **Recepción y envío de mensajes:** Permite recibir o enviar una copia del mensaje hacia o desde otros servicios empleando el motor de coreografía. Dicho mensaje deberá cumplir con el formato XMSG que se explica más adelante. Algo importante a destacar es que el envío de mensajes es totalmente asíncrono, es decir, cuando un servicio envía un mensaje, éste no se queda bloqueado esperando una confirmación de envío o contestación por parte de otro servicio, sino que continúa con su funcionamiento normal. De esta forma los servicios son más eficientes y se pueden evitar interbloqueos entre servicios.

Para la comunicación entre los distintos servicios se ha diseñado y definido un formato de mensaje común, que los servicios deben ser capaces de entender y crear. Este formato se denomina XMSG (eXtensible MeSsaGe) [27]. XMSG se basa en las recomendaciones [30] de la Fundación para Agentes Físicos Inteligentes (FIPA) y los encabezados del Protocolo Simple de Acceso a Objetos (SOAP) [31] para enrutar y caracterizar los mensajes. Este protocolo está diseñado para permitir enviar mensajes de difusión/multidifusión, así como de igual a igual (P2P). Un mensaje XMSG cuenta con los siguientes campos:

- **Identificador de lenguaje:** Este campo identifica el lenguaje utilizado para crear el mensaje. Dado que éste es un formato orientado a ser extensible para adaptarse a versiones futuras, este campo permite identificar con qué lenguaje se ha construido el mensaje.
- **Identificador de mensaje:** Identifica unívocamente a un único mensaje entre todos los que pueden estar circulando dentro del motor de coreografía. Para la creación de estos identificadores se hace uso del estándar de identificador único universal (UUID) formado por 16 bytes.
- **Emisor:** Contiene la dirección lógica del servicio que envía el mensaje, de esta forma el servicio que recibe el mensaje puede saber quién lo envía. El identificador lógico está compuesto por uno o varios elementos separados por el símbolo del punto, de esta forma es posible realizar jerarquías en las direcciones. El identificador de emisor permite, por ejemplo, saber a quién responder cuando el mensaje recibido así lo requiera.
- **Receptor:** Contiene las direcciones lógicas del servicio o servicios a los que se envía el mensaje. El protocolo debe permitir realizar envíos de mensajes de difusión, multidifusión y P2P. Esto se logra utilizando la combinación de múltiples direcciones de envío, direcciones jerárquicas (usando el punto como separador) y el carácter de comodín *asterisco*, que equivale a cualquier valor dentro de un elemento de la dirección. De esta forma, si se tienen varios sensores que se identifican como: “sensores.temperatura.01”, “sensores.temperatura.02”, se podría enviar un mensaje a todos empleando la dirección de envío “sensores.temperatura.*”, o, si se quiere que cualquier servicio relacionado con la temperatura reciba el mensaje, se puede utilizar “*.temperatura.*”. Para un envío P2P simplemente se incluiría el identificador completo sin comodines.
- **Protocolo:** Este campo indica el tipo de mensaje. El protocolo está diseñado para ser extendido y, por tanto, soportar varios tipos de mensajes que pueden ir creciendo con el tiempo. Actualmente cuenta con los siguientes tipos:
    - **Solicitud:** Este tipo se utiliza para indicar al servicio receptor del mensaje que, tras realizar la tarea solicitada, conteste al servicio solicitante, incluyendo junto a los datos del nuevo mensaje que éste es una respuesta de un mensaje previo concreto (Campo “Identificador de mensaje respondido”). Hay que recordar que el envío de mensajes es totalmente asíncrono.
    - **Información:** Es el tipo utilizado para marcar mensajes que contienen información y que no se espera que el receptor conteste a ellos. De hecho, cuando un servicio procesa un mensaje de tipo “Solicitud”, deberá generar, al terminar la tarea solicitada, otro de respuesta de tipo “Información”, rellenando el campo de “Identificador de mensaje respondido”.

  - **Evento:** Se utiliza cuando se quiere indicar o informar de que se ha producido un acontecimiento concreto. Como, por ejemplo: mandar un valor nuevo de un sensor, indicar cambios en un estado, etc. Cuando se usa este tipo de mensaje, el servicio emisor no espera que se emita una respuesta al mismo.
  - **Administración:** Son mensajes utilizados por el motor de coreografía para realizar acciones internas de gestión y su uso está reservado.
  - **Fallo:** Es el tipo de mensaje utilizado para indicar que se ha producido un fallo. Puede generarlo cualquier servicio o el propio motor de coreografía. Este tipo de mensaje puede usarse, tanto para depurar la fuente del fallo, como para implementar mecanismos de recuperación.
- **Identificador de mensaje respondido:** Cuando un servicio pide a otro que ejecute una tarea mediante un mensaje de tipo "Solicitud", el servicio receptor debe contestar, tras acabar dicha tarea, al mensaje original mediante otro de tipo "Información". En el nuevo mensaje deberá rellenar este campo con el identificador del mensaje "Solicitud" original. Así, el servicio receptor del nuevo mensaje tiene información para detectar que es un mensaje a una solicitud suya previa. Esto es necesario, dado que el envío de mensajes es totalmente asíncrono.
- **Contenido:** Este campo contiene la información necesaria para que el servicio receptor del mensaje sepa qué tarea se le está solicitando y reciba los datos necesarios para ella. Con este objetivo el contenido tiene un formato fijo formado por los siguientes elementos:
  - **Método:** Cada servicio disponible en la coreografía expone una API con los métodos o tareas que otros servicios pueden solicitar, en este campo se pondría el nombre del método o tarea que un servicio solicita a otro.
  - **Parámetros:** Cuando se solicita ejecutar un método, es posible que éste necesite información adicional para completarlo. Esta información llega en la forma de un vector de parámetros. Cada parámetro está formado por su nombre, tipo de parámetro y, por último, su valor.

En relación con los mensajes de coreografía, CHOREMED proporciona diferentes clases y utilidades para la creación y manejo de estos. Permitiendo crear fácilmente mensajes estándar o mensajes más especializados, como: mensajes de tipo "Información" en respuesta a otro de tipo "Solicitud", o mensajes de tipo "Fallo" a partir de excepciones capturadas en el código.

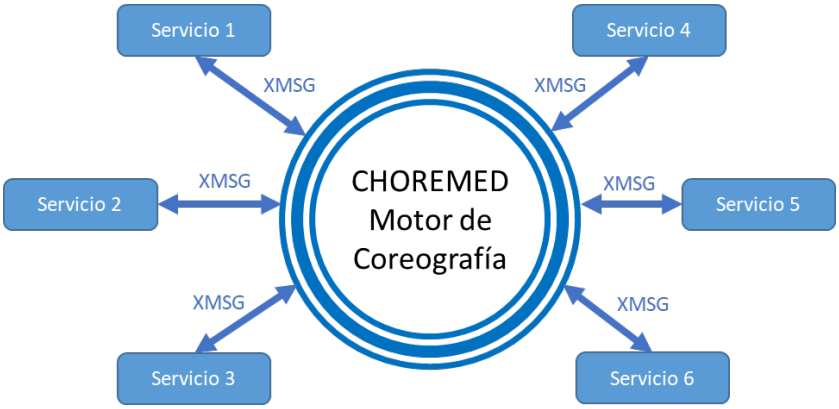

Figura 3: Intercambio de mensajes entre servicios empleando el motor de coreografía de CHOREMED.

De la misma forma, proporciona un mecanismo de serialización en el formato de lenguaje de marcado extensible (eXtensible Markup Language, XML). Un ejemplo pude verse en la Figura 4.

```
<xmsg>
    <message-id>a6dfaea7-5ceb-4e70-8010-ee09529ea05a</message-id>
    <sender>java.ServiceA</sender>
    <receiver>Java.Choreographer.NightService</receiver>
    <protocol>request</protocol>
    <language>XMSG</language>
    <content>
    <Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
        <Body>
            <ExecuteProcess xmlns="http://tempuri.org">
                <ligth1 type="xs:string">
                    Java.Choreographer.ligth.1
                </light1>
                <ligth2 type="xs:string">
                    Java.Choreographer.ligth.2
                </light2>
                <ligth3 type="xs:string">
                    Java.Choreographer.ligth.3
                </light3>
                <ligth4 type="xs:string">
                    Java.Choreographer.ligth.4
                </light4>
            </ExecuteProcess>
        </Body>
    </Envelope>
    </content>
</xmsg>
```

Figura 4: Mensaje XMSG serializado en formato XML

CHOREMED proporciona varios servicios para ayudar en la integración con otros sistemas y la creación de arquitecturas distribuidas de coreografía. Estos servicios se denominan

conectores por la función que realizan, pues su función es la de conectar o comunicar dos extremos para generar un canal de comunicación.

Los conectores proporcionados, generalmente, se catalogan en función del sentido de su comunicación como Receptores o Emisores. En función de si se encargan de recibir mensajes del exterior del coreógrafo e introducirlos, o se encargan de detectar mensajes que deben salir del coreógrafo y enviarlos hacia un tercer elemento externo, un servicio puede realizar ambas cosas, pero los proporcionados por defecto sólo son una de las dos, de forma que se puede personalizar al detalle cómo circula el flujo de información. Para una comunicación bidireccional son necesarios un par de conectores, un receptor y un emisor.

Los conectores proporcionados utilizan mensajes XMSG serializados como formato del mensaje. Para el uso de otros formatos es necesario crearse un conector propio que haga la transformación entre ambos protocolos. Además, estos conectores proporcionan la posibilidad de personalizar si se quiere aplicar cifrado de los mensajes y/o compresión de datos.

Los servicios conectores emisores generalmente disponen de más de una dirección lógica, esto les permite decidir qué tráfico o mensajes retransmitir. Usando comodines en la dirección lógica, pueden capturar todos los mensajes y retransmitirlos, o sólo los que cumplan un patrón. Esto permite definir canales separados de comunicación para evitar sobrecargas, definir canales redundantes para evitar caídas de red, reducir el tráfico entre las partes, o filtrar sólo los mensajes que el otro extremo del canal necesita recibir.

CHOREMED proporciona los siguientes conectores, que utilizan diferentes protocolos de comunicación: Protocolo de control de transmisión (Transmission Control Protocol, TCP), Protocolo de datagramas de usuario (User Datagram Protocol, UDP).

En la Figura 5 se puede ver cómo, mediante el empleo de conectores TCP, es posible crear una arquitectura que integra comunicación entre los servicios de coreografía y una aplicación de terceros externa, así como también proporciona una ejecución distribuida en la que dos coreógrafos ejecutándose en dos servidores distintos se comunican. Los servicios son independientes de la ubicación donde se ejecuta, se localizan mediante su dirección lógica. En una arquitectura como la de la Figura 5 los servicios de ambos servidores se comunican de forma transparente, sin ser conscientes de estar enviando mensajes a servicios que están en una ubicación físicamente distinta.

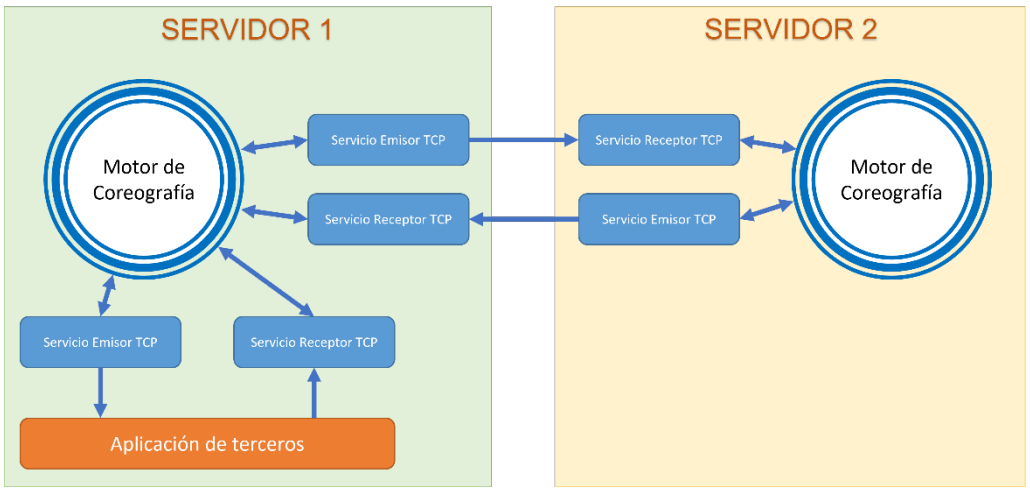

Figura 5: Arquitectura de comunicación distribuida y con sistemas de terceros usando conectores TCP.

Como se ha podido observar durante la introducción, el paradigma de coreografía de procesos surge como una solución a los problemas de integración de entornos heterogéneos en el marco de la eSalud. En los capítulos posteriores se describe cómo se ha aplicado esta tecnología para resolver distintos problemas que estos sistemas afrontan.

2. Hipótesis y aportaciones

2.1. Hipótesis

Esta tesis se apoya en una hipótesis principal y varias hipótesis secundarias:

Hipótesis principal:

- El paradigma de coreografía de procesos permite crear e integrar aplicaciones y plataformas software heterogéneas en el ámbito de la eSalud.

Hipótesis secundarias:

HS1. El uso del paradigma de coreografía de procesos permite crear sistemas de eSalud incorporando dispositivos tecnológicos complejos.

HS2. El uso del paradigma de coreografía de procesos ayuda a la integración de sistemas de terceros.

HS3. El paradigma de coreografía de procesos permite el desarrollo de plataformas de eSalud orientadas al Internet de la Cosas (IoT).

2.2. Aportaciones

En esta tesis se presentan tres casos de estudio que demuestran la viabilidad de las hipótesis planteadas. Los tres casos han sido presentados y validados por tres publicaciones en revista, revisado por pares y con alto factor de impacto. Para lograr esto, el doctorando ha trabajado durante todo su doctorado, realizando aportaciones tanto de tipo técnico como científico:

1. Como trabajo previo al uso del motor de coreografía CHOREMED en los casos de validación, se trabajó de forma activa en el desarrollo y mejora del motor, así como en sus herramientas. Entre las mejoras incorporadas, se realizó la portabilidad de plataforma de .NET 2.0 a .NET 4.5.
2. Para validar la primera de las hipótesis HS1, se realizó en 2016 la publicación de un artículo en la revista Sensors clasificada como Q1 y que lleva por título “Evaluation of Google Glass Technical Limitations on Their Integration in Medical Systems” [32]. Para lograrla se realizó el siguiente trabajo:
    a. Se trabajó con los directores de tesis buscando cómo plantear un caso de uso que permitiese validar la hipótesis HS1 y en cómo orientarlo para su publicación.

   b. Se llevó a cabo la definición de la arquitectura y de la coreografía del proceso.
   c. Se realizó el desarrollo de los nuevos servicios necesarios de tipo específico para el problema para generar DTOs y del servicio genérico de conector REST que se incorporó a CHOREMED.
   d. Se realizó la portabilidad del formato XMSG al formato JSON.
   e. Se colaboró en la experimentación y escritura del artículo.
3. Durante la participación de los directores de tesis y el doctorando en el proyecto MOSAIC, se realizó la aplicación del paradigma de coreografía para el desarrollo de la arquitectura del sistema. Esto permitió, a posteriori, realizar una nueva publicación de tipo Q1 en la revista Sensors en 2018, titulada "Integration of Distributed Services and Hybrid Models Based on Process Choreography to Predict and Detect Type 2 Diabetes" [33]. Para ello se realizó el siguiente trabajo:
   a. Se colaboró en la definición de la arquitectura y las coreografías de proceso necesarias.
   b. Se realizó el desarrollo de los servicios específicos para el sistema: servicio de acceso a datos mediante el software de terceros I2B2, servicio de autenticación y servicios específicos para los modelos de predicción.
   c. Se realizaron e incorporaron a CHOREMED servicios generalistas de ejecución de scripts de código en los motores estadísticos R y Matlab.
   d. Se mejoró el servicio de trazabilidad de CHOREMED
   e. Se amplió el formato de mensajería XMSG empleado en CHOREMED.
   f. Se desarrollaron los clientes web de consulta de datos.
   g. Se trabajó en el despliegue de la arquitectura.
   h. Se colaboró en la concepción y escritura del artículo.
4. Por último, durante la codirección del proyecto de fin de grado de la alumna Weisi Han, se desarrolló un sistema basado en coreografía de procesos para validar la última de las hipótesis planteadas HS3, dando origen a otra publicación de tipo Q1, que se presentó en 2018 en la revista Sensors con el título "Wearable Sensors Integrated with Internet of Things for Advancing eHealth Care" en la revista Sensors [34]. Para ello se realizó el siguiente trabajo:
   a. Formación de la alumna en coreografía de procesos, el uso de CHOREMED, el dispositivo Raspberry Pi y Windows 10 IoT.
   b. Diseño de la arquitectura y de las coreografías de procesos.
   c. Colaboración en la definición y creación de un protocolo para los sensores.
   d. Desarrollo del servicio genérico para CHOREMED de conector a puerto serie y servicios para alojar clientes web.
   e. Se coordinó la concepción y escritura del artículo, así como se colaboró en el diseño de los experimentos y análisis de los datos.

2.3. Metodología y estructura de la tesis

La redacción de la tesis se estructura sobre la presentación de tres casos de uso de aplicación de la coreografía de procesos, cada uno de los cuales permite validar una de las hipótesis secundarias planteadas y en su conjunto la hipótesis principal planteada.

Aunque todos los casos comparten parte de la metodología, cada uno de ellos contiene particularidades específicas al caso descrito, por lo que se ha optado por explicar las metodologías de cada caso de uso en su capítulo correspondiente.

En el capítulo 3, se presenta cómo se ha utilizado la coreografía de procesos para integrar y validar el dispositivo Google Glass. Este trabajo, fue presentado en 2016 en el artículo "Evaluation of Google Glass Technical Limitations on Their Integration in Medical Systems" en la revista Sensors [32].

En el capítulo 4, se presenta la utilización de la coreografía de procesos que permite crear y desarrollar, una arquitectura de integración de servicios distribuidos para aplicar modelos híbridos orientados a la predicción y detección de diabetes tipo 2. Dicho trabajo, se presentó en 2018 en el artículo "Integration of Distributed Services and Hybrid Models Based on Process Choreography to Predict and Detect Type 2 Diabetes" en la revista Sensors [33].

En el capítulo 5, se presenta el diseño y desarrollo de una arquitectura basada en coreografía de procesos para integrar sensores portables en un modelo de internet de las cosas en el ámbito de la eSalud. Este trabajo, se presentó en 2018 en el artículo "Wearable Sensors Integrated with Internet of Things for Advancing eHealth Care" en la revista Sensors [34].

Finalmente, en el capítulo 6, se discuten las conclusiones generales de la tesis, así como las líneas de trabajo futuras.

3. Integración y Validación de las Google Glass en el Ámbito Médico Aplicando Coreografía de Procesos

En este capítulo se presenta el estudio técnico de la integración de un *wearable*, como son las Google Glass, en el ámbito médico. Este tipo de dispositivos, denominados wearables, tienen cada vez mayor penetración en el ámbito de la salud por ser dispositivos portátiles de fácil uso, que permiten la captura o acceso a información. Sin embargo, se trata de dispositivos muy heterogéneos y, generalmente, de difícil integración en los sistemas, dado que cada fabricante implementa protocolos y mecanismos de comunicación diferentes.

El estudio que describe la integración de las Google Glass en el ámbito médico fue publicado en la revista Sensors en el año 2016 [32]. Si bien la publicación se centró en describir las limitaciones técnicas del dispositivo, el trabajo subyacente, y sobre el que se sustentan las pruebas realizadas permitió validar la primera hipótesis secundaria de la tesis HS1, la cual plantea que "*el uso del paradigma de coreografía de procesos permite crear sistemas de eSalud incorporando dispositivos tecnológicos complejos".* En este caso, un dispositivo como las Google Glass, las cuales se integran mediante un sistema basado en coreografía de procesos. Desarrollando servicios, no sólo para su integración, sino también para la realización de pruebas de funcionamiento.

Las Google Glass son un sensor portátil que se presenta para facilitar el acceso a la información y ayudar en la realización de tareas complejas. A pesar de la retirada temporal de soporte al producto por parte de Google, existen múltiples aplicaciones y numerosas investigaciones analizando el impacto potencial de esta tecnología en diferentes campos de la medicina. Las Google Glass satisfacen la necesidad de administrar y tener acceso rápido a información en tiempo real en diferentes escenarios de atención médica.

Entre las aplicaciones más comunes se encuentran el acceso a historias clínicas electrónicas, monitorizaciones de pantallas, apoyo a la decisión y consulta remota en especialidades, que van desde la oftalmología hasta la cirugía y la docencia. El dispositivo permite una interacción manos libres fácil de usar con sistemas de información de salud remotos y la transmisión de intervenciones y consultas médicas desde un punto de vista en primera persona. Sin embargo, la evidencia científica destaca importantes limitaciones técnicas en su uso e integración, como fallos en la conectividad, mala recepción de imágenes y reinicio automático del dispositivo.

Este capítulo presenta un estudio técnico sobre las limitaciones antes mencionadas (específicamente sobre la latencia, la confiabilidad y el rendimiento) en dos esquemas estándar de comunicación con el fin de categorizar e identificar las fuentes de los problemas. Los resultados han permitido obtener una base para definir los requisitos de las aplicaciones médicas para prevenir fallos de red, computacionales y de procesamiento asociadas al uso de las Google Glass. Además, se ha podido comprobar que la aplicación del paradigma de coreografía y uso del coreógrafo CHOREMED ha posibilitado integrar el dispositivo.

3.1. Introducción

En la actualidad es un hecho sobradamente conocido que la pirámide de población se ha invertido [1]. La población está cada vez más envejecida y los sistemas de salud están adaptando sus servicios para dar cobertura a una amplia gama de necesidades generadas por este cambio [2]. Una de las mejoras más importantes es el uso de las tecnologías de la información y las comunicaciones para apoyar las intervenciones y procesos médicos, disciplina denominada eHealth o eSalud [35]. En este contexto, las Google Glass (Google Inc., Mountain View, CA, EE. UU.) son un dispositivo que puede ser muy útil en aplicaciones médicas, ya que permite la presentación de información de forma sencilla y ergonómica, en tiempo real al usuario.

Las Google Glass fueron desarrolladas en Google X [36], una instalación dedicada al desarrollo de productos innovadores de alta tecnología. Los primeros prototipos databan de 2011 y eran muy pesados, pero, durante los años siguientes, el tamaño y el peso se redujeron drásticamente. El dispositivo se lanzó al mercado en 2013 para probadores beta y, un año después, en 2014, se abrió al público en general la versión Explorer Edition [37]. Desde entonces, las Google Glass comenzaron a recibir críticas e incluso acciones legales, principalmente por la preocupación por cuestiones de privacidad. Debido a esta controversia, Google Inc. anunció, en enero de 2015, que detendría su producción [38], pero se comprometió a continuar con el desarrollo del producto para crear una versión mejorada. Posteriormente sacó una versión Enterprise Edition en 2017 sólo para empresas, la cual renovó en 2019 con la Enterprise Edition 2, la cual mantiene su orientación a empresas, pero está disponible también para el público en general [39].

A pesar del retiro temporal de esa versión del producto, el concepto impulsado por Google Glass ha creado un impacto significativo. Por un lado, ha iniciado el desarrollo de pantallas montadas en la cabeza como Microsoft Hololens y Oculus Rift, y otras varias soluciones disponibles en etapas de mercado y pre-mercado. Por otro lado, ha realizado varios avances en su aplicación a la medicina general y dominios específicos [40], impulsando la explotación de dispositivos portátiles en el sector de la salud pública. En la revisión [41] se resumen algunas de las tecnologías clave que impulsarán las aplicaciones

clínicas de próxima generación. En dicho artículo se concluye que el campo de la tecnología portátil es una de las mayores oportunidades, mientras que existen grandes preocupaciones sobre la integración de varios tipos heterogéneos de tecnologías de la información y la comunicación.

La literatura ha recogido muchas experiencias de desarrollo, pruebas y evaluación de soluciones médicas utilizando las Google Glass en muchos campos de la medicina y la salud pública. Entre las aplicaciones más extendidas se encuentran la cirugía [42] [43] [44], la oftalmología [45] [46], la cardiología [47], las urgencias [48] y la atención a pie de cama [49]. La mayoría de los estudios muestran resultados prometedores, pero enumeran una serie de deficiencias al usar las Google Glass. Una de las principales limitaciones destacadas por los autores es la corta duración de la batería en uso autónomo, que es inferior a 60 min [50]. Asimismo, los estudios de evaluación y comparación con tecnologías similares (por ejemplo, Go Pro® Hero) revelan limitaciones en la calidad de las imágenes y la imposibilidad de indicar puntos de interés en las imágenes con una resolución tan baja, hecho que afecta a la interpretación de la imagen. Otras preocupaciones detectadas en los estudios están relacionadas con la seguridad y privacidad de los datos (imágenes, textos y sonidos) que Google Glass registra y almacena [51].

La transmisión de intervenciones médicas se erige actualmente como la aplicación médica más importante de las Google Glass. En 2014, un experto médico colorrectal retransmitió una resección hepática mientras respondía preguntas en vivo de una audiencia de más de 10.000 estudiantes de cirugía de más de 100 países [52]. Esto muestra el potencial para mejorar la atención al paciente, ya que se puede consultar a expertos relevantes en tiempo real sin límites geográficos, pero se ve seriamente afectado por las limitaciones de hardware y red, ya que la mayoría de los estudios citados aquejan fallos técnicos en la transmisión de imágenes, errores de comunicación (pérdida de conexión) y reinicios repentinos durante el uso del dispositivo.

Tales limitaciones están en el ámbito de varios campos de investigación [53] [54] sobre el desarrollo de nuevos paradigmas de comunicación y arquitecturas computacionales para optimizar indicadores clave de rendimiento como la latencia, la sobrecarga computacional y la calidad de los datos.

En el ámbito de la investigación realizada, las limitaciones en el uso médico de las Google Glass se originan en aspectos técnicos, pero aún se desconoce hasta qué punto estos aspectos técnicos se pueden prevenir.

En este estudio se presenta un análisis de las limitaciones técnicas de las Google Glass lo cual permitirá obtener información que ayude a los desarrolladores a minimizar el impacto de esas limitaciones en sistemas futuros. Se ha utilizado un banco de pruebas utilizando dos esquemas de comunicación bien establecidos para identificar y detectar las

características óptimas, que permiten un uso adecuado de las Google Glass en sistemas médicos utilizando coreografía de procesos.

El resto del capítulo está organizado en tres secciones principales. Primero, se describe el dispositivo Google Glass, la arquitectura basada en coreografía de procesos y las pruebas a realizar. Posteriormente, se describen los experimentos realizados y sus resultados. Por último, se realiza una discusión de los resultados y presentan conclusiones.

3.2. Materiales y métodos

3.2.1. Google glass Explorer Edition

Las Google Glass Explorer Edition son un dispositivo portátil basado en una versión especializada del sistema operativo Android. La pantalla de las Google Glass está equipada con un prisma y un haz de luz LED, que proyecta una imagen en la parte superior derecha del ojo derecho mediante filtros polarizadores y espejos semirreflectantes (Figura 6). Esta configuración proporciona al usuario la sensación de que la imagen flota dentro de un prisma transparente, sin limitar el campo de visión natural. Esta pieza es ajustable a la forma del rostro del usuario y la distancia al ojo. El dispositivo también incluye un transductor que transmite vibraciones al cráneo cerca del oído para transmitir el sonido, en lugar de usar el canal auditivo. El dispositivo ha incorporado un touch-pad lateral, próximo al prisma de reflexión, para interactuar con el dispositivo. Además, cuenta con una cámara digital 5 Mega Pixels que toma fotografías en formato jpg y graba videos de 720p en formato mp4. Está equipado con un procesador ARMv7 Dual 1.2 Ghz Texas Instrument, con 2GB de memoria volátil RAM y el almacenamiento disponible en el dispositivo es de 16GB de los cuales sólo 12GB están disponibles para el usuario. La autonomía se gestiona con una pequeña batería de polímero de litio de 570 mAh, con una capacidad operativa de menos de una hora en uso intensivo y de tres a cinco horas en uso normal.

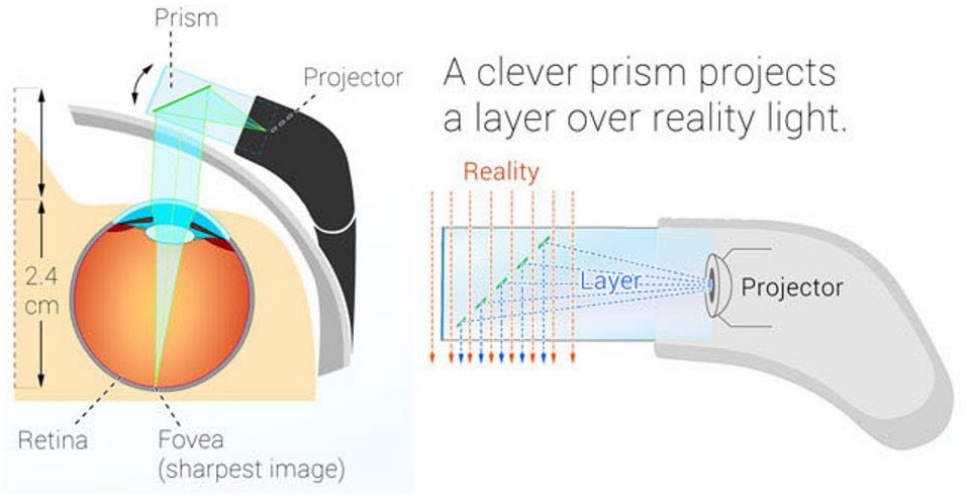

Figura 6: Esquema físico de las Google Glass. Fuente: Google Glass Inc.

Las Google Glass utilizan el sistema operativo "Jelly-Bean", que ejecuta aplicaciones Android Glass Ware, que funcionan de manera muy similar a los servicios de "Google Now".

La interfaz principal de las Google Glass se muestra en forma de menú horizontal o vertical (según el control de usuario activado). El panel táctil externo del lado derecho permite navegar por los elementos del sistema operativo y la página de inicio de las aplicaciones. El Glass Development Kit (GDK) permite la integración de todas las funciones para maximizar el uso de los recursos de hardware en el dispositivo, como el control por voz o las vistas en una interfaz de 640 × 360 píxeles. En cuanto a la conectividad, las Google Glass tienen tres interfaces: Wifi, Bluetooth y micro-USB. Para la elaboración de este estudio se utilizará el canal de comunicación Wi-Fi, ya que ha sido el utilizado en medicina según la literatura de estudio.

Para evaluar el rendimiento técnico de Google Glass, se han implementado y comparado dos esquemas de comunicación tradicionales: uno basado en una conexión bidireccional TCP [55] y otro que consiste en un esquema de comunicación de servicio web, basado en conectores REST [56].

Retrofit [57], una interfaz de programación de aplicaciones (API), se ha utilizado para construir el esquema de comunicación REST. Este esquema se basa en un cliente HTTP para Android y java, en el que se utiliza el método POST con la etiqueta @BODY para marcar los parámetros de entrada del método evaluado. Los paquetes REST se basan en el sistema de mensajería JSON y, para preservar la compatibilidad con las aplicaciones de Glass Ware, se ha utilizado GSON, una biblioteca de Google para administrar los mensajes JSON. Para el otro esquema, se usaron las API TCP predeterminadas del marco de Java para implementar el esquema de comunicación TCP. En este último caso, las gafas de Google disponían de una dirección privada de Protocolo de Internet (IP) en la configuración de la red.

REST es un protocolo implementado en HTTP (ISO Capa 7) y el protocolo de transporte TCP (ISO Capa 4). Por un lado, los mensajes REST se construyen con una mayor sobre carga y, por lo tanto, TCP debería proporcionar mejores indicadores de rendimiento, pero, por otro lado, como protocolo de capa de aplicación, REST permite una implementación óptima de aplicaciones en Google Glass.

3.2.2. Coreografía de procesos

En la realización de este trabajo se ha empleado y modificado el coreógrafo CHOREMED para adaptarlo a los requisitos del problema. Estas modificaciones se han centrado en actualizar el framework de trabajo a .NET 4.5, mejoras de eficiencia, el desarrollo de nuevos servicios y ampliación de los mecanismos de serialización del lenguaje de mensajería.

Entre las utilidades proporcionadas, las más utilizadas son las de serialización y deserialización de los mensajes. Mientras un mensaje está dentro del motor de coreografía es un objeto en memoria, pero, cuando se necesita almacenarlo o realizar comunicaciones entre el motor y un tercer software, es necesario definir un formato para serializar dichos mensajes. Actualmente CHOREMED cuenta con serializadores a formato binario, texto en lenguaje de marcado extensible (XML) y texto en formato JavaScript Object Notation (JSON). Siendo este último uno de los más utilizados por su alta compatibilidad con otras aplicaciones. A continuación (Figura 7), se muestra un ejemplo de serialización de un mensaje a formato texto JSON.

```
{
   "language": "XMSG-SOAP",
   "IdMessage": "4d9b661b032342f3a4ffb06e9968bf86",
   "sender": "sensores.movimiento.dormitorio",
   "receiver": "actuadores.luz.pasillo",
   "protocol": 0,
   "inresponse": "",
   "Content": {
      "method": "CambiarEstado",
      "parameters": {
         "Estado": {
            "type": "xs: string ",
            "value": “ON”
         }
      }
   }
}
```

Figura 7: Ejemplo de mensaje XMSG serializado a formato JSON.

Para la comunicación realizada en este estudio se requería de comunicación basada en transferencia de estado representacional (REpresentational State Transfer, REST). Por lo que se ampliaron las capacidades de CHOREMED con un nuevo conector basado en REST.

Partiendo del motor de coreografía que proporciona CHOREMED y algunos de sus servicios de conexión, se desarrolló el sistema utilizado para verificar los dos esquemas de comunicación propuestos en esta tesis, uno basado en TCP y otro en REST.

Para comunicar e integrar las Google Glass con el coreógrafo se podría haber definido y creado servicios conectores a medida para el dispositivo. Pero, dado que CHOREMED ya disponía de servicios de este tipo para integrar sistemas de terceros siempre que éstos

puedan leer y construir mensajes XMSG, se optó por utilizar dichos conectores y construir unos conversores JSON en la aplicación que se desarrolló para las Google Glass específicamente para este estudio.

La unidad de transferencia mínima u objeto de transferencia de datos (Data Transfer Object, DTO) utilizada fue un mensaje de coreografía de tipo "Solicitud" variando el contenido del mismo. Dichas pruebas de rendimiento y capacidad se realizaron conectando, tanto el servidor, como las Google Glass, a una Red inalámbrica local (WLAN) 802.11g estándar realizada mediante un enrutador D-LINK GO-R-Tn150 a fin de evitar interferencias del tráfico realizado por otros dispositivos.

Los servicios utilizados para construir las coreografías que permitieron estudiar cada esquema fueron:

- **Servicio Conector Emisor TCP:** Servicio proporcionado por CHOREMED, que se configuró para el envío de mensajes a las Google Glass.
- **Servicio Conector Receptor TCP:** Servicio proporcionado por CHOREMED, que se configuró para recibir de mensajes desde las Google Glass.
- **Servicio Conector Receptor REST:** Servicio creado e incorporado a CHOREMED, que se configuró para que las Google Glass pudiesen hacer peticiones de datos y obtener los datos pedidos. En una comunicación REST es el cliente quien conecta con el servidor para solicitar datos, el servidor nunca pude ser el iniciador de la comunicación.
- **Servicio para generar DTOs de texto de múltiples tamaños:** Este servicio se desarrolló expresamente para generar mensajes que contuviesen texto de un tamaño variable prefijado para las pruebas, además de generar el envío a la dirección lógica que se asignó a las Google Glass.
- **Servicio para empaquetar y enviar DTOs de imágenes:** Servicio que se desarrolló para leer imágenes de un conjunto dado, empaquetarlas en un mensaje de coreografía y enviar este a la dirección lógica de las Google Glass.
- **Servicio de trazabilidad de mensajes de coreografía:** Servicio proporcionado por CHOREMED que permite registrar, analizar y guardar en disco los mensajes de coreografía que distribuye el motor. Se utilizó para analizar los tiempos de respuesta y los errores generados.

La Figura 8 muestra la arquitectura de servicios construida en esta tesis para probar los dos esquemas de comunicación propuestos sobre las Google Glass.

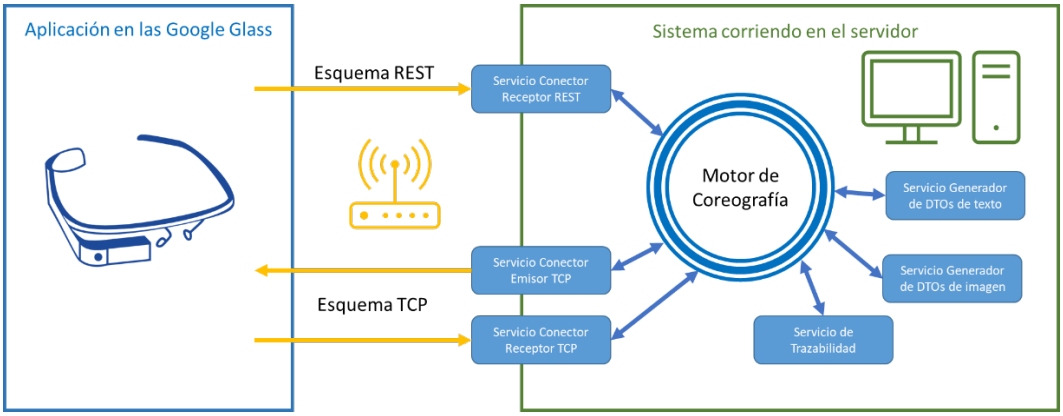

Figura 8: Esquema de comunicación utilizado para realizar los experimentos mediante la implementación del esquema TCP (abajo) y protocolos REST (arriba).

3.2.3. Definición de las pruebas comparativas

La comparación propuesta se basa en un conjunto pruebas comparativas sistemáticas que utilizan los dos esquemas de comunicación con grupos de 10, 100 y 1000 DTOs enviados gradualmente, que contienen mensajes alfanuméricos, generados aleatoriamente, con diferentes tamaños (0 valores, 1 valor, 10 y 100 valores). Para obtener valores más precisos, cada prueba fue realizada como un banco o conjunto de 10 repeticiones. Siendo los valores que monitorizar los siguientes indicadores clave de rendimiento (Key Performance Indicator, KPI):

- Retardo de comunicación (Latencia en milisegundos)
- Éxito de la comunicación (Confiabilidad en % de éxito)
- Carga Computacional en las Google Glass en cuanto a uso de Memoria y de la Unidad Central de Procesamiento (CPU) (Rendimiento en Megabytes (MB) y % de uso)

Asimismo, para evaluar las métricas de intercambio y procesamiento de imágenes, se ha utilizado la base de datos de imágenes médicas del Early Lung Cancer Action Program (ECLAP) de la Universidad de Cornell, que proporciona imágenes médicas en formato .jpg de varios tamaños. Debido a la distribución sesgada de los parámetros de latencia, una prueba de rango con signo de Wilcoxon al 95 % de I.C. se utilizará para evaluar la independencia de las diferencias intra e inter-esquemas. Se asumirá para la significación estadística una $p < 0,05$. El análisis estadístico y gráfico se realizó utilizando la versión Matlab 2016R utilizando una Licencia Académica.

3.3. Experimentos y resultados

Una vez desarrollados los nuevos servicios y configurada la coreografía de procesos se desplegó el sistema, tanto en el ordenador, como en la Google Glass. Esto permitió cuantificar las características de retraso (latencia), la confiabilidad y el rendimiento del dispositivo, que depende del número de mensajes y del tamaño de las imágenes intercambiadas.

Para cada banco de pruebas (definido como un conjunto de 10 repeticiones del intercambio de mensajes), el coreógrafo se utilizó para enviar los DTOs y registrar los KPIs para los dos esquemas REST y TCP. Para cada iteración sobre el número de DTOs y el tamaño del contenido de los mensajes, se registró una matriz bidimensional con un identificador de experimento. Se implementó un script de Matlab para analizar automáticamente la matriz y hacer comparaciones y gráficos estadísticos.

Tal y como se esperaba, la biblioteca GSON permitió convertir tipos de datos sin configuraciones adicionales. Para estos experimentos, se desarrolló un método personalizado de serialización y deserialización para convertir objetos complejos (imágenes) en secuencias alfanuméricas para encapsularlos en mensajes JSON.

3.3.1. Retraso en la comunicación

Los valores de latencia para los esquemas REST y TCP se muestran en la Figura 9. Un aumento en la cantidad y el tamaño de los mensajes se correlaciona con un aumento en el tiempo de recepción hasta un valor crítico, dependiente del tamaño del contenido del mensaje a partir del cual se reduce. El esquema TCP es significativamente más rápido que el protocolo REST ($p < 0,01$), así como la distribución de los resultados para cada banco y tamaño ($p < 0,05$) (Prueba de rango con signo de Wilcoxon para un I.C. = 95%).

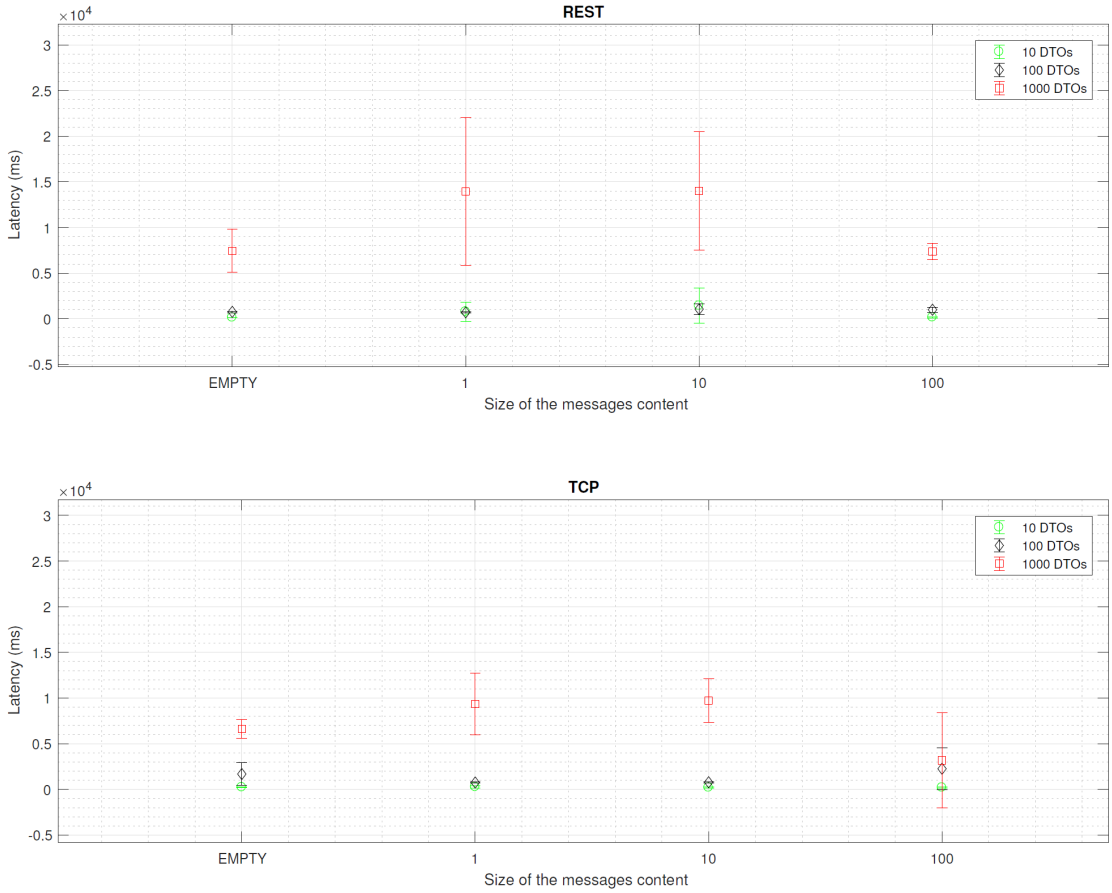

Figura 9: Latencia en milisegundos para el esquema REST y TCP.

3.3.2. Éxito de la comunicación

En cuanto a la fiabilidad del esquema REST, éste alcanzó el 96,62% ± 2,35%, mientras que TCP alcanzó el 96,90% ± 1,71% con un valor de p > 0,05 (p = 0,882), que impide confirmar qué esquema es el más fiable (Tabla 1).

Tabla 1: Porcentaje de éxito en las comunicaciones (Confiabilidad) para los experimentos del esquema REST para cada bloque de Objetos de transferencia de datos (DTO).

Message Size	REST			TCP		
	10 DTOs	100 DTOs	1000 DTOs	10 DTOs	100 DTOs	1000 DTOs
0	100	98	95.49	98	95.75	98.34
1	100	98	96.76	98	98	93.7
10	94	98	93.85	98	97.9	94.64
100	94	98	93.85	98	97.9	94.64

3.3.3. Carga computacional

La Figura 10 muestra una comparación de la carga computacional en las Google Glass para los dos esquemas de comunicación. Existen diferencias significativas para el uso de CPU ($p < 0.01$) y Memoria ($p < 0.05$) para cada esquema de comunicación. Como era de esperar, el esquema TCP necesita menos recursos para procesar los DTOs, ya que los mensajes requieren menos datos que el REST, que debe incorporar todas las cabeceras y datos de los protocolos HTTP y REST. Sin embargo, la forma exponencial de los resultados, a medida que aumenta la cantidad de DTOs, confirma la hipótesis sobre la excepción OutOfMemory de la gestión de memoria.

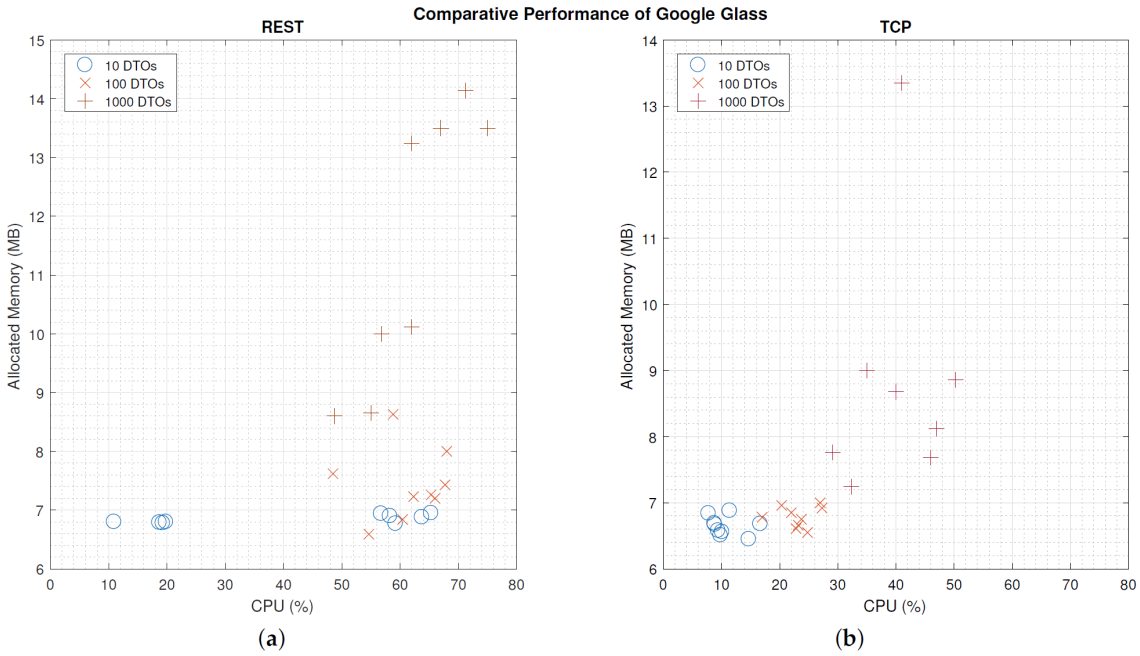

Figura 10: Comparación del desempeño de Google Glass (CPU y Memoria) para los dos esquemas de comunicación. La pendiente exponencial de los puntos de cruce confirma nuestra hipótesis en la gestión de la memoria y la excepción OutOfMemory, aunque la figura (b) muestra una mejor gestión de la CPU que (a). a) resultados sobre el rendimiento de la gestión de mensajes REST; (b) Resultados sobre el desempeño de la gestión de mensajes TCP.

3.3.4. Procesado de Imagen

Se envió un conjunto total de 64 imágenes médicas, con tamaños que van desde 100 KB hasta más de 2 MB, utilizando el esquema de comunicación TCP. Las métricas de rendimiento, como el uso de la CPU y la memoria, fueron registrados, así como el retraso en el procesamiento de la imagen a su llegada. De las 64 imágenes entregadas, 46 se mostraron correctamente y las 20 imágenes restantes colapsaron la memoria del

dispositivo, provocando un reinicio automático. Para presentar los resultados, se ha clasificado la imagen en tres categorías: ligera (menos de 1 MB de tamaño), media (entre 1 y 2 MB de tamaño) y pesada (más de 2 MB de tamaño).

Tabla 2: Tasa de fallos en función del tamaño de fichero.

Tamaño Imagen	Nº Envíos	Nº Fallos	% Fallos
Ligera	21	4	0,190476
Media	31	9	0,290323
Pesada	12	5	0,416667

Como se puede observar en la Tabla 2, la tasa de fallos aumenta proporcionalmente al tamaño de la imagen, tal y como cabría esperar. No obstante, también se observó que cuando el uso de la CPU del dispositivo supera un valor de umbral (alrededor del 39,78 % ± 0,35 % en nuestros experimentos), el recolector de basura no puede administrar el uso de recursos y el dispositivo realiza un reinicio automático (Figura 11).

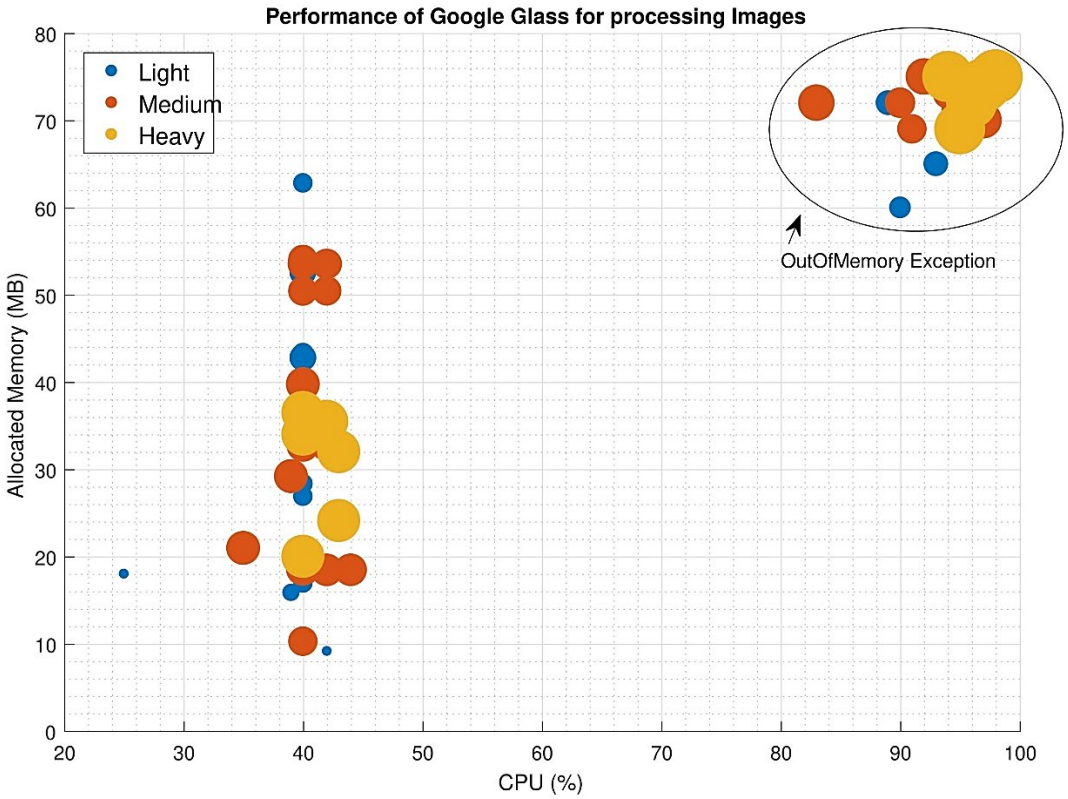

Figura 11: Rendimiento (CPU y uso de memoria) de las Google Glass al procesar imágenes de distinto tamaño. El área de los círculos está en proporción con el tamaño de la imagen.

Con respecto al tiempo que necesitan las Google Glass para procesar las imágenes (Delay), en la Figura 12 se nota una relación lineal entre el tamaño de la imagen y el retraso. Según nuestros experimentos, a medida que la imagen se serializa, antes de enviarse en el esquema TCP, el dispositivo realiza una operación de almacenamiento en búfer para analizar y deserializar el objeto, y luego transformarlo en un objeto de imagen para ser proyectado en el prisma. Aproximando la distribución de la Figura 12 con una función lineal ((3.1), (R^2 = 95.1%), se obtiene un modelo para dicho retardo (en el que el tamaño de la imagen está en MB y el tiempo de retardo está en milisegundos).

$$delay = 3786\ x\ size + 2975$$

(3.1)

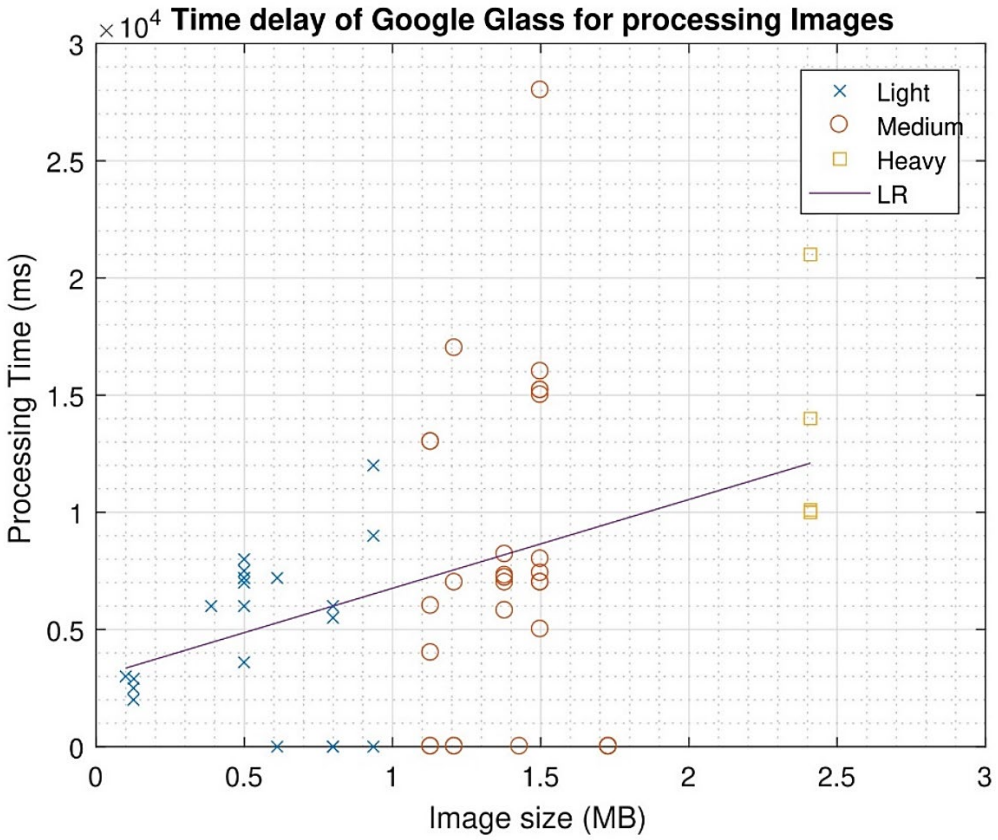

Figura 12: Relación entre el tamaño de la imagen y el tiempo de procesamiento de las Google Glass para mostrar la imagen en el prisma. Las pruebas fallidas se marcan con un retraso de 0 (superpuesto con el eje x). La bondad de ajuste de la regresión lineal (LR) tiene un R2 = 95,1%.

3.4. Conclusión y discusión

En este apartado se explica cómo las Google Glass han sido integradas en un sistema software empleando coreografía de procesos, para después analizar y cuantificar las limitaciones técnicas más extendidas de las Google Glass en aplicaciones de salud pública, en relación con estudios publicados anteriormente. El estudio ha evaluado métricas objetivas, como la latencia, la confiabilidad y la carga computacional al enviar varios paquetes de datos de diferente tamaño e imágenes también de diferentes tamaños.

La implementación de la arquitectura basada en coreografía de procesos ha funcionado sin problemas, ayudando en la creación del sistema en un corto periodo de tiempo. Esto es debido a la utilización de servicios sencillos, que ofrecen solución a una parte del proceso para luego ir creciendo en complejidad. Se definió la coreografía del proceso y los servicios necesarios, decidiendo cómo debían interactuar para obtener la funcionalidad necesaria.

Una vez definidos qué servicios son necesarios, el uso de CHOREMED ha permitido llevar a cabo la experimentación de este capítulo, mediante la implementación de únicamente 3 servicios específicos para intercambio de información con las Google Glass. Esto es debido a que se pudieron reutilizar varios servicios ya existentes, como los conectores TCP, que, configurándolos adecuadamente y utilizando el formato de mensaje XMSG, ha permitido interactuar con las Google Glass. Del mismo modo, también se han podido reutilizar los servicios de trazabilidad empleados para medir indicadores de tiempo o error, facilitando las tareas de validación y evitando el desarrollo de código extra para las mediciones. La reutilización de servicios realizada en el desarrollo del sistema es una de las ventajas de las arquitecturas SOA mencionadas en el punto 1.5. En esta línea, los nuevos servicios desarrollados para integrar las Google Glass podrían ser reutilizados en otros sistemas de eSalud en el futuro.

Respecto a los resultados de la evaluación de las Google Glass, se confirma que la pérdida de conexión es un problema importante, y los resultados permiten suponer que el fallo real experimentado en la literatura es un error de memoria tipo OutOfMemory, ya que la memoria del dispositivo es muy limitada y la aplicación se cierra automáticamente, evitando así mensajes de alerta al usuario antes de volver al menú principal. Por otro lado, el análisis comparativo de la tasa de éxito de comunicación de los distintos esquemas planteados no permite confirmar cuál es el más adecuado para otorgar un nivel de éxito aceptable, sin embargo, los resultados indican que los mensajes de gran tamaño con múltiples DTOs tienen más probabilidades de experimentar transmisiones fallidas, ya que están expuestos a las fluctuaciones de la red.

Los mensajes REST presentan una mayor sobrecarga de datos que los TCP. Por lo tanto, el estudio sobre el retraso de la comunicación muestra una latencia significativamente menor para los TCP. No obstante, los mensajes REST son paquetes dirigidos a aplicación,

lo que permite el uso de APIs para un desarrollo más sencillo en las Google Glass para mostrar los resultados a los usuarios (Figura 10).

Se ha demostrado que las Google Glass se ven afectadas por los mecanismos de optimización de la máquina virtual Java, mostrando que, a partir de un determinado umbral de gasto de memoria, optimizan el rendimiento para mensajes grandes. En la Figura 9, la prueba para 100 DTOs de mayor tamaño presenta una latencia significativamente menor a la esperada. Esto, probablemente, sea debido al tamaño del mensaje. El dispositivo activa los métodos del recolector de basura para reasignar memoria a medida que disminuye la carga computacional en la CPU.

El esquema TCP ofrece un mejor desempeño en términos de uso de CPU ($p < 0.01$) y Memoria ($p < 0.05$), pero los resultados muestran una tendencia exponencial en el uso de estos recursos a medida que aumenta el número de DTOs. Por lo tanto, aunque sería recomendable un esquema TCP, es obligatorio administrar la recolección de basura en la memoria del dispositivo antes de enviar paquetes.

En estas pruebas, centrándose en el comportamiento de las Google Glass para enviar/recibir mensajes con múltiples tamaños y múltiples DTOs, se han mostrado las limitaciones en la capacidad de procesamiento de las Google Glass. La principal limitación es la cantidad de memoria que pueden utilizar las aplicaciones. Además, también se ha medido el retraso de la conexión entre servidor y dispositivo enviando mensajes vacíos. Hay que mencionar que todas las pruebas se han realizado en una red de área local inalámbrica dedicada específicamente a los experimentos y, por lo tanto, los resultados de las pruebas no se ven afectados por la carga de la red externa. Sin embargo, se puede concluir que el tamaño del mensaje enviado al dispositivo es un factor crucial, que debe monitorizarse para ayudar a los desarrolladores de aplicaciones médicas a superar las limitaciones técnicas de las Google Glass.

Como era de esperar, el procesamiento de imágenes es un problema importante que depende del tamaño de la imagen, al menos para la gestión de la CPU y la memoria. Las deficiencias observadas en la literatura están vinculadas a una gestión débil de los recursos de memoria local. Pero los experimentos realizados demuestran que, no sólo el tamaño de la imagen está vinculado a una excepción OutOfMemory, sino que también influye la carga de CPU y la gestión que realiza el recolector de basura de Java. Sin embargo, el tiempo necesario para procesar las imágenes tiene una relación lineal con el tamaño de la imagen, por lo que se ha obtenido una Ecuación (3.1) que puede ayudar a los investigadores a identificar el intervalo de tiempo mínimo en la entrega de la imagen para evitar fallos.

Las tecnologías móviles y portátiles están penetrando progresivamente en aplicaciones específicas de atención de la salud [41]. Uno de los campos más prometedores para el uso de las Google Glass y otras pantallas montadas en la cabeza es la formación médica [58],

en las que el dispositivo se utiliza para grabar a los residentes durante las lecciones prácticas y el entrenamiento en el campo, proporcionando nuevas perspectivas para el análisis y evaluación de sus habilidades manuales y de comunicación interpersonal sin la sensación de ser observado.

Sin embargo, como se ha analizado en este artículo, las Google Glass tienen deficiencias críticas que, si no se previenen, pueden provocar una pérdida de funciones en los sistemas de atención médica. Hay una serie de preocupaciones relacionadas con el uso general de los recursos de software y hardware, en particular con el rendimiento del procesamiento de imágenes y el retraso en las comunicaciones.

Para la interpretación de señales continuas en tiempo real (es decir, ECG) o teleconferencias remotas, en las que el retraso, la confiabilidad y el tiempo de procesamiento son factores críticos, las Google Glass tienen límites que, probablemente, se pueden manejar mediante un diseño e implementación de software adecuados, y, además, se espera que sean mejoradas en futuras versiones.

Un tema importante que no se ha abordado, común a otras tecnologías móviles, es la duración de la batería, que en uso continuo no dura más de una hora, y debe tenerse en cuenta al planificar el uso de las Google Glass en aplicaciones para el cuidado de la salud.

Por lo tanto, las Google Glass no son adecuadas para llevar a cabo aplicaciones complejas y pesadas, como el análisis de datos de gran tamaño, el procesamiento de imágenes o el procesamiento de múltiples hilos. De igual forma a la recomendación realizada por Nguyen and Gruteser en su artículo sobre el campo de la tecnología móvil [59], se recomienda usar las Google Glass como un dispositivo portátil para registrar y mostrar información, usando en su lugar servidores dedicados para albergar la ejecución de tareas complejas y pesadas.

El tiempo es un factor crítico en el cuidado de la salud y las Google Glass puede ayudar a los profesionales de la salud a tener un acceso rápido y nuevos caminos a la información, pero, como toda tecnología médica, tiene que ser confiable.

La latencia, la fiabilidad y el rendimiento de las Google Glass dependen de las características técnicas del software del dispositivo y del esquema de comunicación. Es fundamental conocer el tipo de tráfico que soportará una aplicación clínica dentro de un sistema médico y, además, saber cuáles son los requisitos del sistema para garantizar un uso eficaz del dispositivo Google Glass. TCP permite niveles de latencia más bajos que REST y, por lo tanto, las tasas de CPU y memoria siguen siendo mejores para TCP. La gestión de datos y de imágenes médicas debe optimizarse para evitar que la aplicación se bloquee debido a excepciones de memoria.

El correcto funcionamiento del sistema durante toda la experimentación, así como la facilidad y rapidez con la que se ha podido integrar el dispositivo Google Glass para su

validación, demuestran la hipótesis HS1 planteada, que “el paradigma de coreografía de procesos permite crear sistemas de eSalud incorporando dispositivos tecnológicos complejos”.

Posteriormente a la publicación de esta investigación, Google realizó la puesta en el mercado de dos modelos superiores de las Google Glass. La primera de ellas, las Google Glass Enterprise Edition, en julio del 2017 el cual sólo estaba disponible para empresas. La segunda de ellas, las Google Glass Enterprise Edition 2, en mayo del 2019, modelo que en este caso sí es posible comprar por el público en general, aunque, dado su elevado precio, superior a los 1200€, están claramente enfocadas al uso por empresas o desarrolladores.

Las Google Glass Enterprise Edition 2 mejoran notablemente aspectos técnicos del primer modelo en varios puntos:

- Se mejoran los materiales plásticos y metálicos sustituyéndolos por titanio y resinas de nailon.
- Incorpora un procesador Qualcomm Snapdragon XR1 quad core a 1.7GHz.
- Ya no emplea un Sistema operative adaptado, si no Android Open Source Project 8.1 (Oreo), como el usado en móviles.
- Se le añade resistencia a salpicaduras y polvo. Norma IP53.
- La cámara pasa de 5Mpx y video 720p a 8Mpx y video 1080p
- La memoria RAM pasa de 2GB a 3GB
- El almacenamiento aumenta de 16GB a 32GB
- El altavoz conductivo se sustituye por un altavoz integrado.
- La conectividad Wifi 8012b/g pasa a ser Wifi 802.11a/g/b/n/ac
- La conectividad Bluetooth pasa a ser Bluetooth 5.0
- La batería aumenta de los 520mAh a 800mAh con carga rápida.
- El conector de carga y datos pasa a ser USB tipo C.

Las mejoras realizadas en el procesador, orientadas a su uso en cálculos de realidad aumentada e inteligencia artificial, así como la mejor gestión de memoria RAM hacen pensar que la nueva versión soportará una mayor carga de trabajo y de forma más estable, sin tantos fallos de memoria.

El uso de una versión de sistema operativo igual a la empleada en dispositivos móviles, en lugar de una versión adaptada, hace pensar que el dispositivo sea más estable y no surjan problemas con el recolector de basura que se pudieron observar en las pruebas. Y, dado que se ha mejorado la conectividad Wifi, debería mejorar los tiempos de respuesta.

Por último, algo muy importante para implantar su uso en el trabajo diario, la batería ha pasado de unas teóricas 5 horas de funcionamiento a 8 y, además, dispone de carga rápida.

4. Integración de Servicios Distribuidos y Modelos Híbridos Basados en Coreografía de Procesos para Predecir y Detectar Diabetes Tipo 2

En este capítulo se presenta el estudio técnico de la utilización de tecnología de coreografía para crear la arquitectura de un sistema, que permite la integración de servicios distribuidos e híbridos para la evaluación de escalas de riego asociadas a la enfermedad de diabetes tipo 2. Dicho estudio se presentó en una publicación en la revista Sensors en el año 2018 [33] y permitió validar el segundo de los objetivos secundarios de esta tesis HS2, que dice *"el uso del paradigma de coreografía de procesos ayuda a la integración de sistemas de terceros".*

La esperanza de vida es cada vez mayor y, por tanto, los años de vida de los pacientes con enfermedades y comorbilidades. La diabetes tipo 2 es una de las enfermedades crónicas más prevalentes, específicamente relacionada con el sobrepeso y edades altas superiores a los sesenta años. Estudios recientes han demostrado la eficacia de nuevas estrategias para retrasar e incluso prevenir la aparición de diabetes tipo 2, mediante una combinación de un estilo de vida activo y saludable en cohortes de sujetos de riesgo medio/alto. La investigación prospectiva ha sido impulsada en grandes grupos de la población para construir escalas de riesgo, que tienen como objetivo obtener una regla para la clasificación de los sujetos según la probabilidad de desarrollar la enfermedad. Actualmente hay más de doscientos modelos y escalas de riesgo para hacer esto, pero pocos han sido evaluados adecuadamente en grupos externos e integrados en una aplicación clínica para apoyo a la toma de decisiones. En este trabajo se presenta una novedosa arquitectura de sistema basada en coreografía de procesos y modelado híbrido, que permite una integración distribuida de bases de datos clínicas, motores estadísticos y matemáticos e interfaces web, para ser desplegados en un entorno clínico. El sistema se evaluó durante un período continuo de ocho semanas, con ocho endocrinólogos de un hospital, que evaluaron 8080 pacientes con siete modelos diferentes de riesgo de diabetes tipo 2, implementados en dos motores matemáticos. Finalmente, se evaluó el rendimiento global a partir de un conjunto de indicadores técnicos.

4.1. Introducción

La diabetes es un conjunto de trastornos patológicos relacionados con una alteración en la producción y/o acción de la insulina [60]. Específicamente, la Diabetes Mellitus Tipo 2 (DM2) se caracteriza, tanto por una resistencia a la acción de la insulina, como por una disfunción progresiva del proceso de liberación de insulina endógena. Se diferencia de otros tipos de diabetes por el factor desencadenante, que se relaciona con el estilo de vida poco saludable y el defecto a largo plazo originado por el envejecimiento [61]. La prevalencia de DM2 está aumentando rápidamente en todo el mundo [62]. En 2013, había 382 millones de personas con DM2 y hay estimaciones de que la proporción de diabetes no diagnosticada representa el 30% de la población mundial [63].

La prueba diagnóstica para confirmar la DM2 se basa en la comparación de pruebas de laboratorio y rangos específicos [64]. Aunque la glucosa en ayunas y la HbA1C se utilizan para identificar sujetos con alto riesgo de adquirir DM2, la prueba de referencia es la tolerancia oral a la glucosa a las 2 h (2h-OGTT). En esta prueba, el sujeto ingiere una dosis de 75g de glucosa diluida en 3dL de agua (concentración <25 g/dL) por vía oral en menos de 5 min. Previo a la prueba, el sujeto debe seguir una prescripción alimentaria específica, abstenerse de fármacos relacionados con la glucosa y realizar un ayuno de 8 horas previo a la prueba.

Los investigadores clínicos y los epidemiólogos se esfuerzan por producir algoritmos de clasificación y modelos predictivos para comprender por qué las personas desarrollan este tipo de diabetes [65] [66]. Los beneficios de la detección temprana de etapas prediabéticas están ampliamente confirmados por la literatura [67]. En este contexto, el uso de técnicas de modelado se ha vuelto popular, con una amplia gama de algoritmos de investigación para detectar individuos con un alto riesgo de desarrollar DM2 [68]. Aunque, en los países europeos, los cuestionarios de detección siguen usándose ampliamente como fuente de datos, existe un conjunto de registros de salud electrónicos en continuo crecimiento, tanto en la atención primaria, como en la secundaria, que podrían usarse para desarrollar y validar algoritmos predictivos [69].

Una escala de riesgo de DM2 tiene que estimar con precisión el riesgo de que un sujeto desarrolle DM2 [70]. Esta escala puede basarse en una discriminación numérica, que asigna un valor numérico a un individuo, o en una predicción de riesgo cualitativa, sobre la base de una probabilidad alta, media y baja de desarrollar DM2 en el futuro. Los algoritmos de discriminación y predicción son modelos estadísticos, que combinan información de varias fuentes de datos clínicos y de estilo de vida. Los tipos más comunes de modelos incluyen modelos de regresión logística, redes bayesianas, máquinas de soporte vectorial, modelos de riesgos proporcionales de Cox y árboles de clasificación [71]. Cada tipo de modelo produce una estimación del riesgo de un individuo basado en sus datos específicos. Sin embargo, varios factores pueden hacer que una escala de riesgo

tenga un desempeño deficiente cuando se aplica a otros individuos e, incluso, a otras poblaciones [68]. Puede suceder que la predicción de un modelo no sea reproducible debido a deficiencias en los datos de referencia (valores faltantes, datos erróneos), o en los métodos de modelado utilizados en el estudio del que se deriva el modelo, principalmente, debido a sobreajuste, diferencias entre las características de los pacientes, métodos de medición, particularidades de los sistemas de salud o calidad de los datos [72].

La validación de la escala de riesgo requiere una especificación completa del modelo existente (es decir, tanto las variables de entrada como los parámetros del modelo) para predecir el resultado. Dicha especificación también debe incluir la estrategia de desarrollo (entrenamiento y validación) y, si corresponde, la comparación de las predicciones del modelo y los resultados reales del paciente (análisis de discriminación). En la práctica clínica se utilizan pocos modelos predictivos, muy probablemente, debido a la falta de validación externa [73] [74]. Además, la mayoría de los modelos publicados en la literatura requieren de la recogida de datos, que pueden no estar disponibles en el sistema sanitario, ya que se obtienen en el marco de la ejecución de un ensayo clínico [68] [71].

Una escala de riesgo debe ser clínicamente creíble, precisa (bien calibrada y con buena capacidad de discriminación), generalizable (es decir, externamente validada) e, idealmente, debe demostrar su eficacia clínica; es decir, proporcionar información adicional útil a los profesionales que mejore la toma de decisiones y, por lo tanto, el resultado para el diagnóstico del paciente [74]. Es crucial cuantificar el rendimiento y la relevancia de un modelo predictivo en una nueva serie de pacientes antes de aplicar el modelo en la práctica diaria para guiar su diagnóstico [75]. Existen varios criterios para evaluar la selección de una herramienta de apoyo a la decisión, que deben incluir indicadores ampliamente conocidos de efectividad (sensibilidad y valor predictivo), poder predictivo y la posibilidad de ser aplicada a todas las categorías de riesgo [76]. Además, también se debe considerar su accesibilidad para el personal clínico, la posibilidad de evaluar a lo largo de la línea de tiempo (proporcionar una línea base para evaluar la intervención a lo largo del tiempo sus costes) así como la facilidad de uso.

La combinación de diferentes técnicas de modelado permite obtener modelos de mayor rendimiento que cada uno de ellos por separado [77]. El modelado híbrido consiste en mezclar diferentes enfoques de modelado sobre un conjunto de datos de alta dimensión para maximizar la probabilidad de discriminación [78], esta técnica se usa ampliamente con fines de investigación y para producir escalas de riesgo de DM2 [79] [80]. Sin embargo, la implementación real de tales modelos híbridos sigue siendo un desafío en entornos clínicos, debido a la confluencia de numerosos factores relacionados con el marco de trabajo tecnológico, como el acceso a motores matemáticos y la calidad de los datos, que dificultan su aplicación para identificar pacientes de alto riesgo de manera confiable.

Con este fin, se propone el uso de una arquitectura heterogénea distribuida como una solución para resolver los problemas descritos anteriormente. En este contexto, la especificación de un modelo suele ser abordada por matemáticos y bioestadísticos; luego, los diseñadores e ingenieros informáticos lo implementan usando un entorno/software o librería específica y, finalmente, los médicos lo utilizan en una aplicación web o de escritorio.

En este artículo se presenta una arquitectura novedosa para superar las principales limitaciones de la validación de modelos de discriminación y predicción. El objetivo principal de este trabajo es proporcionar una plataforma capaz de transferir la investigación clínica sobre las escalas de riesgo de DM2 a un entorno real y promover la medicina basada en la evidencia. El enfoque consiste en utilizar un repositorio de datos común, que integre varias fuentes de datos reales del Sistema de Información Hospitalaria (HIS) y crear un sistema en el que los componentes independientes puedan ejecutarse de acuerdo con un flujo de trabajo predefinido y ser utilizados por los endocrinólogos.

Este trabajo describe en detalle una versión funcional de un sistema de soporte de ayuda a la toma de decisiones, compuesto por una arquitectura distribuida, algoritmos de modelado matemático y protocolos para el intercambio de información clínica. El sistema se probó en un piloto clínico para evaluar la viabilidad, confiabilidad y efectividad de integrar escalas de riesgo en las instalaciones clínicas mediante el registro de distintos parámetros.

Este capítulo está estructurado de la siguiente manera. En primer lugar, se presentan los antecedentes de las técnicas para el modelado de datos en las escalas de riesgo de DM2, las necesidades de infraestructura de datos y el contexto empresarial. Posteriormente, se presenta la especificación arquitectónica y la descripción de la implementación, mostrando también los resultados del ensayo clínico. El capítulo concluye revisando los resultados obtenidos y brindando pautas para el trabajo futuro.

4.2. Materiales y métodos

El objetivo principal de esta investigación es proporcionar una estructura tecnológica en la que la investigación clínica se pueda aplicar directamente a los pacientes para la toma de decisiones basadas en la evidencia médica. Para tal fin, se ha definido, implementado y evaluado una arquitectura distribuida, basada en el paradigma de coreografía, capaz de integrar modelos híbridos para discriminar pacientes con alto riesgo de desarrollar DM2.

Sackett define la práctica de la medicina basada en la evidencia como "*un proceso de aprendizaje autodirigido de por vida en el que el cuidado de los pacientes crea la necesidad de información clínicamente relevante sobre el diagnóstico, el pronóstico y la terapia*"

[81]. Tal paradigma tiene que: (1) usar los datos para responder preguntas de relevancia clínica; (2) rastrear la mejor evidencia para responderlas; (3) evaluar críticamente esa evidencia por su validez (cercanía a la verdad) y utilidad (aplicabilidad clínica); (4) integrar esta evaluación con nuestra experiencia clínica y aplicarla en la práctica; y (5) evaluar su desempeño. Adoptando esta definición, primero se tuvo que definir el contexto empresarial, con una adecuada identificación de las partes interesadas y su entorno.

4.2.1. Definición de contexto empresarial

El contexto empresarial del sistema para la ejecución de modelos de riesgo de DM2 en entornos clínicos debe tener en cuenta, por un lado, las partes interesadas y, por otro, los servicios ofrecidos (funcionalidades).

Las partes interesadas son los roles abstractos, que utilizan el sistema desde diferentes perspectivas y para diferentes propósitos (puntos de vista). Éstas son:

- **Usuarios finales:** usuarios finales no técnicos, como profesionales relacionados con el cuidado médico, gestores de mutuas, pacientes y ciudadanos; usuarios técnicos, como profesionales de la salud, gerentes y creadores de políticas de salud, e investigadores médicos relacionados con la salud pública. Los buenos entornos de desarrollo y las interfaces amigables conducen a un software de mejor calidad y atraen a los profesionales para que utilicen las herramientas. La comunicación eficiente entre los proveedores de servicios da como resultado servicios que satisfacen mejor los requisitos del usuario final.

- **Los proveedores de servicios:** Éstos están preocupados por la explotación comercial del sistema. Necesitan mantener una comunicación efectiva con sus usuarios finales y una interacción fluida con el entorno en tiempo de ejecución para explorar las posibles integraciones.

- **Los investigadores técnicos:** Se preocupan, principalmente, por disponer de buenos entornos de desarrollo, una comunidad de desarrolladores bien informada y acceso a recursos para implementar sus algoritmos. El sistema debe apoyar a los investigadores como parte interesada principal y permitirles participar en la mejora del sistema (junto con los proveedores de servicios y los usuarios finales). Dentro de este punto de vista se encuentran dos dominios principales de investigación: la investigación en minería de datos y la investigación en software. El primer tipo se centra en el desarrollo de nuevos algoritmos y modelos para realizar análisis de estratificación y asociación de variables. El segundo tipo tiene como objetivo mejorar la calidad de los servicios, las interfaces y la gestión de la base de datos.

Los requisitos del sistema pueden convertirse en funcionalidades y clasificarse en módulos, que son las entidades que prestan servicios y operan dentro del sistema. Estos módulos pueden ofrecer servicios para ser consumidos entre ellos o directamente por las partes interesadas:

- **El módulo de almacenamiento de datos "Data Storage":** Se encarga de proporcionar un almacén para todos los datos dentro del sistema. Desde un punto de vista conceptual, el modelo de datos es único para todo el sistema, conteniendo Historias Clínicas Electrónicas (Electronic Health Records, EHR) y otro tipo de datos (logísticos y administrativos).

- **El módulo central de gestión de los modelos "Model Host":** Es el núcleo del sistema. Está a cargo de administrar las solicitudes de los clientes (interacciones de los usuarios), ejecutar las escalas de riesgo y consultar los almacenes de datos. Reúne en un servidor de aplicaciones las herramientas (modelos), que ejecutarán los algoritmos sobre los datos de los EHR, y proporciona los servicios para administrarlos desde el lado del cliente. El módulo Model Host también contendrá componentes para proporcionar servicios horizontales, que incluyen funciones de seguridad, seguimiento y administración del sistema.

- **El módulo de complementos de interfaces gráficas de usuario "GUI Plug-in":** Es la parte del sistema que aloja las interfaces de usuario o da soporte a la interconexión de interfaces externas. Estas interfaces de usuario son páginas web diseñadas específicamente para el uso previsto de cada tipo de usuario y escenario. La integración con los sistemas de gestión de enfermedades existentes se articuló creando complementos o plug-ins, que encapsulaban dichos sistemas o permitían la comunicación, dependiendo de cada caso de integración.

Aunque la anterior lista de partes interesadas es muy generalista, proporciona una buena división de las funciones y servicios que construyen la arquitectura del sistema. La Figura 13 relaciona cada categoría de las partes interesadas con cada uno de los módulos conceptuales de la arquitectura propuesta. En el enfoque propuesto, sólo los usuarios finales tienen una relación con los tres módulos (Data Storaga, Model Host, GUI Plug-in), mientras que los investigadores y proveedores de servicios sólo están relacionados con el Model Host y el Data Storage. Además, esta figura muestra que el modelo GUI Plug-in depende de las características del Model Host y el Data Storage; sin embargo, estos dos últimos componentes son independientes (no tienen flechas entre ellos).

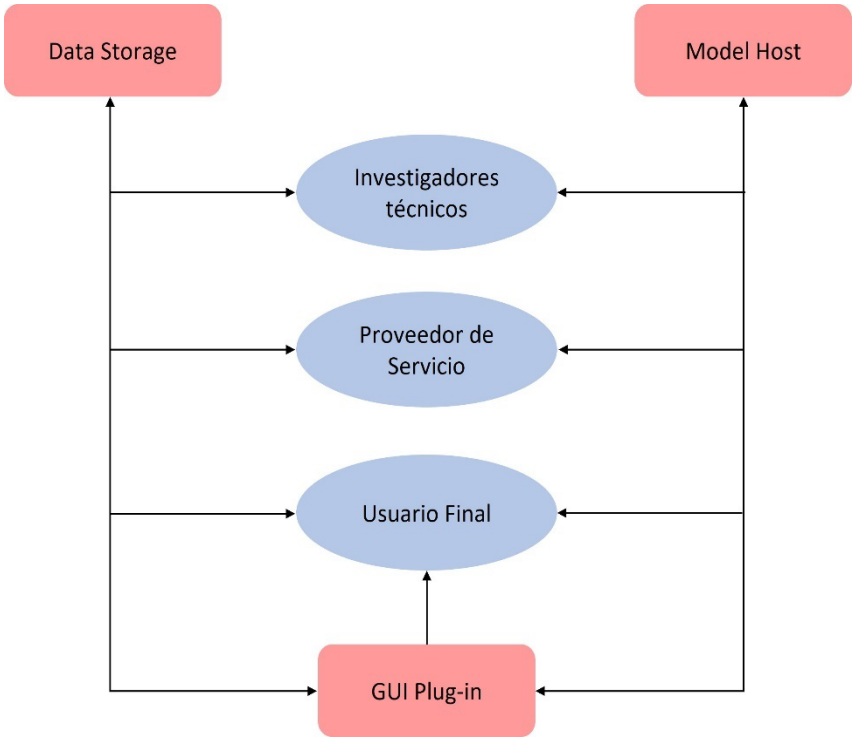

Figura 13: Contexto empresarial que muestra las relaciones entre los servicios y las partes interesadas.

Después de haber identificado a las partes interesadas relevantes, se tuvo que investigar y comprender sus expectativas, es decir, los beneficios esperados que les proporcionaría el sistema, y definir las métricas de calidad de referencia para satisfacer sus expectativas.

4.2.2. Definición del entorno empresarial

El entorno empresarial se define mediante el mapeo del contexto empresarial en componentes implementables reales. UML es un lenguaje de marcado, que permite realizar este mapeo definiendo los aspectos estructurales de los componentes. Los módulos del sistema Data Storage, Model Host y GUI Plug-in de la Figura 13 se asignan a componentes de alto nivel, que implementarán los servicios (definición de bajo nivel). Los usuarios del sistema (investigadores técnicos y usuarios finales) definirán las características de los componentes, y esta definición se utilizará para la descripción de los servicios en la arquitectura del sistema. Es importante resaltar que, si bien existen servicios prestados por los módulos del sistema, en otras aplicaciones (arquitecturas concretas) cada servicio podría ser proporcionado por una entidad comercial separada, implementada y operada de forma independiente, con el único requisito de cumplir con el protocolo de servicio interoperable.

Los principales conceptos de información que se utilizan para calificar los servicios prestados se describen mediante descriptores UML (Figura 14Figura). Estos conceptos están relacionados, principalmente, con los servicios ofrecidos y cómo la arquitectura maneja y procesa esos servicios en general, ayudando a contextualizar su uso.

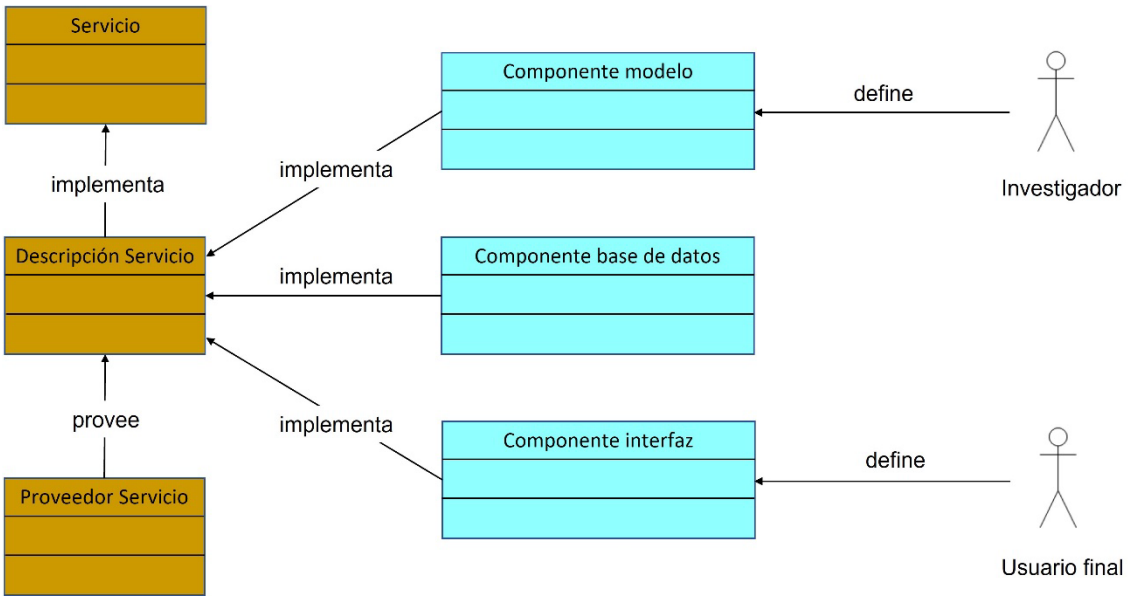

Figura 14: Descriptores de componentes y servicios de UML.

En la arquitectura del sistema existen servicios de diferente nivel. Hay servicios más generales o con funcionalidad muy acotada que sólo están en un módulo, y servicios de más alto nivel o cuya funcionalidad es más compleja, que ofrecen la funcionalidad completa que el usuario percibe. Estos servicios están compuestos o requieren de la interacción con varios servicios más generalistas y estarán presentes en varios módulos. En su descripción se hará referencia a la descripción de los otros servicios.

4.2.3. Patrón de colaboración de servicios

En el siguiente apartado, se verán los detalles de cada uno de los tres módulos del sistema y sus componentes. El objetivo es identificar los servicios de referencia de alto nivel que se brindan en dos niveles diferentes, como se muestra en la Tabla 3.

Tabla 3: Tipos de patrones de colaboración de servicios entre los componentes del sistema.

Tipos de Servicios	Descripción
Módulo a Módulo (B2B)	Servicios que son proporcionados por un módulo del sistema a otro(s) módulo(s) de un tipo diferente (por ejemplo, un proveedor de servicios de los modelos requiere datos de un proveedor de servicios de base de datos remota).
Módulo a Cliente (B2C)	Servicios que proporciona un módulo a las partes interesadas del cliente (por ejemplo, un proveedor de servicios proporciona la ejecución remota de un modelo).

La arquitectura del sistema ha sido diseñada como una Arquitectura Orientada a Servicios [21], en la que los componentes de los diferentes módulos acceden a toda la funcionalidad del sistema, que puede estar ubicado en diferentes localizaciones físicas (uno o varios servidores) a través de la colaboración de los diferentes servicios, aplicando el paradigma de coreografía de procesos.

Los servicios se enumeran según su naturaleza y propósito; por ello, se han agrupado en varios componentes diferentes, que pertenecen a cada uno de los tres módulos y que se detallan el en apartado 4.3.

4.2.4. Infraestructura de Almacén de Datos (Data Warehouse, DW)

Un almacén de datos se compone de una o más bases de datos o subconjuntos de datos, también conocidos como “data marts”, que almacenan estructuras y modelos de datos heterogéneos. Esta heterogeneidad dificulta el desarrollo de funciones de consulta eficientes para almacenes de datos [82] [83]. El uso de descriptores de dominio de conocimiento y referencias semánticas, a través de la definición de una ontología, es clave para formalizar y mapear el tipo de datos alojados en un almacén de datos [84].

Aunque los motores de SQL clásicos siguen siendo difíciles de superar en ciertos contextos (Tabla 4), existen múltiples motores de bases de datos comerciales y no comerciales con características avanzadas, como, por ejemplo, los sistemas MongoDB y NoSQL. Sin embargo, independientemente del rendimiento del motor, la interoperabilidad es un factor clave para diseñar un sistema de almacenamiento de datos adecuado.

Tabla 4: Comparación de soluciones para el almacenamiento de datos.

	Código Abierto	Almacenamiento Estructurado	Almacenamiento No Estructurado	Escalabilidad	Ontologías
MongoDB	SI	NO	SI (Json)	SI	NO
Hadoop	SI	SI	SI	SI	NO
OracleDB	NO	SI	SI (para Oracle NoSQL)	SI	NO
MySQL	SI	SI	NO	Limitada	NO
SQLServer	NO	SI	NO	Limitada	NO
I2B2	SI	SI	NO	Limitada	SI
Cassandra	SI	NO	SI	SI	NO

Entre los motores comparados, I2B2 no admite el almacenamiento no estructurado, pero es capaz de abstraer los conceptos en una ontología.

Con el fin de maximizar la interoperabilidad, se empleó la tecnología del centro I2B2 (Informatics for Integrating Biology and the Bedside), uno de los siete centros financiados por NIH Roadmap for Biomedical Computing. El cual tiene como misión proporcionar a los investigadores clínicos una infraestructura de software capaz de integrar registros clínicos y datos de investigación.

La arquitectura I2B2 se compone de tres capas: una capa de presentación, una capa de servicio y una capa de datos. El usuario accede a I2B2 en la capa de presentación, que expone una interfaz de usuario (User Interface, UI) a través de un cliente web o una aplicación local.

Los datos se almacenan en la capa de datos, que contiene el I2B2 DW. La única forma en que la interfaz de usuario puede acceder a los datos es a través de la capa de servicio. Esta capa es una colección de servicios web, cada uno llamado celda. La colección de estas celdas constituye la llamada colmena I2B2. Las celdas principales de la colmena son: la celda de gestión de proyectos (Project Management, PM), la celda de gráfico para investigación clínica (Clinical Research Chart, CRC) y la celda de gestión de ontologías (Ontology Management, ONT).

La celda PM accede a un conjunto de estructuras de datos en el DW, que asocian usuarios con contraseñas, preferencias y proyectos. Cuando un usuario inicia sesión en el cliente web I2B2, la celda PM administra el proceso de autenticación. Cada vez que otra parte de la colmena intenta realizar una acción en nombre del usuario, se dirige a la celda de gestión de proyectos para recopilar las autorizaciones adecuadas. Una vez autenticado, el usuario (a través del cliente web) realiza consultas a través de la celda CRC, también conocida como celda del repositorio de datos. Para facilitar el proceso de consulta para el usuario, los datos se asignan a conceptos organizados en una estructura similar a una ontología, que es administrada y accedida por la celda ONT.

El modelo de datos I2B2 se basa en un esquema en estrella. El esquema en estrella tiene una tabla central de hechos donde cada fila representa un solo hecho. En I2B2, un hecho

es una observación sobre un paciente. Las observaciones sobre un paciente son registradas por un observador específico en un rango de tiempo específico (definido por fechas de inicio y finalización) y están relacionadas con un concepto específico, como una prueba de laboratorio o diagnóstico, en el contexto de un encuentro o visita. El concepto puede ser cualquier atributo codificado sobre el paciente, como un código para una enfermedad, un medicamento o un resultado de prueba específico. Esta forma de representar conceptos se basa en trabajos previos conocidos como el modelo Entidad-Atributo-Valor (Entity-Attribute-Value, EAV) [30]. La razón por la que los desarrolladores de I2B2 decidieron implementar este modelo es que la consulta de datos modelados con un esquema en estrella representado en un formato EAV es eficiente [85].

4.2.5. Escalas de riesgo de DM2

Los modelos de riesgo de DM2 del estado del arte se basan en modelos matemáticos ejecutados sobre variables numéricas y/o categóricas (Tabla 5). Según el resultado, dichos modelos pueden proporcionar la probabilidad *p* de desarrollar o tener DM2 (ecuaciones (4.1) y (4.3)) o la tasa de riesgo de desarrollar DM2 a lo largo del tiempo (ecuación (4.2)). El rendimiento de un modelo de discriminación se evalúa mediante el estadístico C (también conocido como área bajo la curva ROC) [71].

$$p = \frac{1}{1 + \exp\left(-(\propto + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_m X_m)\right)} \quad (4.1)$$

$$h(t) = h_0(t)\exp\left(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_m X_m\right) \quad (4.2)$$

$$p = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_m X_m \quad (4.3)$$

donde:

• α es la probabilidad a priori.

• $h_0(t)$ es la tasa de riesgo basal.

• β_x es el coeficiente de regresión, que denota el peso relativo del predictor correspondiente.

• X_x son los predictores o variables, que pueden ser numéricas (continuas) o categóricas (0, 1, 2...).

Tabla 5: Evaluación del rendimiento de discriminación de las escalas de riesgo del estado del arte.

Nombre de la escala de riesgo y estudio de validación	**Modelo matemático**	**Rendimiento *(estadístico C)***	**Predictores**
Findrisc [32,33]	Regresión logística ponderada	85%	Edad, medicación AHT, FG, IMC, cintura
ARIC [20,34]	Regresión logística	80%	Edad, origen étnico, FG, HDL, triglicéridos, presión arterial, FHD, cintura, altura
San Antonio [35,36]	Regresión lineal	84%	Edad, sexo, origen étnico, FG, IMC, HDL, presión arterial, FHD
Cambridge [21,37]	Regresión logística	75%	Edad, Género, AHT, Esteroides, IMC, FHD, Tabaquismo
PREDIMED [38]	Modelo multivariante de supervivencia de Cox	78%	AHT, FG, presión arterial, FHD, fumador, consumo de alcohol
Framingham [34,39]	Regresión logística	84%	Edad, sexo, AHT, FG, IMC, HDL, triglicéridos, presión arterial, FHD, cintura
MOSAIC [40]	Red bayesiana	79%	Edad, sexo, FG, fumador, alcohol, AHT, LL, actividad física, triglicéridos, HDL, IMC, cintura, accidente cerebrovascular, FHD

FG: Glucosa en ayunas; AHT: Medicación Antihipertensiva; HDL: Lipoproteína de alta densidad; FHD: Antecedentes familiares de diabetes; IMC: Índice de Masa Corporal; LL: Medicación para la reducción de grasas (hipolepimiante).

Un modelo interesante para la detección de DM2, que no se basa en las regresiones antes mencionadas, es el modelo MOSAIC [86], el cual es de código abierto y está disponible para investigación (https://github.com/sambofra/bnstruct (último acceso el 24 de septiembre de 2022)). Este modelo se basa en una red bayesiana y permite imputar parámetros desconocidos. El modelo MOSAIC fue construido para ser aplicable en diferentes contextos, y los rendimientos son comparables a la puntuación de Findrisc en escenarios donde los datos clínicos no están disponibles. Este modelo muestra un valor predictivo aceptable, cuando se dispone de información clínica para colesterol y glucosa en ayunas [87], por lo que fue seleccionado como metodología de imputación de datos faltantes.

4.2.6. Diseño del Estudio Piloto

El piloto se basó en un estudio aleatorio en un solo centro, que investigó el rendimiento del sistema y la escalabilidad de las herramientas al ser utilizadas por los médicos.

El sistema se evaluó en el Servicio de Endocrinología del Hospital La Fe durante un periodo de 12 semanas, involucrando a los endocrinólogos y al jefe de servicio. Se evaluó la predicción y rendimiento de detección de las escalas de riesgo de DM2 en una población

real, basándose en registros de salud electrónicos (EHR) retrospectivos. El comité de ética de investigación biomédica del Hospital La Fe aprobó en enero de 2015 la solicitud formal de datos y el diseño del estudio.

El plan del estudio constaba de tres etapas:

1. **Sesiones de formación:** Las tres primeras semanas se realizaron tres sesiones grupales para introducir a los participantes en las herramientas y aprender las acciones para visualizar datos y ejecutar los modelos de riesgo. Los participantes que firmaron el consentimiento informado fueron aleatorizados de forma ciega y asignados al programa de la sesión de evaluación.

2. **Evaluación de escalas de riesgo y evaluación clínica:** La etapa de evaluación de las herramientas duró 8 semanas, durante las cuales los endocrinos y el jefe de servicio realizaron sesiones de 2 h con las herramientas.

3. **Análisis de datos:** Durante la última semana se realizó la adquisición de registros, trazas e Indicadores Clave de Rendimiento (KPI) para la evaluación técnica del sistema.

Para cada modelo se definió un escenario de mejor y peor caso, de acuerdo con las especificaciones y comportamiento de las operaciones. Para el modelo de predicción, el mejor caso fue la ejecución para un solo paciente, y el peor fue la ejecución para la población más alta disponible, que era 8080. En el caso del modelo de detección, como éste se ejecuta para un único paciente, el peor caso era cuando el modelo no tenía ninguna variable de entrada (es decir, tenía que estimar los 20 parámetros faltantes) y, el mejor caso, cuando tenía todos los parámetros de entrada y no tenía que estimar.

El rendimiento técnico de la herramienta se evaluó para los siguientes KPI en el mejor y peor escenario:

• Carga computacional (huella de memoria en el servidor).

• Retraso en la respuesta a la(s) solicitud(es) de servicio.

• Tiempo de acceso a la base de datos/caché principal (ms).

• Intervalo(s) de tiempo de uso.

• Máximo retraso de respuesta.

Para confirmar la escalabilidad y confiabilidad de la arquitectura propuesta, es necesario hacer un seguimiento de las características técnicas. Estas dos dimensiones de calidad se han definido previamente, como el nivel de disponibilidad y los indicadores de superación del umbral de CPU [88]. Para probar esto, se usan los umbrales propuestos por Kianpisheh, Kargahi y Charkari [89], los cuales son CPU <83% y el nivel de disponibilidad diferente de “inalcanzable”.

4.3. Descripción de la arquitectura del sistema

La arquitectura del sistema ha sido realizada siguiendo el paradigma de arquitectura orientada a servicios, en concreto, se ha utilizado la coreografía de procesos. En este tipo de paradigma los servicios se proporcionan y se comparten entre los componentes dentro de los tres módulos conceptuales descritos en el contexto empresarial. La comunicación se realiza utilizando un protocolo de comunicación descrito en este apartado, que es controlado por un componente central de coreografía. Este apartado describe en detalle la arquitectura diseñada. La primera parte de la descripción se centra en el tipo de servicios de la arquitectura. La segunda parte describe los módulos y componentes. La tercera y última parte describe el motor de coreografía y el protocolo de comunicación (XMGS).

4.3.1. Vista funcional

Según IEEE 42010 [90], la vista funcional describe las capacidades, la estructura, las responsabilidades y las especificaciones de los componentes del sistema y cómo interactúan entre sí. La vista funcional clasifica los servicios en tres tipos: aplicación, interoperabilidad y servicios de sistema.

La Figura 15 muestra la arquitectura del sistema y las relaciones funcionales entre los módulos. Los tres módulos están conectados por el motor de coreografía CHOREMED descrito en el capítulo 3. Los servicios de la arquitectura propuesta se agrupan en tres categorías: servicios de interoperabilidad de datos, servicios de los modelos y servicios de interfaz de usuario.

Los servicios de interoperabilidad de datos pueden ser reutilizados por cualquier componente dentro del sistema, y se dedican a extraer y almacenar datos del módulo de almacenamiento (por ejemplo, pueden realizar procesos de extracción, transformación y carga de los datos de entrada para los algoritmos o realizan consultas para mostrar datos sin procesar en la interfaz). Los servicios de los modelos son servicios dedicados a la ejecución de algoritmos de predicción y detección. Los servicios de Interfaz de Usuario cubren las operaciones lógicas (incluidas la lógica funcional y la infraestructura) que son comunes a varios escenarios (por ejemplo, mostrar datos en la interfaz web o trazar gráficos).

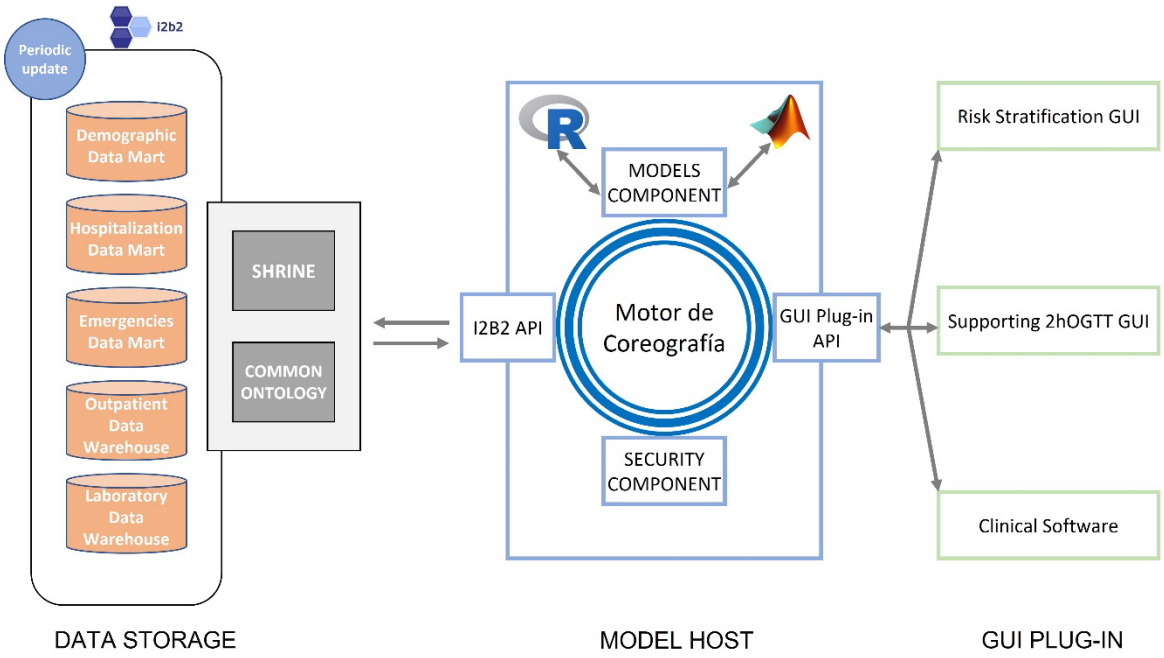

Figura 15: Vista funcional de la arquitectura del sistema.

De izquierda a derecha en la Figura 15, el esquema muestra el módulo Data Storage de almacenamiento de datos y basado en la tecnología I2B2, el módulo Model Host, que permite, entre otras funciones, ejecutar los modelos híbridos, está construido empleando el motor de coreografía y herramientas proporcionadas por CHOREMED, y, por último, el módulo GUI Plug-in que aloja las aplicaciones web, a través de las cuales interactúan los usuarios finales o permite la interconexión con software de terceros.

El módulo Data Storage está compuesto por varias entidades únicas de datos de diferentes fuentes (data marts): hospitalización, pruebas de laboratorio, servicios ambulatorios, etc. Desde un punto de vista lógico, el módulo Data Storage es una parte conceptual única, que se estructura de acuerdo con una ontología común presentada previamente [91]. Esta ontología representa cada evento clínico que le sucede al paciente en cada data mart del almacén de datos, proporcionándole una hora de inicio y finalización y conectándolo con los conceptos específicos relacionados con un evento en particular. Una vez que se prepara una consulta, la ontología común traduce estos conceptos y la red de intercambio de datos (denominado SHRINE) agrega la consulta para que se ejecute en cada uno de los data marts. Desde un punto de vista físico, cada data mart es una máquina aislada, ubicada en otro lugar y accesible a través de Internet.

La conexión del Data Storage y el Model Host se realiza mediante servicios conectores realizados en CHOREMED que permiten interactuar con la capa de servicios web de SHRINE. SHRINE está compuesto por un conjunto de servicios de interoperabilidad, que

permiten realizar consultas federadas a todos los data marts, independientemente de su ubicación física y estructura de datos [92]. Esta configuración permite a los investigadores y médicos elegir la población objetivo para ejecutar los modelos independientemente de la fuente de datos.

El módulo Model Host constituye el núcleo de la arquitectura propuesta, éste ha sido construido sobre el motor de coreografía CHOREMED. Aunque toma el nombre de una de las funcionalidades que proporciona, también realiza otras de tipo más transversal necesarias para el funcionamiento del sistema. Las funcionalidades proporcionadas podrían agruparse en las siguientes categorías:

- **Hospedaje de los modelos:** Ésta es la funcionalidad que da nombre al módulo. El motor de coreografía permite crear y ejecutar los diferentes modelos del sistema (el cual se describe en detalle en el punto 4.3.2).
- **Integración de módulos:** Mediante la creación de diferente servicios preexistentes o creados exprofeso para esta arquitectura, se permite que los tres módulos del sistema se comuniquen entre sí para ejecutar las coreografías que forman el proceso asociado a una funcionalidad. Se han desarrollado servicios conectores para integrar el servicio web de SHRINE asociado al módulo Data Storage, se han utilizado conectores de tipo TCP preexistentes en CHOREMED para integrar las interfaces de usuario y se han desarrollado conectores de tipo servicio web para aquellos sistemas de terceros que sólo admitían este tipo de conectividad.
- **Servicios de sistema:** En esta categoría se agruparían aquellos servicios, ya existentes en CHOREMED o creados expresamente, que dan soporte a funcionalidades del sistema. Entre ellas, las más destacadas son los servicios relacionados con la trazabilidad y la seguridad, que se explicarán en los puntos 4.3.4 y 4.3.5.

Como se mencionó anteriormente, los requisitos para proporcionar los parámetros de entrada y ejecutar los algoritmos específicos involucran muchos elementos software dentro del sistema, que deben poder funcionar de manera distribuida y controlada. Este tipo de ejecución de procesos complejos se resuelve utilizando la coreografía de procesos, en la que los servicios pueden intercambiar datos para ejecutar procesos de forma distribuida, sin necesidad de un elemento que los coordine [93]. Para ello, se han definido diferentes coreografías de servicios, que permiten ofrecer la funcionalidad o flujo de trabajo correcto a cada interfaz de usuario.

El uso de la coreografía de procesos permite una mayor flexibilidad a la hora de crear y modificar los procesos o flujos de trabajo definidos. Es posible conectar o desconectar componentes/servicios dinámicamente, modificando así el flujo de trabajo. Los servicios no requieren conocer dónde está ubicado físicamente otro servicio para hacer uso de éste, por lo que permite crear sistemas distribuidos y ayudar al escalado del sistema.

4.3.2. Componente Model Host

Éste es el componente principal del sistema, ya que es el encargado de poner a disposición del resto de componentes los diferentes modelos que forman el sistema híbrido.

Tradicionalmente, cuando un modelo se proporciona a un usuario final para su validación, éste se proporciona en forma de programa compilado para obtener el máximo rendimiento de la herramienta en cuanto consumo de recursos y tiempo de cómputo. En el diseño de este sistema, uno de los requisitos era proporcionar a los investigadores un mecanismo ágil y robusto para desplegar nuevos algoritmos, probarlos y aplicar modificaciones, tratando de minimizar el impacto sobre el sistema y el tiempo necesario para estas acciones.

Teniendo en cuenta los requisitos mencionados, se optó por proporcionar una arquitectura que permitiese a los investigadores desplegar sus scripts originales, en los que codifican sus modelos, y ejecutarlos sobre el motor específico que entiende dicho script. Dichos motores se encuentran instalados en el servidor de ejecución. De esta forma, un investigador puede mejorar un modelo simplemente sustituyendo el script de ejecución. Script que será muy similar al utilizado en sus pruebas de laboratorio, simplemente adaptándolos ligeramente para la aceptación de parámetros y generación de resultados en el formato de intercambio definido para la plataforma. Esto permite una reducción en los tiempos de despliegue y una reducción de los errores derivados de los procesos de recodificación.

Para lograr esta arquitectura, se ha dividido el módulo en diferentes servicios, siguiendo el paradigma de coreografía de procesos y empleando el coreógrafo CHOREMED (Figura 16). Así, se han definido varios tipos de servicios:

- **Conectores a motores de ejecución:** Se han creado servicios conectores que dado un script son capaces de ejecutar dicho script sobre el motor de ejecución correspondiente instalado en el servidor, para posteriormente recuperar el resultado. En concreto se ha creado un conector a R y otro a Matlab.
- **Modelos de ejecución:** Existe un servicio de este tipo por cada modelo existente en la plataforma. Este tipo de servicio, ante una solicitud de ejecución y partiendo de un script base, deberá interactuar con otros servicios para obtener los datos de entrada al script, completar dicho script y solicitar su ejecución al conector del motor adecuado. Cuando la ejecución termine deberá recibir la salida de la ejecución y desempaquetar el resultado para devolverlo a quien lo haya solicitado.

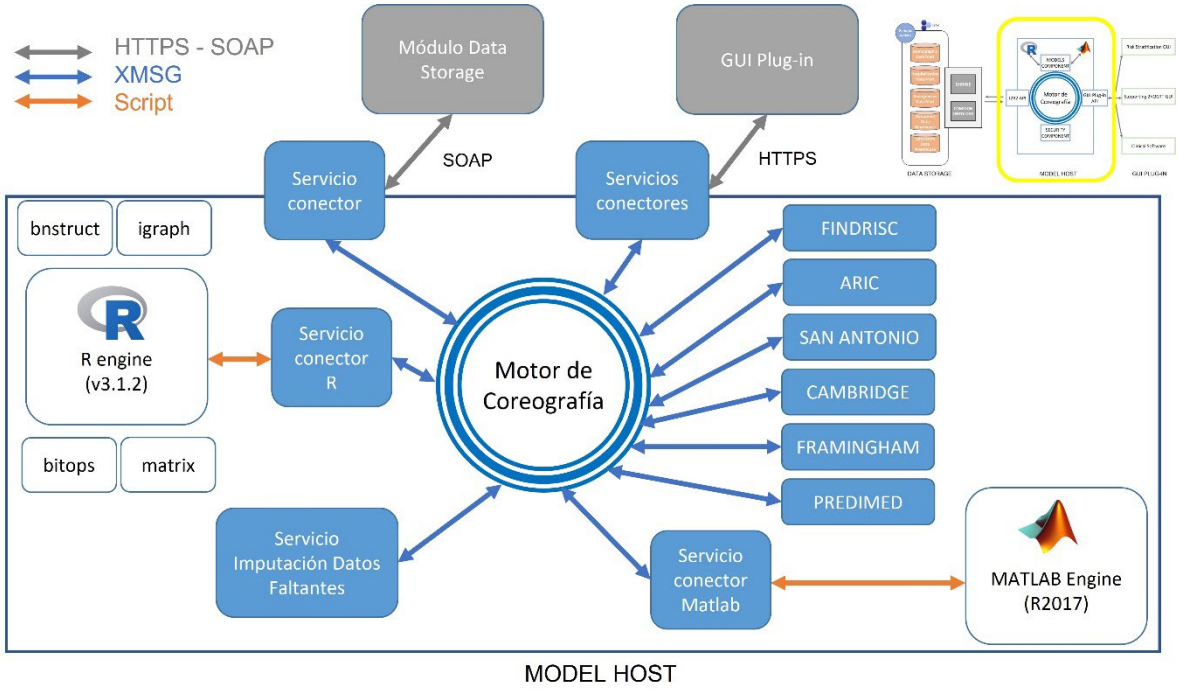

Figura 16: Ejecución de las ecuaciones de puntuación de riesgo utilizando motores matemáticos.

4.3.3. Componente de seguridad

El Componente de Seguridad está a cargo de proporcionar características horizontales seguras para todos los servicios. Los elementos de seguridad se basan en cuatro dimensiones:

- **Autenticación:** debe ser posible que el proveedor del servicio determine la identidad del solicitante del servicio.
- **Autorización:** el proveedor del servicio debe poder determinar si el solicitante tiene los derechos apropiados para invocar el servicio.
- **Confidencialidad del mensaje:** el contenido del mensaje sólo debe ser visible para el destinatario previsto.
- **Integridad del mensaje:** debe ser posible garantizar que un mensaje no ha sido alterado o manipulado durante el transporte entre el consumidor del servicio y el proveedor del servicio.

Para garantizar estas dimensiones, se tomaron las siguientes medidas:

- Se creó un servicio en el Model Host, para que cualquier servicio del motor de coreografía pudiese solicitar autenticar a un usuario mediante sus credenciales

(usuario y contraseña) y conocer sus permisos de acceso o autorización. Por ejemplo, para acceder a las interfaces gráficas de usuario o cuando éstas solicitan ejecutar un servicio, dichas credenciales viajan con la solicitud para que el servicio las valide.

- La comunicación de las interfaces web o servicios web del sistema se realizó empleando certificados x.509 y el protocolo de seguridad de la capa de transporte SSL (Secure Socket Layer).
- Por último, para las comunicaciones que emplean los servicios conectores TCP de CHOREMED, éste proporciona el cifrado de los mensajes mediante el algoritmo Blowfish.

4.3.4. Componente de trazabilidad

Este componente se crea con el objetivo de disponer de un mecanismo que permita evaluar el funcionamiento del sistema, determinar si está funcionando adecuadamente y medir indicadores, como tiempos de respuesta, elementos más utilizados o comportamiento del usuario en las interfaces gráficas.

Este componente se utiliza para registrar una traza de las actividades que tienen lugar durante el funcionamiento del sistema. Por lo tanto, debe permitir registrar, tanto las comunicaciones entre los servicios que forman cada componente, como la interacción de los usuarios en las interfaces gráficas. Para lograr esto, el módulo hace uso de dos elementos a diferente nivel:

- **Servicios:** Por una parte, para trazar todas las comunicaciones entre los servicios que forman los componentes y a su vez los módulos, se utiliza el servicio de trazabilidad que proporciona CHOREMED. Este servicio permite almacenar en disco los mensajes de coreografía enviados (o filtrar unos específicos empleando las direcciones lógicas) y sus marcas de tiempo. Además, también se dispone de un segundo servicio con interfaz gráfica, que muestra los mensajes en tiempo real para ayudar en la depuración de errores (Figura 17).
- **Interfaz de usuario:** Por otra parte, se hace uso en las interfaces gráficas de librerías de trazabilidad como Log4J, Log4Net y Google Log para guardar las interacciones de los usuarios.

Se utiliza un formato común de fichero para poder integrar los datos de las diferentes fuentes de datos, dicho formato define una interacción en cada línea con los siguientes campos: <Marca de tiempo>, <Módulo>, <control>, <Texto libre>

- Marca de tiempo: dd/mm/aaaa hh:mm:ss.
- Módulo: Para los mensajes de coreografía es el servicio remitente u origen del mensaje. Para la parte gráfica es el módulo (vista o formulario) en el que se encuentra actualmente el usuario.
- Control: Para los mensajes de coreografía es el servicio destino del mensaje. En la interfaz de usuario es el control utilizado (botón, etiqueta, imagen, gráfico, tabla, etc.).
- Texto libre: En el caso de mensajes de coreografía es el mensaje serializado en JSON. En la interfaz gráfica es el texto que indica la interacción o notas para el experto en usabilidad.

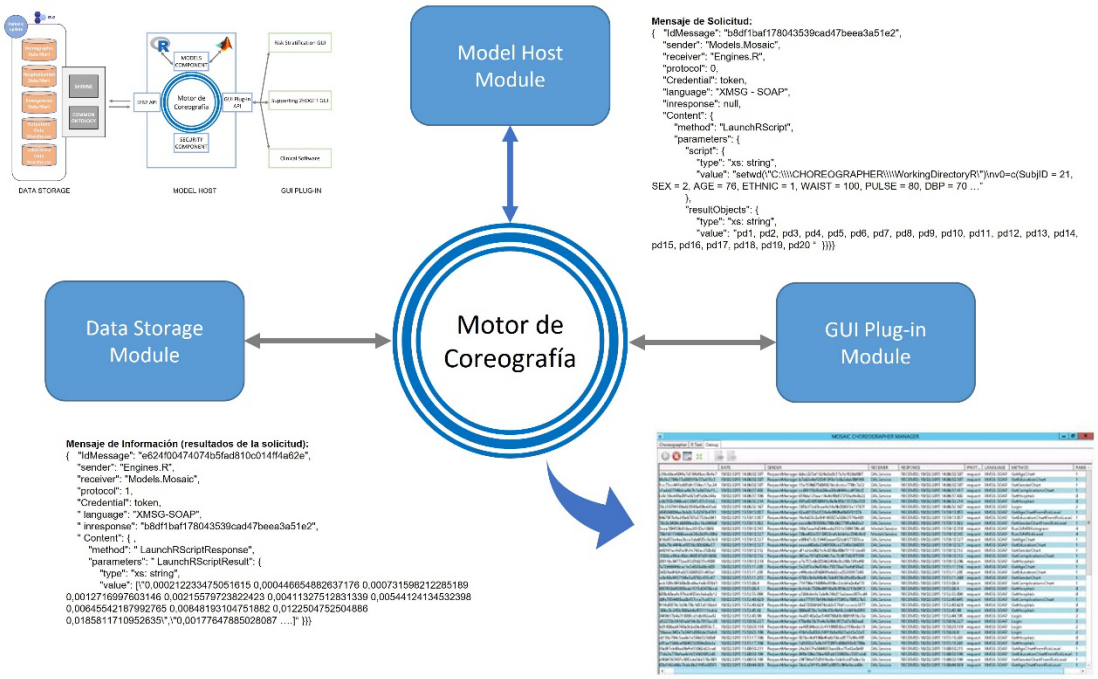

Figura 17: Seguimiento de los mensajes de servicio del sistema.

4.3.5. Mecanismo de comunicación

La base de la arquitectura SOA del paradigma de coreografía de procesos es la interacción entre servicios. Éstos deben poder comunicarse en un formato común, que permita los diferentes tipos de interacciones entre servicios. Para lograr esta comunicación se hace uso del coreógrafo CHOREMED. Como se explica en el punto 3.2.2 de esta memoria, dicho coreógrafo dispone de un motor de ejecución para poner en marcha los servicios y realizar el intercambio de mensajes entre ellos, además de proporcionar elementos para ayudar con el desarrollo de servicios y de disponer de servicios propios para tareas transversales.

En la Figura 18 se puede ver cómo el motor de intercambio de mensajes se sitúa en el centro de la arquitectura, se encarga de poner en marcha los diferentes servicios y de intercambiar mensajes de coreografía que se envían entre ellos. En la figura se puede ver que hay servicios desarrollados empleando las herramientas de CHOREMED y que se conectan localmente al motor. Por ejemplo, los servicios de ejecución de modelos, servicios de login o trazabilidad. Pero, también, se ven servicios externos que emplean servicios conectores TCP o de servicio web para integrarse en la arquitectura, como el servicio SHRINE para acceso a datos o las herramientas gráficas.

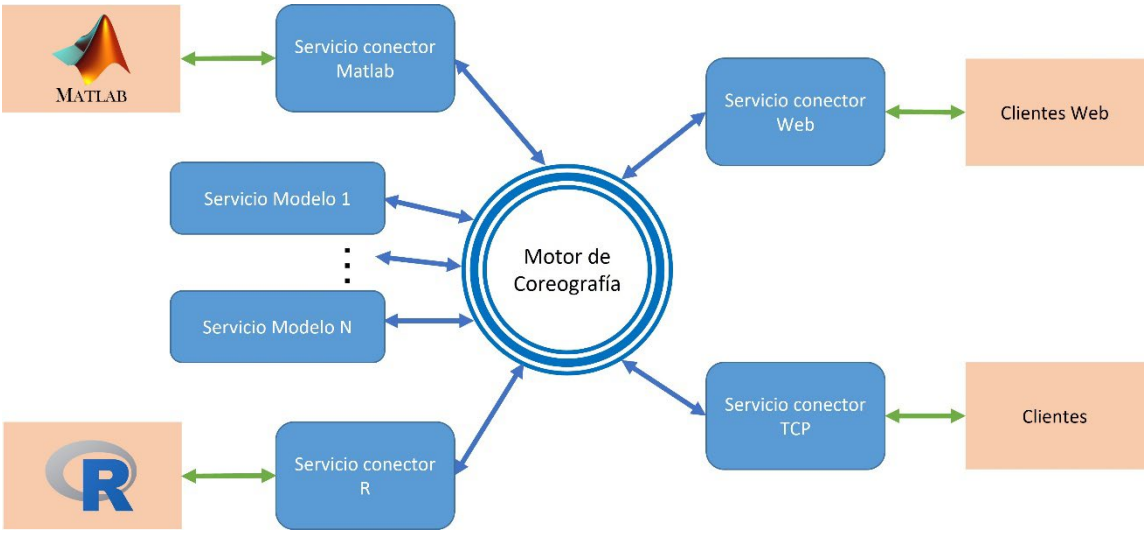

Figura 18: Esquema funcional del sistema de coreografía.

Para el intercambio de mensajes se hace uso del formato específico de CHOREMED denominado XMSG, este protocolo se explicó en el punto 1.6.1 de esta memoria. Dicho formato permite que los servicios se comuniquen de forma asíncrona entre ellos para solicitar funcionalidad. Tal como ya se explicó, los servicios pueden realizar comunicaciones uno a uno o de difusión/multidifusión. Además, el servicio desconoce la ubicación física de los otros servicios, permitiendo así crear una arquitectura con servicios locales o de terceros, como la presentada. Además, permite desarrollar arquitecturas con varios coreógrafos conectados entre sí, por ejemplo, en el caso presentado se podrían separar los servicios de los modelos en diferentes coreógrafos situados en servidores diferentes, esto permitiría repartir la carga de cómputo y contribuir a aumentar la escalabilidad del sistema.

Gracias a que el formato XMSG está creado para ser extendido, para incluir el mecanismo de seguridad se mejoró la definición del formato con un nuevo campo:

- **Credenciales:** Es un campo opcional orientado a implementar la autenticación entre los servicios que componen la coreografía. Este campo permite almacenar información de las credenciales del servicio que envía el mensaje. Si bien se incluye el campo para las credenciales, no se restringe el formato para permitir una mayor flexibilidad y crear servicios de autenticación a medida para cada caso.

A continuación, en la Tabla 6, se muestra un ejemplo serializado a JSON del intercambio asíncrono de mensajes entre el servicio del modelo de imputación de datos y el servicio que ejecuta el script de dicho modelo en el motor de R. Primero, se envía un mensaje de tipo *Solicitud* que solicita ejecutar el método *LauchRScript* que contiene el script a ejecutar. De forma asíncrona cuando la ejecución en R termina se responde con un mensaje de tipo *Información* con los resultados.

Tabla 6: Ejemplo de XMSG entre dos servicios.

Mensaje de Solicitud:


```
{
   "IdMessage": "b8df1baf178043539cad47beea3a51e2",
   "sender": "Models.Mosaic",
   "receiver": "Engines.R",
   "protocol": 0,
   "Credential": token,
   "language": "XMSG - SOAP",
   "inresponse": null,
   "Content": {
      "method": "LaunchRScript",
      "parameters": {
         "script": {
            "type": "xs: string",
            "value": "setwd(\"C:\\\\CHOREOGRAPHER\\\\WorkingDirectoryR\")\nv0=c(SubjID = 21, SEX = 2,
AGE = 76, ETHNIC = 1, WAIST = 100, PULSE = 80, DBP = 70 …"
         },
         "resultObjects": {
            "type": "xs: string",
            "value": "pd1, pd2, pd3, pd4, pd5, pd6, pd7, pd8, pd9, pd10, pd11, pd12, pd13, pd14, pd15, pd16,
pd17, pd18, pd19, pd20 "
         }
      }
   }
}
```


Mensaje de Información con los resultados de respuesta:


```
{
   "IdMessage": "e624f00474074b5fad810c014ff4a62e",
   "sender": "Engines.R",
   "receiver": "Models.Mosaic",
   "protocol": 1,
   "Credential": token,
   " language": "XMSG-SOAP",
```

```
" inresponse": "b8df1baf178043539cad47beea3a51e2",
" Content": {
  "method": " LaunchRScriptResponse",
  "parameters": " LaunchRScriptResult": {
    "type": "xs: string",
    "value": [\"0,000212233475051615 0,000446654882637176 0,000731598212285189
0,0012716997603146 0,00215579723822423 0,00411327512831339 0,00544124134532398
0,00645542187992765 0,00848193104751882 0,0122504752504886
0,0185811710952635\",\"0,00177647885028087 ….]"
  }
 }
}
```


4.4. Resultados Experimentales

En este apartado se presentan los resultados experimentales de la arquitectura propuesta.

4.4.1. Escenarios para la evaluación de escalas de riesgo de DM2

El impacto esperado del sistema es mejorar la caracterización de la aparición de DM2 y de la población con riesgo de desarrollar DM2 en el futuro, o, que ya tiene DM2, pero sin diagnosticar. Dadas como entrada las variables disponibles en un registro de salud electrónico para un paciente o una población dada, los modelos pueden estimar la probabilidad de estar en alto riesgo y, para los modelos de detección, indicar el valor más probable de entre los posibles diagnósticos [87].

Se definen dos escenarios clínicos diferentes (casos de uso) en el cribado y la estratificación del riesgo:

1. Estimar las variables faltantes usando las variables disponibles medidas durante las visitas del médico general y de las pruebas de laboratorio de la historia clínica electrónica, para realizar la estratificación del riesgo.

2. Calcular el rango de glucosa de la prueba de tolerancia oral a la glucosa de 2 horas (2h-OGTT) usando todas las demás variables disponibles (ayudando al especialista en diabetes a decidir si esta prueba es necesaria).

4.4.1.1. Escenario 1: Estratificación del riesgo

En este caso, los datos de entrada provienen del sistema de información de salud (EHR) de una institución o agencia de salud. Los datos de entrada son variables demográficas, y

cuándo están disponibles variables medidas en las visitas del médico general y análisis de sangre. El resultado es una vista (empleando gráficos, por ejemplo, gráficos circulares) de la distribución de la población con mayor riesgo de tener DM2 y ser prediabético.

Caso 1, agencia de salud con disponibilidad limitada de EHR: Se supone que la información disponible para la agencia de salud se limita a variables demográficas (género, edad, etc.), porque el sistema de información de salud aún no está integrado en estos entornos. Antes de realizar una solicitud al hospital o a la institución de atención primaria para que les proporcione información fenotípica y metabólica de su población atendida, este sistema podría usarse para estratificar mejor esta solicitud y acotarla sólo a la población que realmente tiene la mayor probabilidad de estar en riesgo.

Caso 2, organismo sanitario con plena disponibilidad de EHR: En este caso, los datos de entrada al sistema serán todas las variables habitualmente disponibles en la historia clínica de un ciudadano normal. El resultado se utilizará para determinar los subgrupos en riesgo de tener DM2 o ser prediabéticos. Otro resultado podría ser la determinación de meta variables, como ser fumador, tener el colesterol alto o no tener un estilo de vida óptimo. La herramienta podría ayudar a tomar decisiones sobre políticas de salud pública relacionadas con las campañas de cribado, ayudando a estimar mejor su impacto, por ejemplo, cuántas pruebas de 2h-OGTT se necesitan, análisis de glucosa en sangre en ayunas, visitas de detección, etc.

Caso 3, compañía de seguros de salud: En este caso, la herramienta del sistema se puede utilizar para ayudar a la empresa a evaluar el riesgo de los gastos de atención médica de un grupo objetivo (una empresa atendida o un grupo de personas) y desarrollar mejor las actividades rutinarias, como las previsiones financieras, actividades de detección y campañas de promoción de la salud mejor adaptadas y personalizadas a sus clientes.

4.4.1.2. Escenario 2: Apoyo a la decisión de 2h-OGTT

En este caso, la herramienta tendría como entrada el EHR de un paciente, y la salida principal es tener una estimación del rango de glucosa de 2h-OGTT, dadas todas las demás variables disponibles. Gracias a esto, la herramienta puede apoyar la decisión de recomendar o no OGTT, con beneficios evidentes en términos de resultados de salud y ahorro de costes.

4.4.2. Evaluación técnica

El personal clínico del Hospital La Fe (Tabla 7) utilizó el sistema para identificar subgrupos de riesgo y analizar sujetos de riesgo alto-bajo durante ocho semanas consecutivas (Tabla 8).

Tabla 7: Médicos incluidos en el estudio piloto para evaluar los dos escenarios.

Sexo		Hombre (2)/Mujer (6)
Edad (Años)		42 ± 13
Experiencia Profesional		14 ± 10
Alfabetización en TIC (Auto-reportado)		Alto = 3; Medio = 3; Bajo = 2;
Número de pacientes atendidos	Total	319,33 ± 247,66
	Con DM2	127,44 ± 75,22
	Con Alto riesgo de desarrollar DM2	48,00 ± 33,79

Tabla 8: Distribución de las sesiones de evaluación (número, duración, número de pacientes por día y por sesión).

Indicador de Uso	**Media**	**Desviación estándar**	**Min**	**Max**
Número de usuarios por día	2,5	16,43	1	4
Duración de las sesiones (min)	26,16	13,72	0,25	45,93
Número de pacientes evaluados por médico	6,25	4,97	1	15
Número de pacientes evaluados por día	10,71	12,18	0	26
Número de sesiones por médico (usuario)	1,82	1,16	1	5

La evaluación técnica de los componentes en ejecución se ha realizado con la versión del sistema desplegada para el piloto. El módulo Model Host se ejecutó en un Windows Server 2012 R2 Standard, con un procesador Intel Xeon E5405 de 2 GHz con una memoria RAM de 24GB. El rendimiento y la utilización de recursos se han supervisado mediante el módulo de trazabilidad del coreógrafo, y las herramientas de análisis de rendimiento predeterminadas de Windows/Ubuntu.

4.4.3. Mapa de Evaluaciones

El sistema presentado está diseñado empleando el paradigma de coreografía de procesos y se compone por tres módulos principales: (1) Data Storage; (2) Model Host; y (3) GUI Plug-in. Varios componentes desplegados en diferentes tecnologías conforman cada uno de estos módulos, y la colaboración y comunicación fluida entre ellos fueron aspectos críticos para garantizar la correcta ejecución de los flujos de trabajo definidos. Las evaluaciones se han realizado como los escenarios clínicos del estudio (Apartado 4.4.1), pero, más concretamente, los componentes afectados son:

• Almacenes de datos (Data Warehouses, DW)

• Capa de acceso a datos (Motor de consulta (Query Engine, QE)): múltiple/único sujeto

• Imputación de datos faltantes (Missing Data Imputation, MDI)

• Módulo de escalas de riesgo (Risck Score Module, RSM)

• Coreógrafo

• Módulo de interfaz

La ejecución de los componentes mencionados no siguió un esquema, ya que algunos de ellos operan en segundo plano y actualizan nueva información o resultados del modelo a medida que están listos para ser enviados a componentes relacionados (por ejemplo, el QE verifica si los datos de resultados ya están disponibles de solicitudes anteriores y muestra los resultados almacenados en caché, sin invocar MDI/RSM nuevamente).

La actuación técnica se ha realizado sobre los componentes mencionados y buscando los siguientes indicadores:

- Verificación de la ejecución del modelo:
  - Idoneidad de la consulta
  - Homogeneización de unidades
  - Manejo de los resultados y almacenamiento
- Rendimiento de la ejecución del modelo:
  - Mejor caso versus peor caso
  - Latencia (tiempo de retardo de la respuesta)
  - Carga de memoria del servidor del sistema (Uso de RAM)
  - Carga de la unidad de proceso central (Uso de CPU)
  - Recursos de red (Ancho de banda)

4.4.4. Verificación de la Ejecución de los Modelos

El sistema evaluado ha integrado el estado del arte de modelos estadísticos (escalas de riesgo e imputación de datos) ejecutando directamente los scripts de dichos algoritmos, no en forma de programas compilados ejecutables. De esta manera, el sistema puede realizar actualizaciones en caliente (sin parar y reiniciar) y modificaciones fácilmente (recalibración). La Figura 19 muestra un ejemplo de integración del modelo de imputación de datos faltantes: en el lado izquierdo, el código original; en el lado derecho, el script de integración, que implementa una llamada al motor R y el archivo de script en formato nativo.

R Console for testing the model	**Model execution in Model Host Module**

> library(bnstruct)
input <-list(
SEX = 2, AGE = 69, C_SMK = 2,
H_SMK = 2, H_STR = 2, H_HBG = 2, BMI = 28.50,
WAIST = 92, SBP = 154, DBP = 96,
AHT_M = 1, LLO_M = 1, CHOL =237,
TRIG = 9.8904, HDL = 32, GLO = 4.88, MS = 2 .....
)
>source("BNscript.R")
>v1;v2;v3;v4;v5;v6;v7;v8;v9;v10;v11;v12;v13;v14;
v15;v16; v17;v18;v19;v20;v21
 	
[1] 0.58741910 0.25497897 0.02094656
0.11775952 0.01889585
[1] 0 1
[1] 0 1
[1] 0.68429253 0.28048591 0.03522156
[1] 0 1
[1] 0 1
[1] 0 1 0
[1] 0 0 1
[1] 0 1
[1] 0 1
[1] 1 0
[1] 1 0
[1] 0 1
[1] 0 1
[1] 1 0
[1] 0.02868447 0.21274401 0.75857153
[1] 1 0 0
[1] 0 1
[1] 0.895596644 0.098982698 0.005420658
	REQUEST:
	"IdMessage":
	"a5c02448239c47fe8be2a17af23ff9e8",
	"sender": "ModelService",
	"receiver": "ModelService.R",
	"Credential": token,
	"language": "XMSG-SOAP",
	"inresponse": null,
	"Content": {
	"method": "LaunchRScript",
	"parameters": {
	"script": {
	"type": "xs:string",
	"value":
	"setwd("C:\\\\CHOREOGRAPHER\\\\WorkingDirectoryR\")\n AGE = 69, \n SEX = 2, \n ETHN = 1, \n C_SMK = 2, \n H_SMK = 2, \n H_STR = 2, \n H_HBG = 2, \n BMI = 28.50, \n WAIST = 92, \n SBP = 154, \n DBP = 96, \n AHT_M = 1, \n LLO_M = 1, \n CHOL = 237, \n GLO = 4.88888888888889, \n MS = 2
	.....
	v1<-posterior$AL_WK
	v2<-posterior$C_SMK
	v3<-posterior$H_SMK
	v4<-posterior$PHY_W
	v5<-posterior$PHY_F
	v6<-posterior$H_CVD
	v7<-posterior$H_STR
	v8<-posterior$H_HBG
	v9<-posterior$BMI
	v10<-posterior$WAIST
	v11<-posterior$SBP
	v12<-posterior$DBP
	v13<-posterior$AHT_M
	v14<-posterior$LLO_M
	v15<-posterior$CHOL
	v16<-posterior$TRIG
	v17<-posterior$HDL
	v18<-posterior$INS0
	v19<-posterior$GL0
	v20<-posterior$MS
	v21<-posterior$GL120"
	},
	"resultObjects": {
	"type": "xs:string",
	"value": "v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21"
	}}}}
	
	RESPONSE:
	{
	"IdMessage":
	"2f78fa985d854536a485cc6b77a1b8b3",
	"sender": "ModelService.R",
	"receiver": "ModelService",
	"Credential": token,

```
 "language": "XMSG-SOAP",
 "inresponse":
"a5c02448239c47fe8be2a17af23ff9e8",
 "Content": {
   "method": "LaunchRScriptResponse",
   "parameters": {
     "LaunchRScriptResult": {
       "type": "xs:string",
       "value":    "[\"0.58741910 0.25497897
0.02094656 0.11775952 0.01889585\",\"0 1\",\"0
1\",\"068429253 0.28048591
0.03522156\",\"0.7040993
0.2959007\",\"0.374341423298932
0.625658576701068\",\"0 1\",\"0 1\",\"0 1 0\",\"0
0 1\",\"0 1\",\"0 1\",\"1 0\",\"1 0\",\"0  1\",\"0
1\",\"1 0\",\"0.02868447 0.21274401
0.75857153\",\"1 0  0\",\"0
1\",\"0.895596643647435
0.0989826984480582 0.00542065790450731\"]"
}}}}
```


Figura 19: Comparación de la ejecución aislada e integrada del script en R para imputación de datos.

4.4.4.1. Integración de modelos

El primer paso fue comprobar que el script (o conjunto de scripts) que se iban a ejecutar en el motor estadístico (R y/o MATLAB) estuviera correctamente formateado. Para verificar esto se utilizó el servicio de trazabilidad proporcionado por CHOREMED, realizando un seguimiento de los mensajes intercambiados entre los componentes del sistema y su contenido. Antes del lanzamiento del sistema y en la versión de desarrollo del sistema, se ejecutó una consulta para cada uno de los modelos y se analizó el mensaje de seguimiento, como se describe en la Figura 19.

Se realizó una evaluación técnica para dos escenarios límite (el mejor y el peor de los casos representados en la Tabla 9). Los resultados se proporcionan en tablas y figuras, que representan una ventana de tiempo de 60 segundos de las operaciones descritas.

Tabla 9: Resultados de la evaluación técnica para el mejor y peor escenario en la escala de riesgo de predicción y el modelo de imputación de datos.

	Escala de riesgo de predicción				
	Pacientes	**Latencia (s)**	**CPU (%)**	**Memoria (KB)**	**Ancho de banda (Kbps)**
Mejor caso	1	0,016	20,20	374.012	9,8
Peor caso	8080	25,876	60,50	463.853	173,35
	Modelo de imputación de datos				
	Variables de entrada estimadas	**Latencia (s)**	**CPU (%)**	**Memoria (kB)**	**Ancho de banda (kbps)**
Mejor caso	0	1,486	48,50	360.416	40,23
Peor caso	20	1,860	49,5	360.748	63,56

En la Tabla 9, el peor caso para el modelo de imputación de datos ocurre cuando no hay parámetros de imputación, por lo que la red bayesiana tiene que realizar todas las operaciones para estimar las variables desconocidas; mientras que, si el modelo tiene todas las variables de entrada (20 en el mejor de los casos), no se necesita ninguna operación de estimación.

De la Figura 20 a la Figura 22 se muestra el rendimiento de ejecución de una escala de riesgo para el peor de los casos (ejecución sobre 8080 sujetos). La CPU se utiliza en un promedio de 60,5% durante 25,876 s. No se producen interrupciones por asignaciones de memoria, problemas de red o desbordamiento de CPU.

La Tabla 10 muestra el rendimiento del motor de base de datos para cada uno de los servicios clínicos (data marts en el almacén de datos). El uso promedio de la CPU es del 43,70% y la latencia depende de la cantidad de sujetos que se deben cargar. El peor de los casos se encuentra al cargar datos de laboratorio para 6.402 sujetos, lo que lleva 248 min para la carga de la configuración y 3,462 s para consultas posteriores. Este hecho puede deberse al tamaño de los datos y no al motor de consulta, que asigna los conceptos de ontología a los atributos de datos específicos en este data mart.

Tabla 10: Rendimiento del Módulo de Gestión de Base de Datos entre diferentes servicios y consultas periódicas.

Rendimiento del módulo de base de datos						
Servicio	**Número de sujetos**	**Tiempo de carga de la configuración (min)**	**Latencia por Paciente (s)**	**CPU (%)**	**Memoria (KB)**	**Ancho de banda (kbps)**
Urgencias	658	79	7,412			
Consultas externas	1020	67	1,766	43,70	137.733	720
Laboratorio	6402	248	3,462			
Consultas regulares	-	-	0,254	60,20	80.457	72.459

La Figura 20 muestra el uso de CPU (%) durante la ejecución de una escala de riesgo de predicción basada en un modelo de regresión logística para el peor de los casos. La línea naranja representa el motor de coreografía, que integra el script del modelo y ejecuta el algoritmo para n = 8080 pacientes. El uso de la CPU está por debajo del umbral del 83 %, que se encuentra por debajo del objetivo para una ejecución adecuada [89].

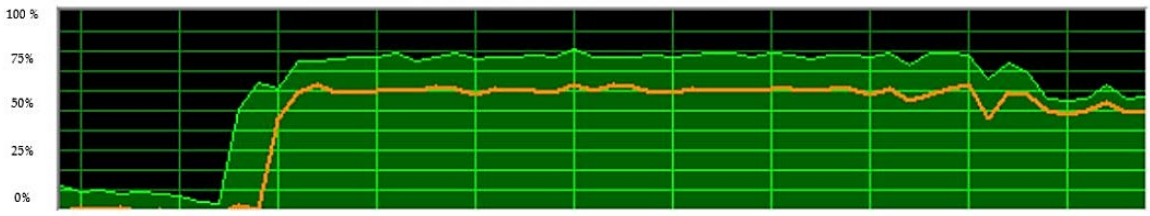

Figura 20: Uso relativo de CPU (%) para la ejecución de la escala de riesgo de predicción en el peor de los casos (Línea naranja).

La Figura 21 muestra el uso de la memoria para el mismo caso. La figura muestra un leve repunte que ocurre debido a la paginación automática de memoria realizada por el sistema operativo.

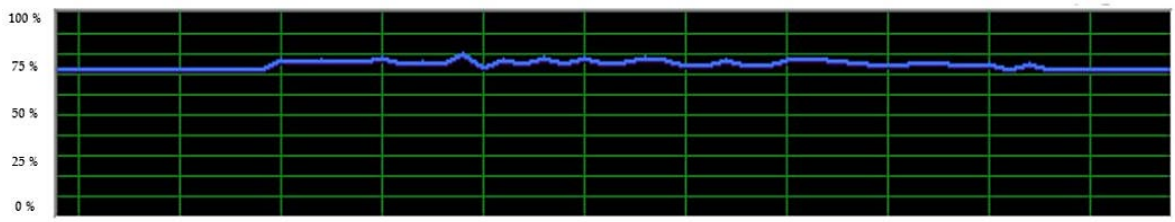

Figura 21: Uso de memoria de la ejecución del modelo de predicción para el peor de los casos.

La Figura 22 muestra al mismo tiempo cómo se gestionan los recursos de la red. El coreógrafo (línea naranja) muestra el momento en el que el servicio encargado de realizar consultas al módulo de almacenamiento de datos realiza una consulta para recuperar los datos de los n = 8080 pacientes, lo que conduce a un breve período de alta transferencia de datos (175.296 Kbps). Después de recuperar los datos para ejecutar los modelos, el módulo permanece sin más demandas de red.

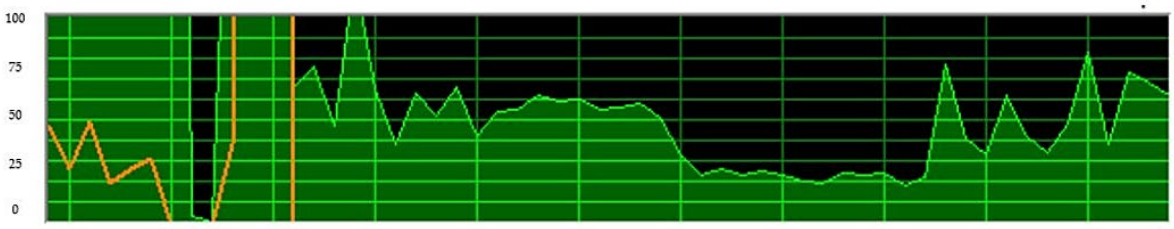

Figura 22: Recursos de red de la ejecución de la escala de riesgo de predicción para el peor de los casos (línea naranja). El desbordamiento de la gráfica se produce debido al modo de escalado automático del monitor de recursos. Pico = 175.296 Kbps.

4.5. Discusión

Los sistemas de salud deberían cambiar para realizar campañas proactivas sobre la promoción de la salud y la prevención de enfermedades. La explosión de los HIS, las tecnologías de almacenamiento de datos y la inteligencia artificial ha allanado el camino para afrontar los retos de una población cada vez más sedentaria y envejecida. Sin embargo, todavía existe una brecha entre los resultados de la investigación y las aplicaciones clínicas.

En este capítulo se ha presentado una arquitectura distribuida basada en coreografía de procesos, utilizada para validar que es posible integrar modelos híbridos de inteligencia artificial con la EHR. Ésta se ha utilizado para proporcionar herramientas de apoyo a la toma de decisiones en la identificación de sujetos con alto riesgo de DM2. Los resultados obtenidos en la validación de la implantación de la arquitectura en un piloto real demuestran que es factible el enfoque propuesto, lo cual era uno de los propósitos iniciales de este trabajo.

Como en todo despliegue, al implantar el sistema en el entorno real surgieron algunos problemas técnicos menores, por tener que integrar múltiples sistemas heterogéneos. Gracias a la arquitectura basada en coreografía de procesos y el uso del coreógrafo CHOREMED, éstos pudieron resolverse rápidamente con pequeñas modificaciones, sin afectar la ejecución del piloto, lo que permite respaldar las ventajas de aplicar el paradigma de coreografía para integrar sistemas de terceros.

Las razones por las que los médicos y los investigadores no son propensos a utilizar modelos predictivos se identifican como la falta de confiabilidad y la falta de validación de los modelos, ya que, en la mayoría de los casos, esto se hace sólo como validación interna. La arquitectura híbrida distribuida propuesta en este documento implementa una integración y coordinación mediante coreografía de procesos de los tres módulos principales necesarios para superar las barreras antes mencionadas [94]. Al utilizar la interacción de los servicios mediante el paradigma de coreografía, el sistema es capaz de modificar los flujos de trabajo para adaptarse a las diferentes funcionalidades y proporcionar el mejor resultado.

Utilizando la tecnología I2B2, los datos de EHR se integraron mediante la definición de una ontología común que abarca todos los diferentes parámetros, entre ellos en el módulo de Data Storage. Junto a este módulo, el módulo Model Host, utilizando el paradigma de coreografía de procesos y desarrollado con la tecnología de CHOREMED, reúne los modelos de discriminación (red bayesiana) y predictivos (regresión logística/modelo de supervivencia de Cox) para evaluarlos con las fuentes de datos.

Estos modelos utilizaron directamente el código script, que codifica el algoritmo del modelo, para ejecutarlo directamente sobre los motores de ejecución (R y Matlab), en su

uso para la discriminación y predicción de DM2 de forma independiente. Este enfoque, permitió realizar mejoras en el algoritmo del modelo sin necesidad de parar el componente y volver a compilar, desplegar y depurar un componente nuevo. En estos términos, un modelo puede ser validado externamente dentro de la misma infraestructura del sistema y, por lo tanto, ser proporcionado a los usuarios finales a través de la integración de la herramienta de discriminación o predicción en el sistema de gestión de software actual utilizada en el entorno clínico.

Tal como muestran los indicadores de rendimiento, el sistema permitió a los clínicos acceder a los datos clínicos y ejecutar los diferentes modelos para detectar sujetos con alto riesgo de DM2 con éxito. En la Figura 20 a Figura 22 se muestra cómo el sistema fue capaz de ejecutar los modelos de regresión logística, en el peor de los casos de forma estable, con porcentajes de CPU del 60,5% inferiores al valor que la literatura define como adecuado del 83% [89]. Además, se observó que la carga de memoria en el peor de los casos era de 463.853KB, lo que supone sólo 89.841KB más respecto al mejor de los casos, siendo ambos valores aceptables y fácilmente soportables por cualquier servidor de rango medio-bajo, como el empleado, que disponía de 24G de RAM.

Donde se experimentaron los mayores retrasos fue en la carga de la configuración inicial de grandes volúmenes de datos en el modelo ontológico de I2B2, como se observa en la Tabla 10, donde para datos pesados, como los de laboratorio, cuando se cargan 6402 pacientes se requiere de 248 minutos. No obstante, una vez cargada la configuración inicial la latencia por paciente es de 3,462 segundos, por lo que, tras la inicialización, el sistema rinde de forma eficaz.

El correcto funcionamiento del sistema presentado demuestra que el paradigma de coreografía de procesos tiene la fortaleza de integrar de forma fácil y robusta elementos complejos de terceros, como los motores matemáticos R/MATLAB, para los que hubo que desarrollar integraciones específicas o sistemas de acceso a datos I2B2 DW disponibles mediante servicios web. Esto se realizó creando servicios que colaboran entre sí de forma distribuida para formar coreografías, que ofrecen las funcionalidades del sistema. Esto, a su vez, permitió realizar la integración de múltiples modelos para la detección temprana de DM2 utilizando técnicas de modelado híbrido.

Las aplicaciones probadas en el estudio piloto se centraron en la ejecución de escalas de riesgo para DM2; sin embargo, el enfoque modular impulsado por la arquitectura coreografiada, permitió la integración de otro tipo de aplicaciones, modelos y bases de datos. Consiguiendo así que los modelos híbridos funcionaran juntos, evitando fallos en el sistema, excepciones y tiempos de retraso excesivos. Esto resultó en un flujo de trabajo fluido y sin cortes, lo que puede contribuir a generar una mayor satisfacción de uso por parte del personal clínico (habiendo sido expresada esta mayor satisfacción de uso de forma oral por varios de los profesionales participantes en el piloto).

Los tipos de modelos integrados en esta plataforma se basan en ecuaciones matemáticas y probabilísticas, que son fáciles de ejecutar en poblaciones reducidas, pero difíciles en poblaciones grandes [95]. El enfoque de distribuir las operaciones para el almacenamiento de datos y la ejecución del modelo fue crucial para lograr un desempeño técnico razonable. Además de las limitaciones técnicas, según el almacén de datos y los motores matemáticos necesarios, la implementación podría requerir que el centro clínico compre licencias de software especiales para ejecutar los modelos (R no requiere licencia, pero MATLAB sí). Además, el uso de clientes web, como interfaces gráficas de usuario, permitió a los médicos acceder a las herramientas desde cualquier ordenador y tableta. Los ordenadores de las oficinas clínicas tienen recursos computacionales muy limitados, por lo que, aprovecharlos sólo para interactuar con un software de servidor que sí tiene alta cantidad de recursos, resultó en una ejecución eficiente de las operaciones ETL (Extract, Transform, Load) y la evaluación de modelos de riesgo.

Una limitación de la arquitectura propuesta es que prima la flexibilidad sobre la eficiencia. Si las escalas de riesgo y los algoritmos de imputación no requirieran cambios, la solución elegida hubiera sido integrar modelos compilados basados en C/C++ o herramientas específicas del motor de ejecución [96], mejorando la eficiencia y uso de recursos. Sin embargo, uno de los principales requisitos era proporcionar una arquitectura flexible, capaz de modificar en el entorno de ejecución los parámetros de los modelos e incluso las consultas de datos. Por lo tanto, se implementaron diversos servicios que, mediante las coreografías adecuadas, han ofrecido la misma funcionalidad, pero permitiendo una modificación dinámica de los algoritmos y un seguimiento del flujo de datos transparente (Figura 19).

4.6. Conclusiones

El presente manuscrito ha descrito la utilidad y confiabilidad de una arquitectura basada en coreografía de procesos para integrar modelos híbridos para su uso en entornos clínicos. Esto incluye la integración con sistemas de terceros y su aplicación a un entorno hospitalario real. La medicina basada en la evidencia requiere marcos tecnológicos para implementar y probar los resultados de la investigación en escenarios clínicos. Se ha demostrado que los modelos de riesgo de DM2 funcionan bien en ensayos clínicos retrospectivos y prospectivos; sin embargo, aún se desconoce su rendimiento utilizando conjuntos de datos de población. Nuestro estudio piloto ha confirmado la capacidad de un sistema distribuido basado en la coreografía de procesos para integrar técnicas de modelado heterogéneas, fuentes de datos clínicos e interfaces de usuario basadas en la web, lo que allana el camino hacia la implementación de estudios de evaluación en conjuntos de datos de población real.

5. Sensores Portables Integrados con el Internet de las Cosas para Avanzar en la Atención de la eSalud

En este capítulo se presenta el estudio técnico de la utilización de tecnología de coreografía para demostrar que es posible crear una arquitectura que integre dispositivos portables y sensores del ámbito IoT, con el objetivo de proporcionar servicios de telemonitorización que apoyen el cuidado del paciente, sustituyendo así a los sistemas fijos más complejos y caros. Dicho estudio se presentó en una publicación en la revista Sensors en el año 2018 [34] y permitió validar el tercero de los objetivos secundarios de esta tesis HS3, que plantea que *"el paradigma de coreografía de procesos permite el desarrollo de plataformas de eSalud orientadas al Internet de la Cosas".*

Los indicadores sanitarios y sociológicos alertan de que la esperanza de vida es cada vez mayor, por lo que también lo son los años de vida de los pacientes con enfermedades crónicas y comorbilidades. Con el avance de las TIC, se exploran nuevas herramientas y paradigmas para brindar una atención médica eficaz y eficiente. La telemedicina y los sensores de salud son herramientas indispensables para promover la participación del paciente, el autocontrol de enfermedades y ayudar a los médicos a realizar un seguimiento remoto de los pacientes. En este capítulo, se evalúa una solución de prototipado rápido para la integración de información basada en cinco sensores de salud y dos componentes de computación ubicua de bajo costo: Arduino y Raspberry Pi, comparando con la misma solución desplegada en un ordenador personal (Personal Computer, PC). La integración se realizó utilizando un motor de coreografía (CHOREMED) para transmitir datos desde sensores a una interfaz web y un protocolo de comunicación. El rendimiento de las dos configuraciones se comparó por medio de la latencia en la transmisión de datos desde los dispositivos y la pérdida de datos.

5.1. Introducción

El Internet de las cosas (Internet of Things, IoT) es un marco en el que los sensores, dispositivos y actuadores se pueden gestionar de forma ubicua y distribuida [97]. El sector de la salud no está ajeno a la revolución del IoT y ya existen múltiples aplicaciones y servicios para mejorar la calidad asistencial [98].

En este contexto, la telemedicina es un escenario ideal para la expansión y mejora de las tecnologías IoT en salud [99]. La monitorización remota mediante sensores accesibles y fáciles de usar son la vanguardia en la aplicación de este tipo de tecnologías [100].

Los sistemas distribuidos para la monitorización remota se han presentado en múltiples ocasiones [101] [102] y describen dos tipos principales de arquitecturas. Por un lado, el primer tipo de sistemas permite almacenar variables biométricas, de comportamiento y de contexto de sensores comerciales en dispositivos (teléfonos móviles, tabletas y computadoras) para generar informes completos que apoyen la toma de decisiones relacionadas con la salud [103]. Por otro lado, el segundo tipo de sistemas reenvía automáticamente los datos adquiridos (sin almacenarlos) usando transmisiones inalámbricas Bluetooth o Wifi [104] [105] a servidores centrales para su posterior explotación.

Uno de los pilotos más grandes realizados hasta la fecha sobre monitorización fue el realizado en el *Whole System Demonstrator Programme* (WSD) [106]. Este programa fue impulsado por el Sistema Nacional de Salud del Reino Unido para estimular la adopción de la teleasistencia. El objetivo principal era proporcionar a más de 6000 pacientes herramientas para manejar sus condiciones crónicas, con una estricta supervisión del personal clínico (hasta 238 médicos), mediante el uso de sensores para monitorizar señales fisiológicas integradas en un complejo sistema de comunicación.

El WSD constaba de tres implementaciones con diferentes opciones tecnológicas, pero la arquitectura era la misma: una unidad base para visualizar datos y cuestionarios, así como dispositivos periféricos de seguimiento de la salud. En cada punto desplegado se utilizaron diferentes protocolos para la asignación de sensores: un pulsioxímetro, un glucómetro, balanzas, etc. Los datos se transmitieron desde la unidad base a un centro de monitorización empleando una conexión segura a Internet. Estos sensores eran capaces de monitorizar variables importantes como la glucosa en sangre, el peso corporal, el oxígeno en sangre, el pulso y la presión arterial, entre otras. Sin embargo, dichos sensores no estaban integrados con la unidad base y los dispositivos móviles en todos los casos. El informe provisional de resultados del piloto mostró algunas mejoras, pero el informe final concluyó que el grupo de intervención no se estaba beneficiando del uso de los cuidados remotos [107].

Los autores de WSD indicaron que el impacto de las intervenciones de atención remota depende de la arquitectura y el rendimiento del sistema, como confirmaron posteriormente otros autores [93]. Existen herramientas cualitativas y cuantitativas para medir la respuesta del usuario a un sistema de telemedicina [108]; sin embargo, se debe disponer de detalles sobre la implementación y la evaluación técnica para poder poner los resultados en un contexto realista. Los resultados clínicos pueden verse distorsionados por errores de transmisión, duplicación y pérdida de datos debido al vencimiento de los tiempos de espera.

La Personal Connected Health Alliance (PCHA) (http://www.pchalliance.org/continua/) reúne a más de 200 fabricantes de sensores de salud y empresas de software para impulsar dispositivos eSalud interoperables y crear soluciones totalmente integradas. PCHA se estableció como una organización sin ánimo de lucro, para promover la adopción de estándares de dispositivos médicos (hardware o software) como una forma de construir soluciones complejas basadas en el paradigma IoT. Al mismo tiempo, la Organización Internacional de Normalización (ISO) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) lanzaron el compendio de Normas de Comunicación 11.073 que describe el comportamiento, intercambio de información, nomenclatura y reglas de conexión de los dispositivos de salud y bienestar para ser integrados en diferentes escenarios operativos: desde redes de área corporal hasta sistemas distribuidos.

Sin embargo, la complejidad de este estándar ha limitado la adopción generalizada en el ecosistema de dispositivos médicos y portátiles [109]. Además, los productos certificados por PCHA suelen ser más caros que el mismo producto sin el estándar de comunicación, siendo su adopción en sensores comerciales testimonial [110].

Conectar sensores de salud siguiendo el paradigma IoT podría ser una forma fácil y rápida de implementar intervenciones de telemedicina complejas, ya que en IoT no se implementan reglas de conexión complejas y nomenclaturas profundas, si no que se pone un enfoque especial en la simplicidad, la interoperabilidad y la trazabilidad como base para la integración de sensores en un sistema de gestión de la salud.

Health Level 7 Association (HL7) lanzó el protocolo Fast Health Interoperability Resources (FHIR) [111], una versión liviana del Protocolo de control HL7 y el modelo de información de referencia, con el objetivo de atraer a los desarrolladores para que construyan soluciones interoperables eficientes [112]. Bluetooth Low Energy (BLE) define un perfil especial para dispositivos de salud y estado físico, pero ha mostrado varias limitaciones de implementación, que pueden reducir el rendimiento de BLE en un escenario real en comparación con lo esperado teóricamente [113] [114].

Teniendo en cuenta las contribuciones de BLE y FHIR, las principales deficiencias para crear prototipos de nuevas soluciones de eSalud bajo el paradigma IoT son la sobrecarga innecesaria de intercambio de datos y las dificultades para construir escenarios de demostración [115]. Al nivel de la comunicación sensor-dispositivo (por ejemplo, un sensor de electrocardiografía con un dispositivo móvil), hay una gran cantidad de encabezados y descriptores de datos que sólo son útiles para capas de comunicación altas. Si bien los mensajes deben controlarse mediante métricas y procedimientos de calidad estándar, agregar datos innecesarios a las interfaces inalámbricas puede causar más problemas de los que realmente pretenden resolver. Además, cuando los pacientes y los profesionales de la salud intercambian datos, es necesario implementar una interfaz gráfica de usuario (GUI), por ejemplo, una página web que se ejecuta en un servidor web

específico (o incluso una aplicación móvil), lo que agrega más complejidad en la interconexión de sistemas distribuidos.

En este documento, se presenta y evalúa un sistema distribuido embebido con un protocolo personalizado y ligero, para la conexión de dispositivos de salud económicos basados en sensores para el prototipado en eSalud (Arduino, Raspberry Pi y un kit de sensores). Todos los componentes característicos de un sistema IoT están interconectados usando el paradigma de la coreografía de procesos [116] [27].

Este estudio busca demostrar la hipótesis de que es posible crear una arquitectura de coreografía de procesos que integre dispositivos portables y sensores del ámbito IoT, que proporcione servicios de telemonitorización para el cuidado del paciente, pero reduciendo la complejidad del sistema (usando protocolos más ligeros, mayor flexibilidad en la integración de sensores).

Para evaluar hasta qué punto los dispositivos portables se pueden comparar con los sistemas fijos, en la implementación general se configuraron dos escenarios diferentes: (1) ordenador de escritorio con sistema operativo Windows 10; y (2) Raspberry Pi con sistema operativo Windows 10 Core IoT. En ambos casos, se ofrecieron las mismas funcionalidades para solicitar y recuperar datos de los sensores de salud, además de alojar el interfaz de usuario basado en web. Para comparar ambos escenarios se utilizaron indicadores de rendimiento, como el tiempo de demora entre la adquisición y la visualización de la señal.

A continuación, se presentan los diferentes elementos utilizados para construir el sistema y el diseño de los experimentos de validación.

5.2. Materiales y métodos

En este apartado, se describen los materiales y la metodología usados para probar los dos escenarios de implementación para un sistema de telemonitorización. Primero se describe el hardware empleado. Posteriormente, la arquitectura y aplicación de la coreografía de procesos para la integración. Finalmente, se describe el montaje experimental.

5.2.1. Conjunto o Kit de sensores de eSalud

Una de las plataformas de prototipado rápido y desarrollo de hardware que más auge ha experimentado en los últimos años, y que está contribuyendo a democratizar el uso de dispositivos IoT, es la plataforma Arduino. Esta plataforma de especificación abierta está

basada en un microcontrolador, generalmente de la familia AVR Atmel-8 bits de capacidad reducida, pero que permite ejecutar programas de control de hardware.

Arduino proporciona un hardware fácil de usar, pero efectivo para conectar y usar múltiples componentes electrónicos con una amplia variedad de aplicaciones [117] [118]. El sitio web oficial de Arduino [119] tiene una colección completa de información, descargas, tutoriales y ejemplos sobre cómo usar la plataforma (así como de las diferentes versiones de placas disponibles sobre la misma plataforma, desde entornos educativos a profesionales).

En este estudio se utilizó la versión Arduino UNO, orientada a entorno educativo o de laboratorio, por su tamaño y número de pines de conexión. La comunicación con dicha placa se realiza mediante protocolo serie por el puerto USB o conectándole componentes de comunicación externos (Bluetooth, Wifi, etc.)

Arduino proporciona una excelente plataforma para probar y crear prototipos de soluciones mediante la agregación de módulos que se encajan sobre la placa (denominados *shields)* o de hardware compatible conectable a los pines de comunicación (Bluetooth, WiFi, LEDs y servomotores). En esta investigación, se ha utilizado el kit e-Health Sensor Platform V1.0, creado por Libelium [120]. Este kit permite a los usuarios adquirir un conjunto de señales fisiológicas ECG, EMG, frecuencia respiratoria, temperatura superficial y GSR (Tabla 11). Dicho kit ha sido utilizado anteriormente en investigaciones de sensores portables [121] [122].

El hardware de Libelium no es un dispositivo médico certificado. Se utilizó esta plataforma como un dispositivo médico virtual porque implementa las mismas interfaces físicas y circuitos de monitorización que los sensores certificados y comerciales. Además, esto evitó tener que desarrollar protocolos propietarios del fabricante del dispositivo y tener que conseguir acuerdos de confidencialidad. Los fabricantes no siempre están abiertos a proporcionar estos protocolos y acuerdos sin un acuerdo económico de por medio.

Tabla 11: Sensores del kit e-Health Sensor Platform V1.0.

Nombre del sensor	Descripción	Tipo de Datos
Sensor de temperatura	La temperatura corporal depende del lugar del cuerpo en el que se realiza la medición, la hora del día y el nivel de actividad del sujeto. Diferentes partes del cuerpo tienen diferentes temperaturas.	Discretos
Sensor de flujo de aire	La frecuencia respiratoria es un indicador amplio de inestabilidad fisiológica importante. El sensor mide el flujo respiratorio de una persona en las vías respiratorias superiores (nariz). El sensor de flujo de aire también puede proporcionar una advertencia de hipoxemia y apnea.	Continuos

Sensor Galvánico de Respuesta de la Piel (GSR)	Se puede utilizar para medir la conductancia eléctrica de la piel, que varía con su nivel de humedad. La conductancia de la piel se utiliza como indicación de excitación psicológica o fisiológica. GSR mide la conductancia eléctrica entre 2 puntos y es esencialmente un tipo de medidor de la resistencia eléctrica u ohmímetro.	Continuos
Sensor de electrocardiografía (ECG)	Una herramienta de diagnóstico que se utiliza de forma rutinaria para evaluar las funciones eléctricas y musculares del corazón. El ECG se ha convertido en una de las pruebas médicas más utilizadas en la medicina moderna. Algunas enfermedades no muestran modificaciones en la forma de onda del ECG.	Continuos
Sensor de electromiografía (EMG)	Se puede utilizar como herramienta de diagnóstico para evaluar y registrar la actividad eléctrica producida por los músculos. Midiendo la actividad eléctrica de los músculos en reposo y durante la contracción. La EMG se utiliza para identificar enfermedades neuromusculares, evaluar el dolor lumbar, la kinesiología y los trastornos del control motor.	Continuos

Toda la recogida de señales biométricas realizada se basa en el kit de e-Health, que se muestra en la Figura 23. El kit está formado por un shield, que se monta sobre una placa Arduino para recopilar la información biométrica de los diferentes sensores conectados a dicho shield.

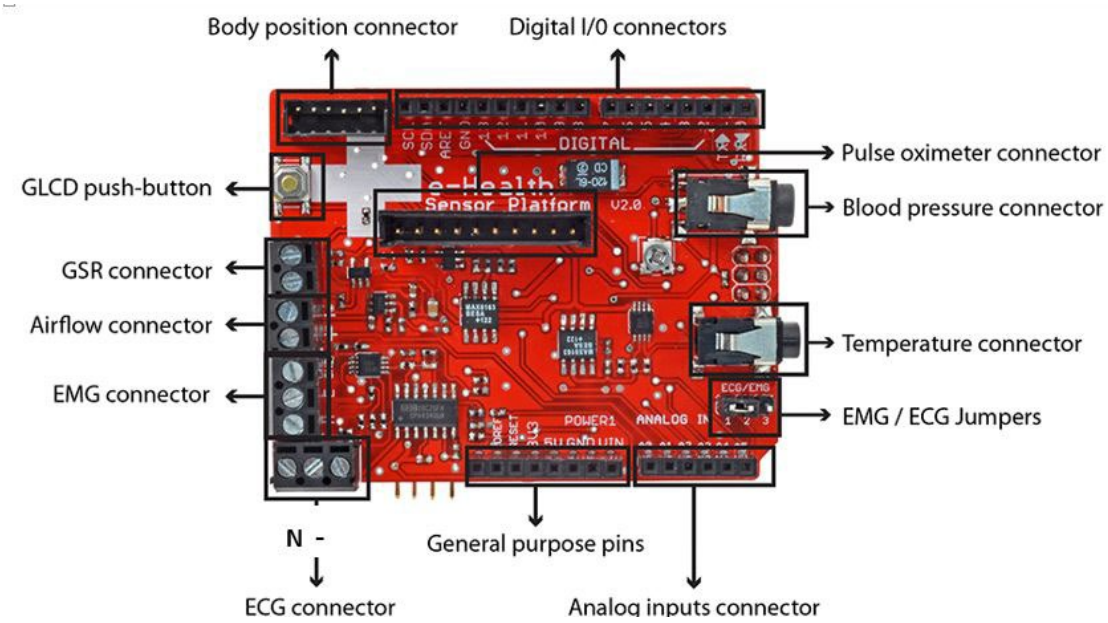

Figura 23: Pines de entrada/salida del kit de e-Health [26].

Versiones certificadas de sensores que miden este tipo de señales, se usan como herramientas de diagnóstico en entornos clínicos y también para monitorizar de forma remota el estado de un paciente en tiempo real. Algunos de los sensores que utilizan técnicas electrofisiológicas (p. ej., ECG, EMG y GSR) requieren una serie temporal para

darles sentido, por lo que se visualizan en forma de gráficas en el tiempo, en lugar de como valores aislados.

5.2.2. Sistema Embebido Portátil

La plataforma Arduino y el kit de e-Health presentados permiten la adquisición de los datos, pero sus capacidades en cuanto a almacenamiento de los datos o capacidad de proporcionar interfaces gráficas para la gestión de los sensores es limitada. Por este motivo, como ya se ha comentado en la introducción, es habitual el uso de una base o elemento de control de mayores capacidades que concentra los diferentes sensores. En este caso, para el estudio se empleó un único elemento Arduino más sus sensores asociados, pero la arquitectura propuesta permite el uso de múltiples sensores sobre la misma base.

Siguiendo el espíritu de utilizar plataformas de prototipado rápido y educativas, para la creación de la base se eligió el dispositivo portable Raspberry Pi. Éste se introdujo en 2012 [123] y ha sido ampliamente utilizado en entornos de monitorización domésticos [124] [125]. En concreto se ha utilizado la versión Raspberry Pi 3 modelo B, que integra un procesador ARMv8 a 1,2 GHz y 1 GB de RAM en una placa reducida, junto con soporte para múltiples periféricos de entrada y salida a través de interfaces estándar (Pines digitales y analógicos, bus serie, USB, Bluetooth, Ethernet, Wifi, HDMI) y permite la instalación de varios sistemas operativos. Las dimensiones del dispositivo son 85,60 mm x 53,98 mm x 17 mm, con un peso aproximado de 45 gramos.

Figura 24: Placa Raspberry Pi 3 Modelo B

La placa dispone de puerto de vídeo HDMI para conectar un monitor o televisor, pero para dotar al sistema de mayor portabilidad se ha conectado la placa a una pantalla de 7 pulgadas de tipo táctil, que permite mostrar información e interactuar con el software instalado. Todo el conjunto se encierra en una caja compacta para una mayor practicidad.

El sistema puede ser alimentado mediante un cargador de 5 voltios o mediante baterías portátiles.

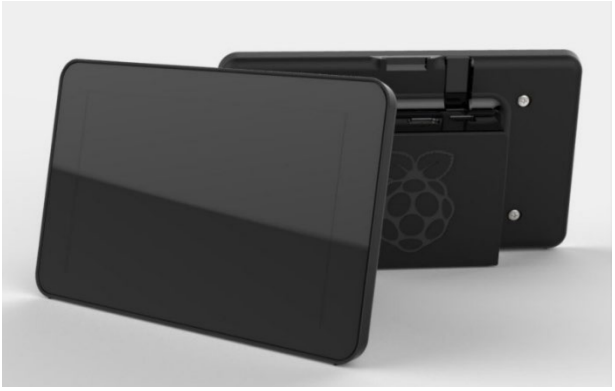

Figura 25: Pantalla y caja de transporte

El sistema operativo instalado para su funcionamiento fue Microsoft Windows 10 IoT Core [126]. Windows 10 IoT Core es una versión optimizada de Windows 10 para dispositivos ARM portátiles pequeños y x86/x64.

Uno de los objetivos de este estudio era evaluar en qué medida los sistemas portables están preparados para sustituir a las computadoras en la forma en que conectan y procesan la información proveniente de varios sensores de eSalud. Con este fin, se evaluó la Raspberry Pi 3 en comparación con un ordenador de escritorio.

5.2.3. Arquitectura hardware

El sistema de telemonitorización desarrollado tiene como requisitos ser portable, utilizar múltiples componentes IoT para la recogida de datos y ser accesible tanto desde la base de control, como de forma remota.

El hardware elegido para los sensores permite utilizar múltiples sensores de tipo IoT ya existentes en el mercado. Por ejemplo, es posible conectar múltiples Arduino al sistema para ampliar la información recogida. Aunque el Arduino permite comunicación inalámbrica, en nuestro caso se ha empleado conectividad Serie por puerto USB, como se puede ver en la Figura 26. Esta decisión se tomó por el alto volumen de datos generado por las señales continuas del ECG, GSR o EMG.

Para el acceso a la información y tareas de control se optó por proporcionar el acceso mediante interfaz web, ya que se quiere permitir el acceso remoto a la base. Dicho interfaz puede ser accedido localmente desde la pantalla de la base o remotamente desde un ordenador o tableta.

La Figura 26 muestra una vista completa desde un punto de vista físico de los diferentes componentes de la arquitectura. Se puede ver cómo el cliente se conecta a un interfaz web proporcionado por la base, dicha base ejecuta el software de control y será un ordenador fijo o una Raspberry Pi 3, según el escenario. A dicha base se podrán conectar múltiples sensores de tipo IoT, como, por ejemplo, el Arduino y el kit e-Health que se ha descrito con anterioridad y que se conectará por USB.

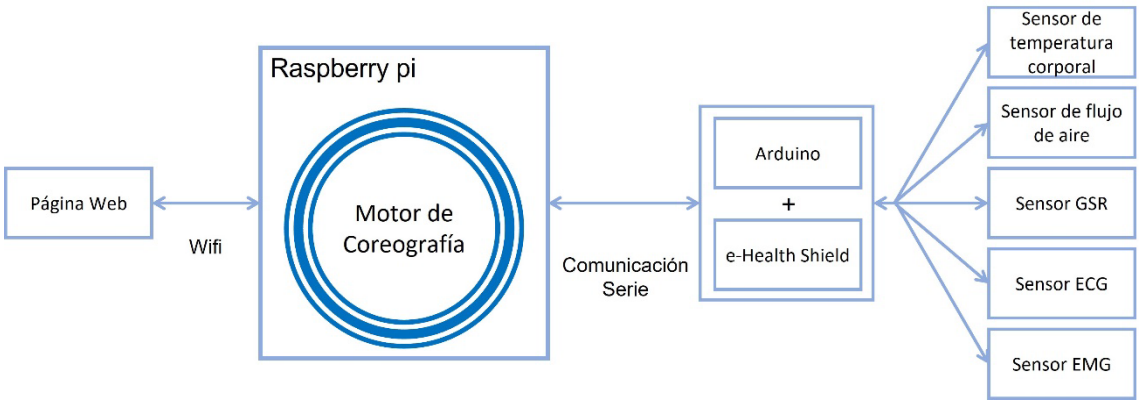

Figura 26: Arquitectura física del sistema que incluye sensores del kit de e-Health.

5.2.4. Protocolo de comunicación

El kit e-Health proporciona unas librerías de control de los sensores, pero se requiere realizar un programa o sketch de Arduino que defina el comportamiento de los sensores. Es decir, cuándo deben leerse datos o cómo transferirlos. Del mismo modo, cualquier sensor que se incorpora a la plataforma debe ser controlado, y para eso se necesita de un lenguaje común que indique a cada sensor qué acción realizar y cuándo. Por este motivo, se definió un protocolo extensible de comunicación propio para interactuar con los diferentes sensores.

Lo primero que se definió fueron los modos operativos del sensor. Un sensor puede trabajar en dos modos operativos:

- **Modo activo:** En este modo el sensor recopila y envía automáticamente las medidas de forma periódica conforme a un intervalo configurado previamente. Esto permite mostrar una gráfica con la evolución de los datos en tiempo real.
- **Modo pasivo:** En este modo el sensor no envía medidas, si no que está a la espera y, cuando le llega un comando solicitando una medida, la captura y la envía en ese momento concreto.

Teniendo en cuenta los dos modos de funcionamiento ya se vislumbran los diferentes comandos necesarios en los mensajes. Se requiere poder enviar comandos para obtener

un dato, para enviar un dato, para configurar un modo de operación (tiempos de intervalo) y poder cambiarlo.

Además, también se debe agregar un mecanismo de detección de errores, ya que los mensajes pueden corromperse durante la transmisión.

Tal como se muestra en la Tabla 12 y la Tabla 13, los mensajes tienen una estructura de tres secciones: Encabezado, Cuerpo y Cola. Donde el Encabezado contiene información general del mensaje, el Cuerpo incluye la información del comando y la Cola contiene elementos de protección. Por lo tanto, el protocolo debe considerar los siguientes descriptores:

- **Tipo de kit:** Identifica el tipo de kit de sensor utilizado, este campo tiene valor fijo en nuestro caso, pero deja el formato preparado para extensiones a otro tipo de sensores o entornos IoT como hogares inteligentes.
- **Destino:** Nombre del sensor que identifica el destino del comando.
- **Comando:** Puede tomar el valor de uno de los diferentes comandos que se pueden usar para solicitar los datos del sensor o cambiar algunas configuraciones predeterminadas.
- **Parámetro:** Este campo complementa al descriptor Comando para incluir parámetros o información de entrada. Por ejemplo, para cambiar algunas configuraciones, como el intervalo de tiempo en modo activo.
- **Tipo de sensor:** nombre del sensor que envía datos o una respuesta.
- **Respuesta a:** este campo complementa el descriptor Tipo de sensor e indica a qué comando está respondiendo este sensor. Esto permite la comunicación asíncrona.
- **Datos:** Complementa al descriptor Tipo Sensor. Ésta es la parte que contiene los datos medidos por los sensores o la respuesta del comando.
- **Unidad:** Este campo corresponde al descriptor Datos y es la unidad en la que se miden dichos datos.
- **Suma de verificación:** el campo de suma de verificación es la suma de 16 bits de todos los bytes de la sección Cuerpo. Se puede utilizar para detectar errores que podrían introducirse durante la transmisión.

El formato del paquete de datos del sensor y el paquete de respuesta se muestran en la Tabla 13.

Tabla 12: Formato de comando basado en texto.

Encabezado		**Cuerpo**		**Cola**
Tipo de kit	Destino	Comando	Parámetro	Suma de verificación
		Delimitador de campo	'|'	

Tabla 13: Datos del sensor y formato de respuesta.

Encabezado	Cuerpo				Cola
Tipo de kit	Tipo de sensor	Respuesta a	Datos	Unidad	Suma de verificación
		Delimitador de campo	'|'		

5.2.5. Integración mediante coreografía de procesos

Para conseguir el objetivo de integrar diferentes sensores de IoT en una arquitectura altamente interoperable usando protocolos ligeros, se ha utilizado el paradigma de coreografía de procesos. Este paradigma, permite crear diferentes servicios para gestionar cada sensor (el servicio se encargará de enmascarar la complejidad interna del mismo) y proporcionar su funcionalidad al resto de servicios mediante el envío de mensajes. La cooperación entre los diferentes servicios, es decir, la coreografía, genera la funcionalidad percibida por el usuario final.

Para aplicar el paradigma de coreografía se ha utilizado el coreógrafo CHOREMED, ya descrito en los apartados anteriores. Dicho coreógrafo permite la creación ágil de servicios y el envío de mensajes de coreografía entre ellos (formato XMSG).

Los servicios fueron creados para encajar en tres categorías:

- **Comunicación serie:** Se creó un servicio de comunicación en serie que establece la conexión con el hardware Arduino para leer datos de sensores portables.
- **Traducción de datos:** Para cada sensor se creó un servicio que traduce, tanto los datos que llegan del sensor, como los comandos enviados al sensor, de acuerdo con el protocolo de comunicación predefinido. También verifica los paquetes de datos para descartar errores de envío.
- **Cliente web:** La parte del cliente se define como un servicio web para ofrecer una interfaz de comunicación con un sensor (dicho servicio se instancia una vez por cada sensor disponible), y el servicio para alojar la página web del cliente. La web del cliente se comunica con el sistema empleando dichos servicios web.

Teniendo en cuenta los servicios creados, el flujo de información corresponde con el que se muestra en la Figura 27. Cuando un usuario quiere interactuar con un sensor genera una petición a través del servicio web del sensor, éste se comunica con el traductor, que a su vez genera la petición en el formato de dicho sensor. A continuación, el servicio de comunicación serie lo hace llegar al Arduino. Cuando el Arduino gestiona la petición devuelve el resultado a través del servicio de comunicación serie, el traductor adecuado recoge el mensaje, transforma los datos al formato adecuado y lo transmite al servicio web solicitante, que, finalmente, lo hace llegar a la interfaz web.

Figura 27: Integración de la coreografía.

Los servicios creados se muestran en la Tabla 14 conforme a las categorías mencionadas.

Tabla 14: Clasificación y listado de servicios.

Categoría	**Servicios**
Comunicación serie	Servicio de comunicación serie
Traducción de datos	Traductor de flujo de aire Traductor de temperatura corporal Traductor de ECG Traductor EMG Traductor de GSR
Cliente web	Servicio de alojamiento de la página web Servicio web del sensor

Tal como ya se ha indicado, el servicio de traductor funciona como un middleware al conectar el servicio web y el servicio de comunicación en serie mientras se intercambian datos entre entidades. La Figura 28 y la Figura 29 muestran el diagrama de secuencia de los servicios de las tres categorías (tanto para el modo activo, como para el modo pasivo).

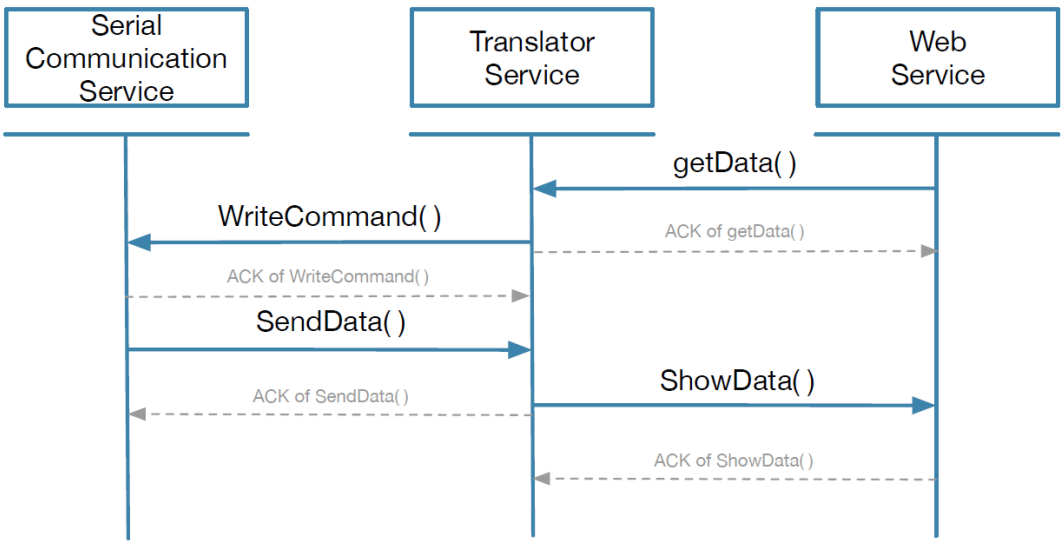

Figura 28: Modo pasivo.

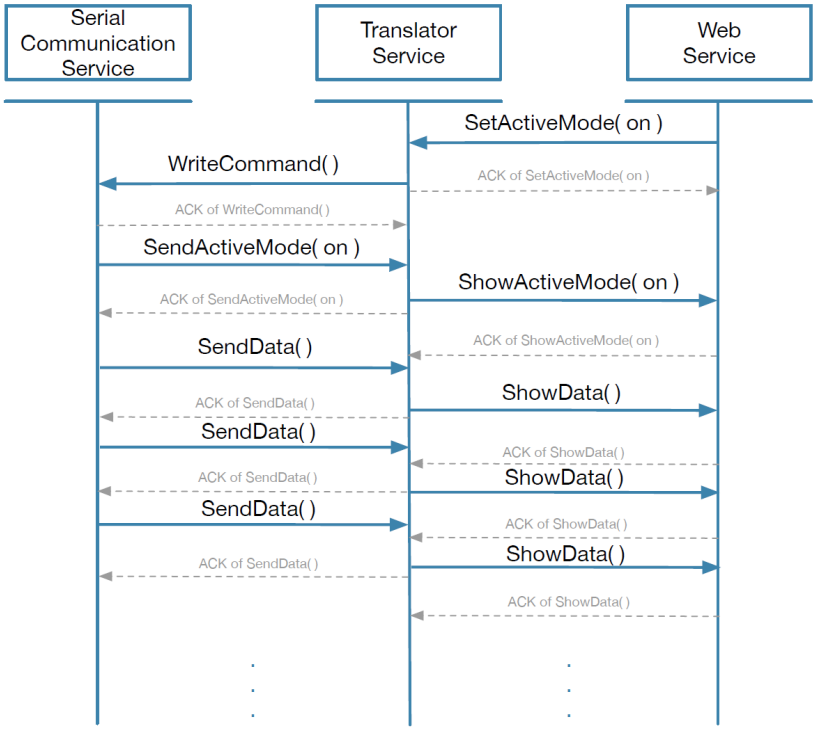

Figura 29: Modo activo.

Ilustrado por el diagrama secuencial en la Figura 29 y la Tabla 15, el servicio de traducción es una de las partes más cruciales de la arquitectura general, ya que es el único servicio que implementa el protocolo de comunicación. Cada sensor tiene un traductor y se utilizan diferentes métodos para analizar la información proveniente del servicio de comunicación en serie, o encapsular varios comandos recibidos del servicio web en paquetes de comando.

Tabla 15: Métodos implementados para el intercambio de información entre los servicios de comunicación y su descripción.

Servicio web → Servicio de comunicación serie	
getData()	Solicitud de los datos del sensor
SetActiveMode(onOff)	Si el parámetro tiene valor *on*, inicia el modo activo del sensor objetivo y solicita los datos continuos. Si el parámetro tiene valor *off*, detiene el modo activo.
Servicio de comunicación en serie → Servicio web	
sendData(figure, unit)	Envía los datos del sensor y su unidad
sendActiveMode(figure)	Envía la respuesta del comando SetActiveMode

Cada sensor tiene una página web individual para controlar y mostrar señales biométricas. Por lo tanto, para el modo pasivo, cuando un usuario toca un botón en la página web para

solicitar datos o configurar modos operativos, el servicio web envía un comando al traductor correspondiente. El servicio de traducción identifica la intención de la solicitud y la traduce al formato predefinido, agrega encabezado y suma de verificación al comando, y luego envía el nuevo mensaje reestructurado al servicio de comunicación serie. Cuando la placa Arduino recibe el mensaje del servicio de comunicación serie, recupera la información de *Destino* y *Comando* para ejecutar el comando de manera efectiva (por ejemplo, leer ECG).

Después de ejecutar el comando, la placa Arduino genera un paquete de respuesta para responder a la solicitud. El servicio de comunicación serie transmite el paquete a todos los servicios de traducción mediante difusión y sólo el traductor adecuado procesará el mensaje. El servicio traductor es capaz de distinguir qué solicitud responde este paquete por medio del campo *Respuesta a*. Luego, la parte de *Datos* y la parte de *Unidades* se extraen y transmiten al servicio web de destino, haciendo referencia a la parte de *Tipo de sensor* del paquete. Finalmente, los datos se muestran en la página web.

Para ilustrar el funcionamiento en detalle, se puede tomar como ejemplo la solicitud de datos de temperatura corporal. Cuando el usuario hace clic en el botón de la página web para solicitar la Temperatura Corporal, ésta contacta con el servicio web de temperatura corporal, dicho servicio inicia la coreografía para obtener dicha temperatura. Para ello, solicita ejecutar el método *getData* del servicio traductor de temperatura corporal y el traductor comienza a construir un paquete de comando para el sensor. Establece el campo *Tipo de kit* como ESALUD, el campo *Destino* como TEMPERATURA CORPORAL, el campo *Comando* como GETDATA, sin parámetro y usa el delimitador '|' para concatenar cada campo individual. Después de calcular la suma de verificación, se agrega como la cola del paquete quedando de la siguiente manera: “EHEALTH|TEMPERATURA CORPORAL|OBTENER DATOS||2657”. El traductor envía dicho paquete de comando al servicio de comunicación serie y éste lo hace llegar al Arduino/Sensor.

De manera similar, cuando la placa Arduino termina de ejecutar el comando solicitado, crea un paquete de datos y lo envía al servicio de comunicación serie. El paquete de datos tendría un formato similar a este: "EHEALTH|BODYTEMPERATURE|GETDATA |36.5|C|3052". Posteriormente el servicio de comunicación serie hace una difusión a los servicios de Traducción y sólo el Servicio de Traductor de Temperatura Corporal recoge el mensaje, donde se calculará nuevamente la suma de verificación de este paquete y se comparará con la original. Si hay una coincidencia, el paquete se valida, se desempaqueta la información y se reenvía al servicio web, que, a su vez, lo hace llegar a la web para que pueda ser mostrado.

Para el modo activo, el servicio web debe enviar un comando *setActiveMode* con el parámetro *on*. Una vez que se ha configurado el modo activo, la placa Arduino enviará los datos del sensor específico a una velocidad predefinida. Este comportamiento no se detendrá hasta que se active otro comando *setActiveMode off*. Cuando se trata de

sensores, como ECG, Airflow y GSR, el modo activo puede ser extremadamente importante. Estos datos continuos se recopilan y utilizan para generar una señal biométrica para la monitorización en tiempo real, mostrando dichos datos en una gráfica continua.

5.2.5.1. Servicios de trazabilidad

Poder realizar un seguimiento del intercambio de mensajes en el motor de coreografía es una funcionalidad muy valiosa, pues permite realizar una traza de la coreografía ejecutada para depurar posibles errores o para realizar una evaluación del sistema, permitiendo obtener indicadores de tiempo de respuesta.

El coreógrafo CHOREMED incluye dos servicios que permiten llevar acabo esta trazabilidad de una forma fácil. Tal como ya se introdujo en los apartados 3.2.2 y 4.3.4, CHOREMED proporciona un servicio que permite registrar en disco los mensajes de coreografía (existe la posibilidad de filtrar por destinatario) y de otro que permite visualizarlos en tiempo real en una interfaz gráfica que ayuda a clasificarlos (Figura 17).

Cada mensaje que se guarda en el fichero tiene los siguientes campos: *<marca de tiempo>, <servicio_remitente>, <servicio_receptor>, <mensaje_en_JSON>*

Para completar el mecanismo de trazabilidad mencionado y medir todos los tiempos desde la interfaz gráfica, todas las interacciones entre la página web de cliente y el servicio web se guardaron en disco en un fichero empleando un formato compatible al anterior: *<marca de tiempo>, <sección_web>, <servicio_web>, <texto_solicitud>*

5.2.5.2. Alojamiento y servidor web

Tal como ya se ha explicado, con el objetivo de poder ofrecer el acceso remoto a la base de control, se desarrolló una interfaz web de cliente para permitir la gestión de los sensores y solicitud de datos. En el desarrollo de dicha interfaz se empleó HTML5, CSS y JavaScript para obtener una interfaz rica e interactiva.

Al mismo tiempo, para que la base fuese autosuficiente, tal como se indicaba en los requisitos, se desarrolló un servicio sobre el motor de coreografía CHOREMED que permitiese alojar contenido web. De esta forma, la base se encarga de servir la web de cliente, sin necesidad de alojarla en servidores externos y pudiendo configurar cada cliente a medida de los sensores de que cada base dispone.

Por último, para lograr la conectividad entre la web de cliente y el coreógrafo, se creó un servicio web basado en la transferencia de estado representacional (Representational

State Transfer, REST), dicho servicio es capaz de recoger las peticiones de la web y enviarlas al servicio de traducción de un sensor concreto. En el modelo REST, cada dirección representa un recurso, es decir, en nuestro caso un sensor, por lo que se crean tantas instancias de este servicio como sensores/traductores existen en el sistema. El cliente web accederá a diferentes direcciones dependiendo del sensor que quiera utilizar.

Para la creación del servicio web basado en REST en realidad se crearon dos versiones del mismo servicio, uno para la versión de la aplicación de coreografía utilizada en el PC y otra para la versión utilizada en la Raspberry Pi. Para el desarrollo en ordenador se utilizaron las librerías de Windows Communication Foundations (WCF) que proporciona .NET v4.5. Dichas librerías no están disponibles para Windows 10 IoT, así que, en este caso, se emplearon las librerías RESTUP [127]. Estudios previos ya mostraron buenos resultados al implementar esta arquitectura REST basada en Apache Server [128].

En la Figura 30 se puede ver cómo, desde el cliente web proporcionado por la aplicación de coreografía, se accede al servicio web RESTful (empleando diferentes versiones según el tipo de librería compatible en cada dispositivo). Una vez se ha conectado con el servicio web se inicia la coreografía de procesos adecuada en cada caso para interactuar con los sensores.

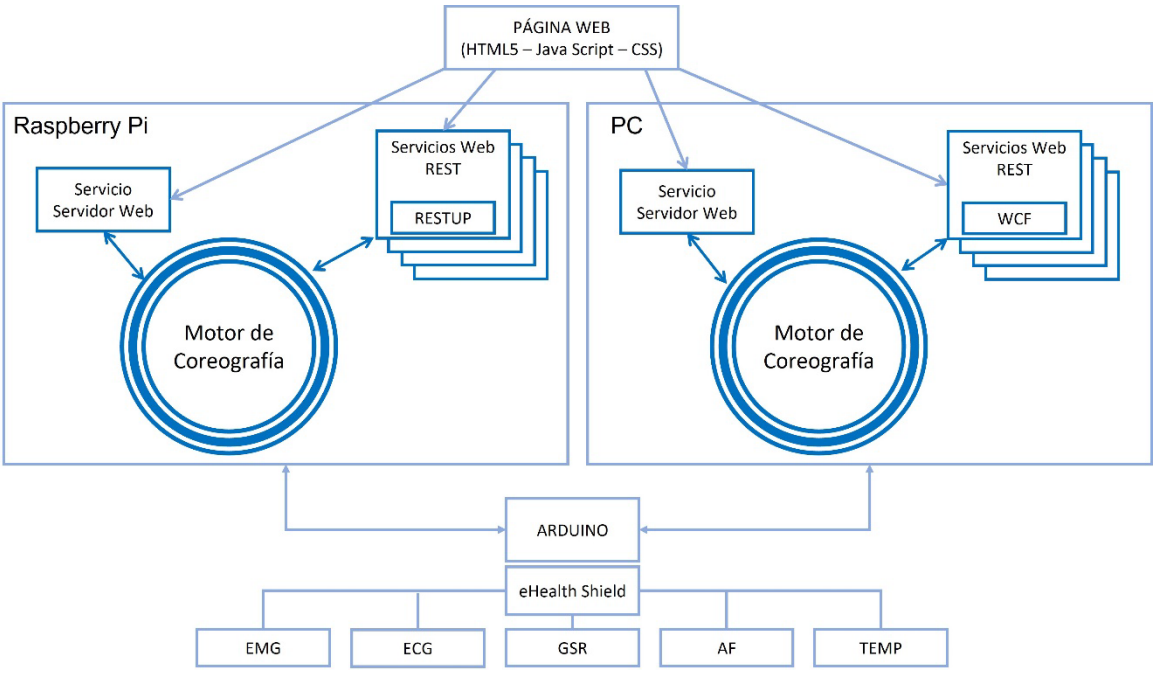

Figura 30: Servicio de alojamiento de la página web cliente y servicio web RESTful.

5.2.6. Diseño de los experimentos

Con el objetivo de evaluar la arquitectura en los dos entornos propuestos, se desarrolló dicha arquitectura de coreografía con el mismo motor CHOREMED y se emplearon los mismos servicios siempre que ha sido posible, salvo en el caso del servicio web (por problemas de compatibilidad de librerías, como se ha explicado en el apartado anterior).

Se instaló el sistema desarrollado para cada entorno en dos equipos diferentes:

- Dispositivo Raspberry Pi 3 con sistema operativo Windows 10 IoT Core. Procesador ARMv8 a 1,2 GHz y 1 GB de RAM.
- Ordenador de escritorio con Sistema Operativo Windows 10. Procesador Dual Core a 2,6 GHz y 4 GB de RAM.

Los experimentos evaluaron el retraso de la comunicación desde la solicitud de servicio hasta la respuesta del servicio para todos los sensores integrados. El principal indicador clave de rendimiento es la latencia, que se define como la diferencia de tiempo entre el inicio de la transmisión del primer mensaje y el final de la correcta recepción del último mensaje [113]. La latencia se midió y comparó en los dos entornos usando el mismo mecanismo de traza descrito en el apartado 5.2.5.1, que proporciona el registro de seguimiento de los servicios ejecutados, marca de tiempo e información de auditoría.

Debido a la distribución sesgada de los parámetros de latencia, una prueba de rango con signo de Wilcoxon al 95 % de I.C. se utilizó para evaluar la independencia de las diferencias intra e inter-entorno. Se asumió significación para $p < 0,05$. El análisis estadístico y gráfico se realizó utilizando la versión Matlab 2016R utilizando Licencia Académica.

5.3. Experimentos y resultados

El motor de coreografía utilizado, CHOREMED, permite su uso como aplicación en segundo plano o como parte de una aplicación gráfica. En el caso de querer interfaz gráfica, se puede integrar en una interfaz propia o utilizar las funcionalidades que CHOREMED proporciona para generar fácilmente una interfaz basada en .NET Windows Presentation Foundation [129]. En el caso de usar estas funcionalidades, al iniciar el motor, éste crea una interfaz basada en pestañas y busca servicios que tengan parte gráfica, colocando cada uno de ellos en una pestaña. CHOREMED proporciona clases de las que heredar e interfaces de programación para facilitar la creación de servicios con parte gráfica.

En este caso, para una mayor comodidad, se ha creado un conjunto de servicios gráficos, que permite la configuración inicial y detección del puerto de comunicación serie con el Arduino, así como probar cada uno de los sensores localmente. Además, el motor ejecuta

cada uno de los servicios no gráficos descritos en los apartados anteriores y que interactúan entre sí (mediante el paso de mensajes), creando las coreografías que ofrecen las funcionalidades de los sensores.

5.3.1. Sistema final

Tras el desarrollo del sistema y montaje de los dispositivos, la Figura 31 ilustra el sistema final desplegado sobre una Raspberry Pi. El despliegue sobre el ordenador es idéntico, pero empleando el ordenador como base. La imagen muestra, en primer lugar, el Arduino Uno montado con el shield de e-Health y los sensores de salud. El Arduino se conecta a través de una interfaz USB a la Raspberry Pi o al PC, en el que se instala una instancia del software de coreografía.

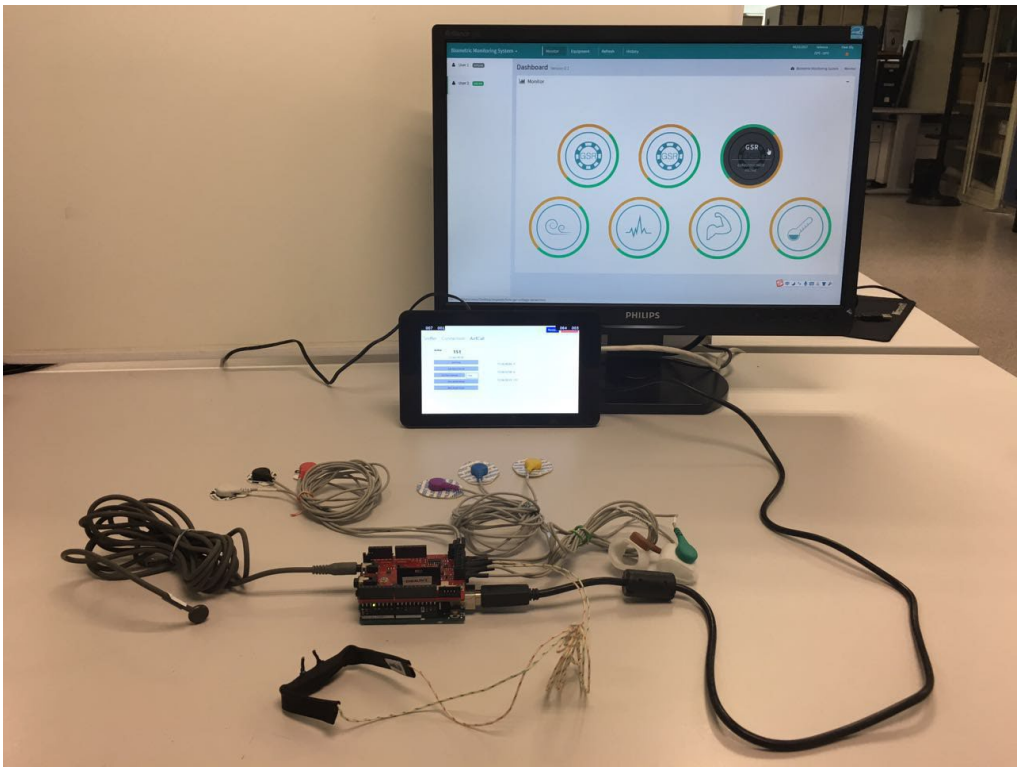

Figura 31: Despliegue del sensor

La pantalla del ordenador de la Figura 31 está conectada a un ordenador externo y muestra un navegador a pantalla completa con la página de bienvenida de la aplicación web de cliente. Dicha página, como se puede ver en la Figura 32, muestra el acceso para analizar los datos de los cinco sensores (los tres botones superiores son para las tres variables del sensor GSR). Después de hacer clic en cada botón, el usuario puede acceder a las páginas individuales de cada sensor, donde se pueden ejecutar diferentes operaciones.

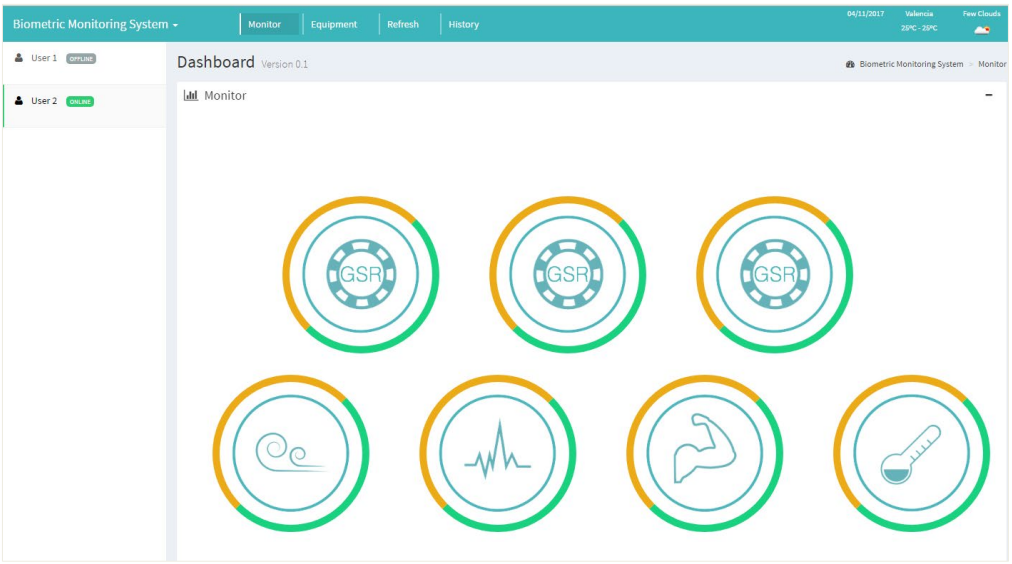

Figura 32: Pantalla de bienvenida del cliente web.

Para sensores como ECG, EMG, GSR y sensor de flujo de aire, visualizar un único dato no es de gran utilidad, por lo que los usuarios disponen de controles para iniciar el modo activo, que permite monitorizar y graficar los datos del sensor en una gráfica que evoluciona en tiempo real (tal como se muestra en la Figura 34). Además, también es posible que los usuarios establezcan el intervalo de tiempo utilizado para recuperar valores desde los sensores. Estos gráficos se actualizan en tiempo real (se pueden consultar los retrasos de tiempo en la Figura 34 y Figura 35) y se actualizan cada 250 ms. Para el sensor de temperatura corporal, además de las funciones anteriores, los usuarios pueden solicitar puntualmente el dato actual de temperatura corporal haciendo clic en el botón *GETDATA*.

Figura 33: Pantalla de gestión y captura del sensor de ECG

Por último, la Raspberry Pi (debajo de la pantalla en la Figura 31) muestra la Interfaz Gráfica de control local del software de coreografía, con una pestaña de configuración y otra por cada uno de los cinco sensores que permite administrarlos. Al hacerlo, los usuarios pueden interactuar con los sensores localmente sin necesidad de acceder a la página web.

5.3.2. Experimentos

Para los experimentos, los sensores que admiten el modo activo (ECG, EMG, Airflow y GSR) tenían definido en el sistema una frecuencia de muestreo predeterminada establecida en 20 Hz (20 muestras por segundo); esto significa que el intervalo entre dos muestras es de 50 ms tanto para PC como para Raspberry Pi. Además, la página web estaba configurada para solicitar datos cada 250 ms. Los resultados del experimento muestran que el despliegue de PC generó menos latencia en el segmento de comunicación entre el Arduino y el Coreógrafo (Figura 34). Las medidas corresponden al período de muestreo teórico, que es de 50 ms. Sin embargo, no se observa este comportamiento en la Raspberry Pi: el valor medio del tiempo de retardo es de alrededor de 50 ms, pero la desviación estándar es mayor que la desviación estándar del PC. Estos puntos están distribuidos de forma dispersa, pero muestran el patrón de una serie aritmética. El intervalo entre dos puntos correspondientes es de aproximadamente 10 ms a 20 ms.

La comunicación del modo activo entre el coreógrafo y la página web (Figura 35) muestran una situación similar. El PC muestra un buen resultado con todas las medidas concentradas alrededor de 250 ms. Se pude ver un patrón diferente en la Raspberry Pi, con todas las mediciones que usan un intervalo fijo (el valor medio es de 250 ms y el intervalo es de aproximadamente 10 ms a 20 ms).

Se realizó una comparación del segmento de comunicación entre Arduino y Coreógrafo para las dos implementaciones (Raspberry y PC). Este segmento está compuesto por el Servicio de Comunicación Serie y el Servicio de Traductor. Los resultados muestran que la Raspberry Pi tiene un mayor retraso estadísticamente significativo ($p << 0.01$) para los cinco sensores, mientras que, para el segmento entre el coreógrafo y el cliente web, ninguno de los sensores, excepto el ECG, mostraron un aumento del retraso que fuese estadísticamente significativo (para el ECG $p << 0.01$, y $p > 0.5$ para el resto).

La Figura 34 y la Figura 35 muestran los diagramas de dispersión acumulativos para los retrasos medidos con el componente de trazabilidad (Figura 28). Aunque los resultados son dispersos, se puede ver que el retardo experimentado por la Raspberry Pi muestra un patrón claro (especialmente para ECG, EMG y GSR en la Figura 35). Este patrón puede ser causado por el retraso interno del dispositivo al adquirir las medidas del Arduino (10ms-20ms), y la razón de no tenerlo en el PC, seguramente, depende del buffer de comunicación, que es más grande y rápido en el PC. En este caso, el Coreógrafo puede llenar el buffer de memoria con más mediciones, que, al enviarse en bloque, llegan más rápido a la interfaz web.

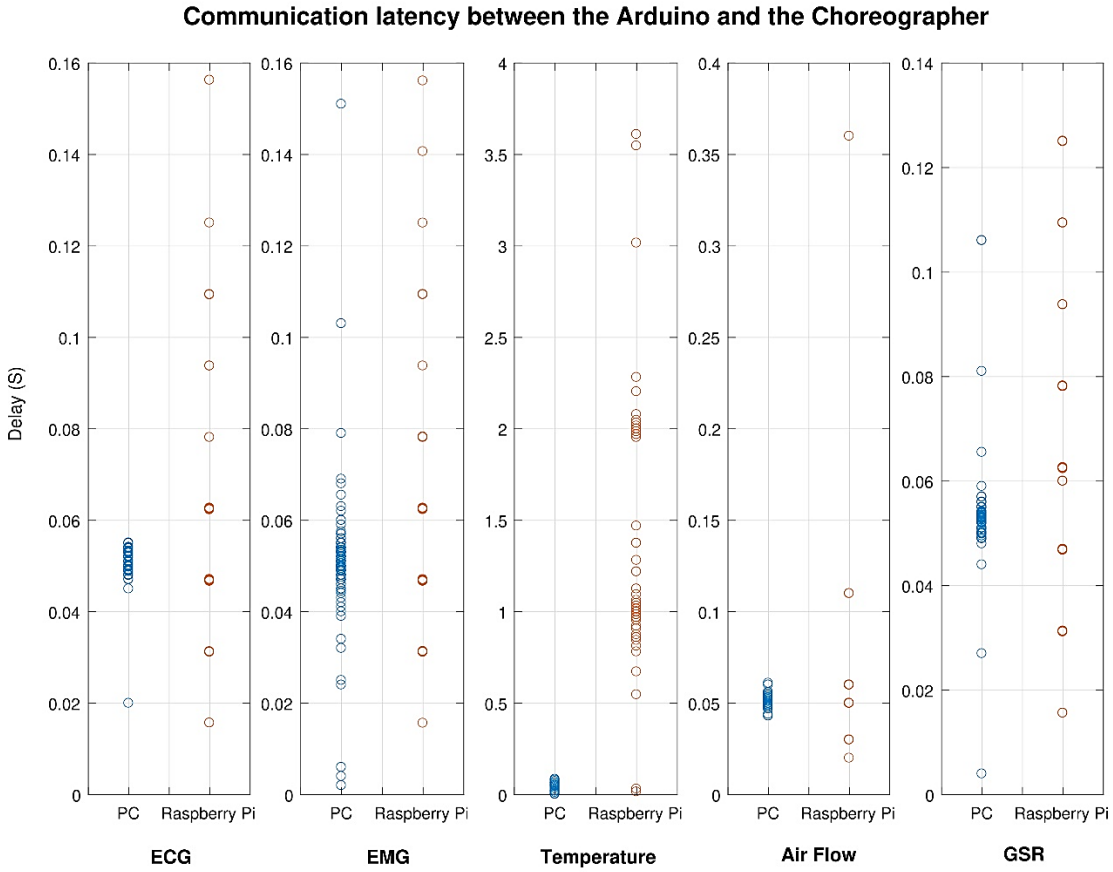

Figura 34: Comparación del retraso en las comunicaciones para el sistema desplegado en un ordenador de escritorio y una Raspberry Pi para el segmento entre el Arduino y el Coreógrafo para el modo de comunicación activo.

Otro de los hallazgos encontrados está relacionado con la precisión de los intervalos de tiempo. El PC muestra una mayor confiabilidad en los intervalos con una desviación estándar más baja y un rango más bajo. Como ejemplo, para la comunicación EMG entre el Arduino y el Coreógrafo (Figura 34), el PC tiene un retraso de 0,0512 ± 0,0035 s y un rango de 0,1490 s (N = 300), mientras que la Raspberry Pi tiene un retraso de 0,0515 ± 0,0088 s y un rango de 0,1404 s (N = 300). Por lo tanto, el Coreógrafo logra una mayor confiabilidad para adquirir medidas si se implementa en un PC.

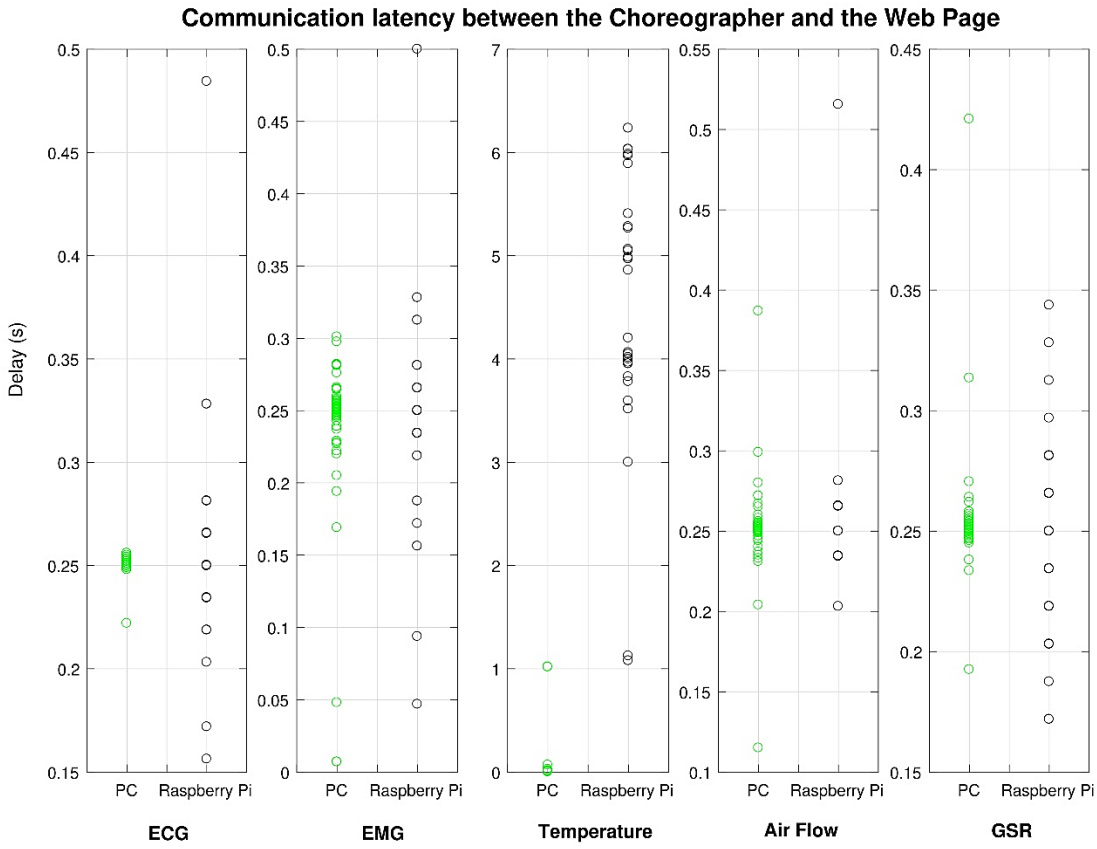

Figura 35: Comparación del retraso en las comunicaciones para el sistema desplegado en una computadora de escritorio y una Raspberry Pi para el segmento entre el Coreógrafo y la página web cliente con el modo de comunicación activo.

5.4. Discusión

En este capítulo, se ha presentado una solución basada en el paradigma de coreografía para integrar elementos de dos tendencias innovadoras: sensores aplicados a la eSalud y el campo del IoT. Estas dos tendencias están ampliamente aceptadas por la comunidad científica y se vislumbran como una solución a futuro de cómo las tecnologías de la información y la comunicación pueden hacer que los sistemas de atención médica sean sostenibles. Además, la conexión de estas dos tendencias con los nuevos modelos de inteligencia artificial y computación ubicua pueden promover el paso crucial para la adopción y el uso generalizado de sensores de salud para el manejo de enfermedades crónicas.

En este contexto, merecen especial atención las previsiones sobre el incremento de la población mayor de 60 años en los países desarrollados y la dimensión pandémica que están alcanzando algunas enfermedades. Los sistemas de atención médica no están

preparados para sostener estas cifras. Además, hay que sumar a eso la demanda creciente de mejores servicios de atención y más personalizados. Todo esto refuerza la necesidad de buscar soluciones económicas y escalables, que permitan a los usuarios conectarlas y usarlas sin la necesidad de comprender estándares o marcos de programación complejos.

Uno de los proyectos emblemáticos sobre la evaluación de la atención remota presentado en el apartado 5.1 (*Whole System Demonstrator Programme*) demostró resultados favorables en el manejo de pacientes con enfermedades crónicas, pero afirmaba que la tecnología aún no estaba lista para su aplicación [107].

El sistema propuesto y evaluado en este estudio se basa en una arquitectura de coreografía de procesos, utilizando el motor de coreografía de CHOREMED como núcleo de la aplicación desarrollada. El motor de coreografía permite interconectar mediante mensajes los diferentes servicios, los cuales han sido creados empleando las librerías que proporciona CHOREMED. El uso de la arquitectura de coreografía ha permitido construir de forma rápida y sencilla un sistema flexible y modular. Esto hace posible integrar múltiples sensores y que éstos se integren de forma transparente mediante el uso de servicios conectores reutilizables (como el servicio de comunicación serie) en combinación con los de traducción, siendo estos últimos los que concentrarían la complejidad de adaptar el protocolo del sensor al protocolo interno, que se ha creado para la plataforma y que se describe en el apartado 5.2.4. A su vez, los servicios web creados permiten la interacción (desde el exterior de la base mediante un cliente web), con los traductores de cada sensor, completando el flujo de información.

Se ha demostrado que emplear la coreografía permite la colaboración mediante mensajes de diferentes servicios, que ofrecen funcionalidades reducidas, pero que, al interactuar, permiten ofrecer funcionalidades complejas. Esto permite conectar y desconectar servicios, o sustituirlos por otros para evolucionar el sistema de una forma sencilla y eficiente. El sistema puede empezar con pocas funcionalidades e ir añadiendo más conforme se requieren o, por ejemplo, cambiar fácilmente de un tipo de sensor a otro (cambio de modelo o de fabricante). Todo esto, sin la complejidad de grandes protocolos o estándares de funcionamiento, simplemente los servicios deben cumplir el interfaz de comunicación para intercambiar mensajes XMSG, las direcciones de los servicios con los que quieren interactuar y qué funciones quieren solicitar.

Para crear el sistema presentado, se escogió como sensores de eSalud el kit de Libellium, puesto que es una solución de bajo coste que incluye una amplia gama de sensores fisiológicos. Si bien estos sensores son para prototipado, son útiles para trabajar en el diseño y evaluación de soluciones para registrar, almacenar y transmitir señales biométricas (ECG, EMG, Airflow, etc.). Se eligió esta tecnología sobre otros sensores de salud comerciales por su coste y la simplicidad de integración. Esto ha permitido centrarse en evaluar la hipótesis propuesta, en lugar de centrar todos los recursos en la obtención de sensores e implementar sus protocolos de comunicación.

Una característica deseable de este tipo de sistemas es su capacidad de portabilidad y de ser plug-and-play. Para esto, al sistema propuesto se le ha dotado de un servicio capaz de hacer las funciones de un servidor web compacto, que permite alojar y servir las páginas del cliente web para el usuario. Esto evita la necesidad de servidores externos o elementos centrales. No obstante, con la arquitectura de comunicaciones propuesta basada en coreografía, resultaría extremadamente simple incluir un servicio extra que, añadido a la coreografía existente, capturase los datos para enviarlos a un servidor en la nube para su posterior uso o procesado por algoritmos de inteligencia artificial.

Gracias a la arquitectura elegida y el coreógrafo CHOREMED ha sido posible el despliegue de este sistema, tanto en un PC, como en una Raspberry Pi, para poder realizar la comparación de ambos casos. De acuerdo con los experimentos (Figura 34 y Figura 35), existe un mayor retraso estadísticamente significativo en la Raspberry Pi con respecto al PC, lo cual es comprensible, considerando el conjunto desequilibrado de recursos computacionales. Sin embargo, para algunos casos específicos, este retraso es aceptable para el propósito médico y de seguimiento; (en el caso de ECG, la diferencia de latencia es inferior a 0,030 s). Además, están surgiendo nuevos modelos de Raspberry Pi y placas similares que en el futuro podrían equilibrar este desequilibrio.

Como ya se ha mencionado, este retraso significativamente mayor de la Raspberry Pi puede estar relacionado con las características técnicas del procesador, que tiene menos capacidades que el del PC (ARMv8 a 1,2 GHz y 1 GB de RAM frente a Dual Core a 2,6 GHz y 4 GB de RAM), así como con las capacidades físicas de comunicación del puerto USB, que es más rápido y dispone de un buffer de memoria mayor en el PC, lo que le permite un envío más rápido y estable. No obstante, los resultados muestran que la latencia experimentada de lado a lado (es decir, de los sensores a la página web), aun teniendo una diferencia significativa, es asumible para la mayoría de los usos de seguimiento médico. El bajo coste y requisitos de un sistema portable como el propuesto, son menores que el coste y requisitos de un sistema basado en un ordenador estándar. Por lo tanto, los experimentos sugieren que las nuevas arquitecturas para monitorizar bioseñales podrían basarse en la implementación que emplee una Raspberry Pi, sin comprometer excesivamente la latencia.

5.5. Conclusiones

En este capítulo, se presenta y evalúa un sistema escalable basado en coreografía de procesos, que emplea cinco sensores portables y que permite la implementación plug-and-play en diferentes escenarios de uso (en Raspberry Pi y ordenador de escritorio). Se ha observado que el dispositivo Raspberry Pi produce un retraso significativamente mayor con respecto a la misma implementación en un ordenador personal. Sin embargo, el retraso medido es aceptable para la telemonitorización en tiempo real. La

implementación de una base de eSalud como parte del Internet de las Cosas usando una Raspberry Pi tiene beneficios con respecto al uso de un ordenador de escritorio, lo que allana el camino para la implementación de nuevos sistemas portátiles para la gestión remota de condiciones crónicas. El uso de una arquitectura basada en coreografía (CHOREMED) ha permitido crear el sistema en un corto plazo de tiempo y ha ayudado con la integración de los diferentes componentes. Esto ha permitido crear un sistema robusto y fiable con pocos recursos. Además, esta arquitectura permite ampliar y escalar el sistema fácilmente (añadiendo nuevos servicios) en caso de nuevas necesidades de integración de sensores o conectividad con servidores en la nube.

6. Conclusiones generales y líneas de futuro

El principal objetivo de esta tesis ha sido demostrar que "*el uso del paradigma de coreografía de procesos permite crear e integrar aplicaciones y plataformas software heterogéneas en el ámbito de la eSalud*". Con este objetivo en mente, se han desarrollo tres sistemas de eSalud diferentes basados en este paradigma. Para ello, se ha trabajado en el desarrollo y evolución del motor CHOREMED empleado en los tres sistemas.

Un sistema de eSalud necesita ser capaz de adaptarse a la rápida aparición de los nuevos dispositivos tecnológicos que surgen en el mercado. En el capítulo 3, se ha demostrado la hipótesis secundaria HS1, que "*el uso del paradigma de coreografía de procesos permite crear sistemas de eSalud incorporando dispositivos tecnológicos complejos*". Concretamente, mediante un caso de uso basado en la tecnología Google Glass.

Para probar y validar las Google Glass, se empleó el motor de coreografía CHOREMED y se planteó una arquitectura en la que el dispositivo conectaba con un ordenador mediante dos esquemas de comunicación: Envío de mensajes por TCP y por REST. El desarrollo llevado a cabo permitió que la aplicación creada para las Google Glass se comunicase con el sistema desarrollado para el ordenador, empleando el formato de intercambio de mensajes XMSG definido en CHOREMED y los conectores TCP y REST. También se desarrollaron varios servicios para la generación de los mensajes de prueba y se utilizó el servicio de trazabilidad de mensajes para los experimentos de evaluación.

Las pruebas realizadas de evaluación mostraron que las Google Glass tienden a tener fallos de memoria (OutOfMemory) conforme el tamaño de los mensajes es mayor, lo que tiene sentido dada la capacidad limitada de memoria del dispositivo. En estas pruebas también se pudo contrastar que los mensajes REST presentan mayor sobrecarga de datos que los TCP, los cuales presentan menor latencia. Se observó que el funcionamiento del recolector de basura tenía un impacto notable sobre el funcionamiento del sistema.

Las pruebas realizadas demuestran que la principal limitación de las Google Glass es la cantidad de memoria y cómo es gestionada por el recolector de basura. Eso hace que este tipo de dispositivo sea recomendable para el uso médico, siempre que no se trate de tareas pesadas en cuanto tamaño de los datos o procesado de algoritmos complejos.

El uso de la Coreografía de Procesos y CHOREMED facilitó la rápida creación del sistema, reutilizando servicios de conexión para las comunicaciones, el protocolo de mensajería y el servicio de trazabilidad. Además, ayudó en la creación de servicios a medida específicos para la validación a realizar, que, una vez puestos en marcha, se integraron en la coreografía del proceso definido para ofrecer la funcionalidad buscada.

Una de las características fundamentales de los sistemas de eSalud es su alta heterogeneidad. Resulta, por tanto, muy importante disponer de tecnologías que permitan integrar dichos sistemas. En el capítulo 4, se ha demostrado la hipótesis secundaria HS2, que "*el uso del paradigma de coreografía de procesos ayuda a la integración de sistemas de terceros*".

En este caso, se ha desarrollado un sistema que permite integrar métodos híbridos de inteligencia artificial, accediendo directamente a la información del EHR para la detección de diabetes tipo 2 (DM2) en entornos clínicos. Los resultados obtenidos en la validación de la implantación de la arquitectura en un piloto real demuestran que es factible el enfoque propuesto.

El sistema se diseñó en tres módulos: Data Storage, Model Host y GUI Plug-in. El primero es el encargado de proporcionar el acceso a los datos de fuentes distribuidas al sistema. El segundo es el núcleo del sistema y se encarga de administrar las solicitudes de los clientes y ejecutar las escalas de riesgo. El último de los módulos aloja las interfaces de usuario y APIs de comunicación que permiten a los usuarios realizar peticiones.

Tal como muestran los indicadores de rendimiento, el sistema permitió a los clínicos acceder a los datos clínicos y ejecutar los diferentes modelos para detectar sujetos con alto riesgo de DM2 con éxito. El correcto funcionamiento del sistema presentado demostró que el paradigma de coreografía de procesos permite la integración de forma fácil y robusta de elementos complejos de terceros, como los motores matemáticos R/MATLAB o sistemas de acceso a datos I2B2 DW. Esto se realizó creando servicios que colaboran entre sí de forma distribuida para formar coreografías, que ofrecen las funcionalidades del sistema. Esto, a su vez, permitió realizar la integración de múltiples modelos para la detección temprana de DM2 utilizando técnicas de modelado híbrido.

La medicina basada en la evidencia necesita herramientas para desarrollar y probar los resultados de investigación en entornos clínicos. Se ha demostrado que los modelos de riesgo de DM2 funcionan bien en ensayos clínicos retrospectivos y prospectivos; sin embargo, aún se desconoce su rendimiento en conjuntos de datos poblacionales. El estudio ha confirmado que es posible crear un sistema distribuido basado en la coreografía de procesos para integrar técnicas de modelado heterogéneas, fuentes de datos clínicos e interfaces de usuario basadas en la web.

En los sistemas de eSalud de última generación la presencia de dispositivos del ámbito de IoT es cada vez más frecuente para el seguimiento del estado de salud de los pacientes. En el capítulo 5, se ha demostrado la hipótesis secundaria HS3, que *"el paradigma de coreografía permite el desarrollo de plataformas de eSalud orientadas al Internet de la Cosas*".

Como ejemplo de plataforma de IoT para la telemonitorización de pacientes, se ha desarrollado y evaluado un sistema escalable basado en coreografía de procesos, que emplea cinco sensores portables y que permite la implementación plug-and-play en diferentes escenarios de uso (en Raspberry y ordenador de escritorio). Se observó que el retraso del dispositivo Raspberry Pi, aunque mayor significativamente que su equivalente en el PC, tiene un valor compatible con su uso en el seguimiento de pacientes para la mayoría de los casos (siendo su coste muy inferior al de un PC). El uso de una arquitectura basada en coreografía (CHOREMED) ha permitido realizar el desarrollo en un corto periodo de tiempo y ha ayudado con la integración de los diferentes componentes. Esto ha permitido crear un sistema robusto y fiable con pocos recursos. Además, esta arquitectura permite ampliar y escalar el sistema fácilmente (añadiendo nuevos servicios) en caso de nuevas necesidades de integración de sensores o conectividad con servidores en la nube.

Como se ha podido observar a partir de los tres ejemplos descritos, cada una de las tres hipótesis secundarias han sido validadas y, por tanto, también la hipótesis principal planteada que *"el paradigma de coreografía de procesos permite crear e integrar aplicaciones y plataformas software heterogéneas en el ámbito de la eSalud"*. Su sencillez y modularidad permiten que este paradigma se adapte a distintos tipos de problemas de integración y el desarrollo de sistemas escalables.

En cuanto a las líneas de trabajo futuro de esta tesis, el objetivo principal es seguir aplicando el paradigma de coreografía de procesos en distintos escenarios y proyectos de investigación para aumentar la evidencia científica sobre su validez. Para ello, paralelamente se deberá trabajar en mejorar y ampliar en funcionalidad el coreógrafo CHOREMED. En concreto, se propone trabajar en las siguientes líneas:

- En la línea del último de los estudios presentados relacionado con IoT, y teniendo en cuenta el gran auge que está experimentando este campo en el ámbito doméstico, el objetivo es seguir trabajando con este tipo de dispositivos personales. Siendo las propuestas de trabajo:

    - Aplicación del paradigma de coreografía en un escenario de uso con asistentes personales de voz, con el objetivo de seguir validando la capacidad del paradigma para integrar dispositivos heterogéneos y estudiar la aplicación a procesos que involucran interacción directa con el usuario comparado con procesos en los que no hay mediación del usuario, como cuando se realiza la captura de datos desde sensores.
    - Otra de las propuestas es la mejora de CHOREMED para hacerlo compatible con el estándar Matter. Este estándar de la Connectivity Standards Alliance surgió en 2019 y recientemente ha lanzado su versión 1.0. Matter es un estándar de conectividad unificado para la capa de

aplicación, es de código abierto y busca permitir que los dispositivos se conecten y construyan ecosistemas seguros, aumentando la compatibilidad entre los dispositivos del hogar [130]. Actualmente, la alianza cuenta con más de 180 compañías que se han comprometido con el estándar, por lo que debería dar igual qué marca de dispositivos se compre, siempre que sea compatible con Matter. El uso de este estándar permitiría validar el paradigma de coreografía de procesos en escenarios con un gran volumen de sensores en el hogar y dispositivos wearables.

- Otra de las líneas de trabajo propuestas entronca con la vertiente de procesos de la coreografía de procesos. La integración de sistemas de minería de datos en los sistemas de eSalud es un campo en el que se están invirtiendo muchos esfuerzos, este tipo de técnicas permite obtener información de grandes volúmenes de datos que de otra forma no sería posible. Dentro de este campo *"La minería de procesos es un marco relativamente nuevo que está pensado para proporcionar información útil y comprensible para el ser humano sobre los procesos que se ejecutan en la realidad. El paradigma de la minería de procesos proporciona herramientas, algoritmos e instrumentos de visualización que permiten a los expertos humanos obtener información sobre las características de los procesos de ejecución, mediante el análisis del rastro de eventos y actividades que se produce en un determinado procedimiento, desde una perspectiva orientada a los procesos"* [131]. El uso de la coreografía de procesos puede sacar gran beneficio de este tipo de técnicas, se propone combinar la minería de procesos con la coreografía de procesos en dos vías diferentes:

    - Se propone utilizar la minería de procesos para descubrir y analizar los procesos de una organización para ayudar en el análisis de procesos necesario para una transición a un sistema basado en coreografía de procesos.
    - Se plantea el uso de los mensajes de coreografía que se intercambian los servicios, como información de entrada a la herramienta de minería de procesos, con el objetivo de mejorar los procesos y descubrir interacciones ocultas entre procesos de negocio.

7. Publicaciones realizadas

7.1. Publicaciones en revistas

1. A. Martinez-Millana, **J. L. Bayo-Monton**, A. Lizondo, C. Fernandez-Llatas and V. Traver-Salcedo, «Evaluation of Google Glass Technical Limitations on Their Integration in Medical Systems», *Sensors,* vol. 16, no. 12, 2016. **Factor de Impacto 2,677, Q1**

2. A. Martinez-Millana, **J. L. Bayo-Monton**, M. Argente-Pla, C. Fernandez-Llatas, J. F. Merino-Torres and V. Traver-Salcedo, «Integration of Distributed Services and Hybrid Models Based on Process Choreography to Predict and Detect Type 2 Diabetes», *Sensors,* vol. 18, no. 1, 2018. **Factor de Impacto 3,031, Q1**

3. **J. L. Bayo-Monton**, A. Martinez-Millana, W. Han, C. Fernandez-Llatas, Y. Sun and V. Traver-Salcedo, «Wearable Sensors Integrated with Internet of Things for Advancing eHealth Care», *Sensors,* vol. 18, no. 6, 2018. **Factor de Impacto 3,031, Q1**

7.2. Publicaciones en conferencias

1. M. Escribá-Del-Arco, **J. L. Bayo-Monton**, A. Martinez-Millana, V. Traver-Salcedo, «Análisis de las limitaciones técnicas de las Google Glass en su aplicación en soluciones médicas», *XXXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB)*, pp. 564-567, 2016
2. **J. L. Bayo-Monton**, A. Lizondo-Garcia, A. Martinez-Millana, C. Fernandez-Llatas, V. Traver-Salcedo, «Aplicación de coreografía de servicios para el acceso y gestión de sensores de monitorización de la salud», *XXXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB)*, pp. 572-575, 2016
3. S. Blanc-Clavero, **J. L. Bayo-Monton**, J. C. Campelo-Rivadulla, C. Fernandez-Llatas, «Process Choreography in Wireless Sensor and Actuator Networks: a proposal to achieve Network Virtualization", *Workshop on Innovation on Information and Communication Technologies (ITACA-WIICT 2016)*, pp. 92-104, 2016
4. C. Fernandez-Llatas, **J. L. Bayo-Monton**, A. Martinez-Romero, J. M. Benedi-Ruiz, V. Traver-Salcedo, «Interactive Pattern Recognition in Cardiovascular Diseases

Management. A Process Mining Approach», *IEEE International Conference on Biomedical and Health Informatics (BHI 2016)*, pp. 348-351, 2016

5. V. Traver-Salcedo, A. Martinez-Romero, **J. L. Bayo-Monton**, M. P. Sala-Soriano, P. Carvalho, J. Henriques, M. G. Ruano, A. Bianchi, C. Fernandez-Llatas, «Enhancing medical evidence discovery through Interactive Pattern Recognition and Process Mining», *Global Medical Engineering Physics Exchanges & Pan American Health Care Exchanges (GMEPE / PAHCE 2016)*, pp. 204, 2016
6. A. Lizondo-Gracia, **J. L. Bayo-Monton**, A. Martinez-Millana, I. Ballesteros-Barrios, W. Han, V. Traver-Salcedo, C. Fernandez-Llatas, «Choreography technologies for Interaction of Internet of Things systems», *Workshop on Innovation on Information and Communication Technologies (ITACA-WIICT 2017)*, pp. 47-52, 2017
7. S. Blanc-Clavero, **J. L. Bayo-Monton**, F. Rodriguez-Ballester, A. E. Martin-Furones, A. B. Anquela-Julian, S. Ibañez-Asensio, H. Moreno-Ramon, «GNSS Reflectometry Application in an Integrated IoT System», *Workshop on Innovation on Information and Communication Technologies (ITACA-WIICT 2017)*, pp. 36-41, 2017

Bibliografía

[1] United Nations, Department of Economic and Social Affairs, Population Division, *World Population Prospects: The 2015 Revision, Key Findings and Advance Tables,* New York, NY: United Nations, Department of Economic and Social Affairs, Population Division, 2015.

[2] E. Abrahams, G. Ginsburg y M. Silver, *The Personalized Medicine Coalition,* 2005, p. 345–355.

[3] B. Rechel, Y. Doyle, E. Grundy y M. McKee, «How can health systems respond to population ageing?,» 2009.

[4] Comisión Europea, Dirección General de Salud y Seguridad Alimentaria, «eHealth Action Plan 2012-2020,» 2012.

[5] H. Oh, C. Rizo, M. Enkin and A. Jadad, "What is eHealth (3): a systematic review of published definitions.," *Journal of medical Internet research,* vol. 7(1).e1, 2005.

[6] (WHO), World Health Organitation, "Resolution WHA58.28 of Fifty-eighth World Health Assembly," 2005.

[7] World Health Organization, «Global diffusion of eHealth: making universal health coverage achievable: report of the third global survey on eHealth,» 2016.

[8] S. Mann, «Smart Clothing: The Shift to Wearable Computing,» *Association for Computing Machinery (ACM),* vol. 39, nº 8, pp. 23-24, 1996.

[9] W. Barfield, «Fundamentals of wearable computers and augmented reality,» *CRC press,* 2015.

[10] K. Tehrani y A. Michael, «Wearable technology and wearable devices: Everything you need to know,» *Wearable Devices Magazine,* vol. 26, 2014.

[11] R. Wright y L. Keith, «Wearable technology: If the tech fits, wear it,» *Journal of Electronic Resources in Medical Libraries. Taylor & Francis,* vol. 11, nº 4, pp. 204-216, 2014.

[12] M. Bower y D. Sturman, «What are the educational affordances of wearable technologies?,» *Computers & Education. Elsevier,* vol. 88, pp. 343-353, 2015.

[13] P. Fernandez, «Wearable technology: beyond augmented reality,» *Library Hi Tech News, Emerald Group Publishing Limited,* 2014.

[14] E. Dimou, A. Manavis, E. Papachristou y P. Kyratsis, Business Models and ICT Technologies for the Fashion Supply Chain, Springer International Publishing, 2017.

[15] K. Ashton y otros, «That 'internet of things' thing,» *RFID journal,* vol. 22, nº 7, pp. 97-114, 2009.

[16] A. Guimaraes, A. Benessia y P. Curvelo, «Report EUR 26459 EN,» European Commission, Joint Research Centre, Institute for the Protection and Security of the Citizen, 2013.

[17] K. L. In Lee, «The Internet of Things (IoT): Applications, investments, and challenges for enterprises,» *Business Horizons, Science Direct,* vol. 58, nº 4, pp. 431-440, 2015.

[18] OASIS, «Reference Architecture Foundation for Service Oriented Architecture Version 1.0,» 2012. [En línea]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html. [Último acceso: 5 Noviembre 2022].

[19] The Open Group, The SOA Source Book, The Open Group.

[20] T. Erl, SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl), The Prentice Hall, 2007.

[21] E. Newcomer y G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005.

[22] C. Peltz, «Web services orchestration and choreography,» *Computer,* vol. 36, nº 10, pp. 46-52, 2003.

[23] N. M. Josuttis, SOA in practice: the art of distributed system design, O'Reilly Media, Inc., 2007.

[24] F. P. Guimaraes, E. H. Kuroda y D. M. Batista, «Performance Evaluation of Choreographies and Orchestrations with a New Simulator for Service Compositions,» *IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD),* pp. 140-144, 2012.

[25] World Wide Web Consortium (W3C), «Web Services Choreography Description Language Version 1.0,» 9 11 2005. [En línea]. Available: http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/. [Último acceso: 2 11 2022].

[26] F. T. Baker, «Chief programmer team management of production programming,» *IBM Systems journal,* vol. 11, pp. 56-73, 1972.

[27] C. Fernández-Llatas, J. B. Mocholí, A. Moyano y T. Meneu, «Semantic Process Choreography for Distributed Sensor Management,» *SSW,* pp. 32-37, 2010.

[28] O. Zimmermann, V. Doubrovski, J. Grundler y K. Hogg, «Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned,» *Association for Computing Machinery,* pp. 301-312, 2005.

[29] A. Martinez-Millana, C. Fernández-Llatas, L. Sacchi, D. Segagni, S. Guillén, R. Bellazzi y V. Traver, «From data to the decision: A software architecture to integrate predictive modelling in clinical settings,» *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* pp. 8161-8164, 2015.

[30] P. D. O'Brien y R. C. Nicol, «FIPA—towards a standard for software agents,» *BT Technology Journal,* vol. 16, nº 3, pp. 51-59, 1998.

[31] M. Gudgin, M. Hadley, N. Mendelsohn, J. J. Moreau, H. F. Nielsen, A. Karmarkar y Y. Lafon, «SOAP version 1.2 part 1: Messaging framework. W3C Recommendation,» *World Wide Web Consortium,* vol. 3, 2003.

[32] A. Martinez-Millana, J. L. Bayo-Monton, A. Lizondo, C. Fernandez-Llatas y V. Traver, «Evaluation of Google Glass Technical Limitations on Their Integration in Medical Systems,» *Sensors,* vol. 16, nº 12, 2016.

[33] A. Martinez-Millana, J. L. Bayo-Monton, M. Argente-Pla, C. Fernandez-Llatas, J. F. Merino-Torres y V. Traver-Salcedo, «Integration of Distributed Services and Hybrid Models Based on Process Choreography to Predict and Detect Type 2 Diabetes,» *Sensors,* vol. 18, nº 1, 2018.

[34] J. L. Bayo-Monton, A. Martinez-Millana, W. Han, C. Fernandez-Llatas, Y. Sun y V. Traver, «Wearable Sensors Integrated with Internet of Things for Advancing eHealth Care,» *Sensors,* vol. 18, nº 6, 2018.

[35] G. Eysenbach, «What is e-health?,» *J Med Internet Res,* vol. 3, 2001.

[36] J. Gertner, «The Truth about Google X: An Exclusive Look Behind the Secretive Lab's Closed Doors».

[37] W. Glauser, «Doctors among early adopters of Google Glass,» *Canadian Medical Association. Journal,* vol. 185, nº 16, p. 1385, 2013.

[38] Google Glass Developer Site, «Google Glass at Work,» 2016. [En línea]. Available: https://developers.google.com/glass/. [Último acceso: 5 Noviembre 2022].

[39] Google Inc., «Google Glass Enterprise Edition 2,» [En línea]. Available: https://www.google.com/glass/tech-specs/. [Último acceso: 3 Noviembre 2002].

[40] «Emergency providers see big potential for Google Glass,» *ED Manag,* vol. 26, pp. 55-58, 2014.

[41] S. Patel, H. Park, P. Bonato, L. Chan y M. Rodgers, «A review of wearable sensors and systems with application in rehabilitation,» *Journal of neuroengineering and rehabilitation,* vol. 9, nº 1, pp. 1-17, 2012.

[42] C. R. Davis y L. K. Rosenfield, «Looking at plastic surgery through Google Glass: part 1. Systematic review of Google Glass evidence and the first plastic surgical procedures,» *Plastic and reconstructive surgery,* vol. 135, nº 3, pp. 918-928, 2015.

[43] M. D. Iversen, S. Kiami, K. Singh, I. Masiello y J. von Heideken, «Prospective, randomised controlled trial to evaluate the effect of smart glasses on vestibular examination skills,» *BMJ Innovations,* vol. 2, pp. 99-105, 2016.

[44] C. A. Liebert, M. A. Zayed, O. Aalami, J. Tran y J. N. Lau, «Novel use of Google Glass for procedural wireless vital sign monitoring,» *Surgical innovation,* vol. 23, nº 4, pp. 366-373, 2016.

[45] C. Longley y D. Whitaker, «Google Glass Glare: disability glare produced by a head-mounted visual display,» *Ophthalmic and physiological optics,* vol. 36, nº 2, pp. 167-173, 2016.

[46] M. G. J. Trese, N. W. Khan, K. Branham, E. B. Conroy y S. E. Moroi, «Expansion of severely constricted visual field using Google Glass,» *Ophthalmic Surgery, Lasers and Imaging Retina,* vol. 47, nº 5, pp. 486-489, 2016.

[47] O. M. Jeroudi, G. Christakopoulos, G. Christopoulos, A. Kotsia, M. A. Kypreos, B. V. Rangan, S. Banerjee y E. S. Brilakis, «Accuracy of remote electrocardiogram

interpretation with the use of Google Glass technology,» *The American journal of cardiology,* vol. 115, nº 3, pp. 374-377, 2015.

[48] M. X. Cicero, B. Walsh, Y. Solad, T. Whitfill, G. Paesano, K. Kim, C. R. Baum y D. C. Cone, «Do you see what I see? Insights from using google glass for disaster telemedicine triage,» *Prehospital and disaster medicine,* vol. 30, nº 1, pp. 4-8, 2015.

[49] T. S. Wu, C. J. Dameff y J. L. Tully, «Ultrasound-guided central venous access using Google Glass,» *The Journal of emergency medicine,* vol. 47, nº 6, pp. 668-675, 2014.

[50] T. L. Lewis y R. S. Vohra, «Smartphones make smarter surgeons,» *Journal of British Surgery,* vol. 101, nº 4, pp. 296-297, 2014.

[51] U. V. Albrecht, U. von Jan, J. Kuebler, C. Zoeller, M. Lacher, O. J. Muensterer, M. Ettinger, M. Klintschar, L. Hagemeier y otros, «Google Glass for documentation of medical findings: evaluation in forensic medicine,» *Journal of medical Internet research,* vol. 16, nº 2, p. e3225, 2014.

[52] B. P. Waxman, «Medicine in small doses,» *ANZ J. Surg.,* vol. 82, p. 768, 2012.

[53] G. Kortuem, M. Bauer y Z. Segall, «NETMAN: the design of a collaborative wearable computer system,» *Mobile Networks and Applications,* vol. 4, nº 1, pp. 49-58, 1999.

[54] T. Roumen, S. T. Perrault y S. Zhao, «Notiring: A comparative study of notification channels for wearable interactive rings,» *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems,* pp. 2497-2500, 2015.

[55] W. R. Stevens, «TCP/IP illustrated vol. I: the protocols,» 1993.

[56] G. Zou, Y. Gan, Y. Chen, B. Zhang, R. Huang, Y. Xu y Y. Xiang, «Towards automated choreography of Web services using planning in large scale service repositories,» *Applied intelligence,* vol. 41, nº 2, pp. 383-404, 2014.

[57] A. Hasibuan, M. Mustadi, I. E. Y. Syamsuddin y I. M. A. Rosidi, «Design and implementation of modular home automation based on wireless network, REST API, and WebSocket,» *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS),* pp. 362-367, 2015.

[58] J. Tully, C. Dameff, S. Kaib y M. Moffitt, «Recording medical students' encounters with standardized patients using Google Glass: providing end-of-life clinical education,» *Academic medicine,* vol. 90, nº 3, pp. 314-316, 2015.

[59] V. Nguyen y M. Gruteser, «First experiences with GOOGLE GLASS in mobile research,» *GetMobile: Mobile Computing and Communications,* vol. 18, nº 4, pp. 44-47, 2015.

[60] C. C. Thomas y L. H. Philipson, «Update on diabetes classification,» *Medical Clinics,* vol. 99, nº 1, pp. 1-16, 2015.

[61] S. E. Kahn, R. L. Hull y K. M. Utzschneider, «Mechanisms linking obesity to insulin resistance and type 2 diabetes,» *Nature,* vol. 444, nº 7121, pp. 840-846, 2006.

[62] L. Guariguata, D. R. Whiting, I. Hambleton, J. Beagley, U. Linnenkamp y J. E. Shaw, «Global estimates of diabetes prevalence for 2013 and projections for 2035,» *Diabetes research and clinical practice,* vol. 103, nº 2, pp. 137-149, 2014.

[63] J. Beagley, L. Guariguata, C. Weil y A. A. Motala, «Global estimates of undiagnosed diabetes in adults,» *Diabetes research and clinical practice,* vol. 103, nº 2, pp. 150-160, 2014.

[64] American Diabetes Association, «Classification and diagnosis of diabetes,» *Diabetes care,* vol. 40, nº s11-s24, 2017.

[65] J. Hippisley-Cox, C. Coupland, J. Robson, A. Sheikh y P. Brindle, «Predicting risk of type 2 diabetes in England and Wales: prospective derivation and validation of QDScore,» *Bmj,* vol. 338, 2009.

[66] J. B. Meigs, P. Shrader, L. M. Sullivan, J. B. McAteer, C. S. Fox, J. Dupuis, A. K. Manning, J. C. Florez, P. W. Wilson, R. B. D'Agostino Sr. y otros, «Genotype score in addition to common risk factors for prediction of type 2 diabetes,» *New England Journal of Medicine,* vol. 359, nº 21, pp. 2208-2219, 2008.

[67] C. L. Gillies, K. R. Abrams, P. C. Lambert, N. J. Cooper, A. J. Sutton, R. T. Hsu y K. Khunti, «Pharmacological and lifestyle interventions to prevent or delay type 2 diabetes in people with impaired glucose tolerance: systematic review and meta-analysis,» *Bmj,* vol. 334, nº 7588, p. 299, 2007.

[68] D. Noble, R. Mathur, T. Dent, C. Meads y T. Greenhalgh, «Risk models and scores for type 2 diabetes: systematic review,» *Bmj,* vol. 343, 2011.

[69] K. G. M. Moons, D. G. Altman, J. B. Reitsma, J. P. A. Ioannidis, P. Macaskill, E. W. Steyerberg, A. J. Vickers, D. F. Ransohoff y G. S. Collins, «Transparent Reporting of a multivariable prediction model for Individual Prognosis or Diagnosis (TRIPOD): explanation and elaboration,» *Annals of internal medicine,* vol. 162, nº 1, p. 55, 2015.

[70] E. W. Steyerberg, K. G. Moons, D. A. van der Windt, J. A. Hayden, P. Perel, S. Schroter, R. D. Riley, H. Hemingway, D. G. Altman y P. Group, «Prognosis Research Strategy (PROGRESS) 3: prognostic model research,» *PLoS medicine,* vol. 10, nº 2, p. e1001381, 2013.

[71] G. S. Collins y K. G. Moons, «Comparing risk prediction models,» *Bmj,* vol. 344, 2012.

[72] R. D. Riley, J. Ensor, K. I. E. Snell, T. P. A. Debray, D. G. Altman, K. G. M. Moons y G. S. Collins, «External validation of clinical prediction models using big datasets from e-health records or IPD meta-analysis: opportunities and challenges,» *bmj,* vol. 353, 2016.

[73] B. M. Reilly y A. T. Evans, «Translating clinical research into clinical practice: impact of using prediction rules to make decisions,» *Annals of internal medicine,* vol. 144, nº 3, pp. 201-209, 2006.

[74] D. G. Altman, Y. Vergouwe, P. Royston y K. G. Moons, «Prognosis and prognostic research: validating a prognostic model,» *Bmj,* vol. 338, 2009.

[75] K. G. M. Moons, P. Royston, Y. Vergouwe, D. E. Grobbee y D. G. Altman, «Prognosis and prognostic research: what, why, and how?,» *Bmj,* vol. 338, 2009.

[76] E. W. Steyerberg, A. J. Vickers, N. R. Cook, T. Gerds, M. Gonen, N. Obuchowski, M. J. Pencina y M. W. Kattan, «Assessing the performance of prediction models: a framework for some traditional and novel measures,» *Epidemiology (Cambridge, Mass.),* vol. 21, nº 1, p. 128, 2010.

[77] R. Raina, Y. Shen, A. Mccallum y A. Ng, «Classification with hybrid generative/discriminative models,» *Advances in neural information processing systems,* vol. 16, pp. 545-552, 2003.

[78] E. Kayacan, B. Ulutas y O. Kaynak, «Grey system theory-based models in time series prediction,» *Expert systems with applications,* vol. 37, nº 2, pp. 1784-1789, 2010.

[79] M. I. Schmidt, B. B. Duncan, H. Bang, J. S. Pankow, C. M. Ballantyne, S. H. Golden, A. R. Folsom, L. E. Chambless y A. R. i. C. Investigators, «Identifying individuals at high risk for diabetes: The Atherosclerosis Risk in Communities study,» *Diabetes care,* vol. 28, nº 8, pp. 2013-2018, 2005.

[80] P. J. Talmud, A. D. Hingorani, J. A. Cooper, M. G. Marmot, E. J. Brunner, M. Kumari, M. Kivimäki y S. E. Humphries, «Utility of genetic and non-genetic risk factors in prediction of type 2 diabetes: Whitehall II prospective cohort study,» *Bmj,* vol. 340, 2010.

[81] D. L. Sackett, «Evidence-based medicine,» *Seminars in perinatology,* vol. 21, nº 1, pp. 3-5, 1997.

[82] I. Herman, «SemanticWeb,» W3C, 2011. [En línea]. Available: http://www.w3.org/2001/sw/. [Último acceso: 28 Diciembre 2017].

[83] M. Lluch-Ariet, «The MOSAIC System-Contributions to Efficient and Secure Exchange of Networked,» *Ph.D. Thesis,* 2016.

[84] R. Roset, M. Lurgi, M. Croitoru, B. Lluch-Ariet y P. Lewis, «Visual mapping tool for database,» *Proceedings of the Third Conceptual Structures Tool,* pp. 44-54, 2008.

[85] S. Murphy, S. Churchill, L. Bry, H. Chueh, S. Weiss, R. Lazarus, Q. Zeng, A. Dubey, V. Gainer, M. Mendis y otros, «Instrumenting the healthcare enterprise for discovery research in the genomic era,» *Genome Res,* vol. 19, nº 9, p. 1675–1681, 2019.

[86] A. Franzin, F. Sambo y B. Di Camillo, «bnstruct: an R package for Bayesian Network structure learning in the presence of missing data,» *Bioinformatics,* vol. 33, nº 8, p. 1250–1252, 2017.

[87] F. Sambo, B. Di Camillo, A. Franzin, A. Facchinetti, L. Hakaste, J. Kravic, G. Fico, J. Tuomilehto, L. Groop, R. Gabriel, T. Tuomi y C. Cobelli, «A Bayesian Network analysis of the probabilistic relations between risk factors in the predisposition to type 2 diabetes,» *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* pp. 2119-2122, 2015.

[88] B. Rood y M. J. Lewis, «Grid resource availability prediction-based scheduling and task replication,» *Journal of Grid Computing,* vol. 7, nº 4, pp. 479-500, 2009.

[89] S. Kianpisheh, M. Kargahi y N. M. Charkari, «Resource Availability Prediction in Distributed Systems: An Approach for Modeling Non-Stationary Transition Probabilities,» *IEEE Transactions on Parallel and Distributed Systems,* vol. 28, nº 8, pp. 2357-2372, 2017.

[90] A. Heuer, T. Kaufmann y T. Weyer, «Extending an IEEE 42010-compliant viewpoint-based engineering-framework for embedded systems to support variant management,» *Proceedings of the 4th International Embedded Systems Symposium,* p. 283–292, 2013.

[91] A. Dagliati, L. Sacchi, M. Bucalo, D. Segagni, K. Zarkogianni, A. Martinez-Millana, J. Cancela, F. Sambo, G. Fico, M. T. Meneu-Barreira y otros, «A data gathering framework to collect Type 2 diabetes patients data,» *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI),* pp. 244-247, 2014.

[92] G. M. Weber, S. N. Murphy, A. J. McMurry, D. MacFadden, D. J. Nigrin, S. Churchill y I. S. Kohane, «The Shared Health Research Information Network (SHRINE): A Prototype Federated Query Tool for Clinical Data Repositories,» *Journal of the American Medical Informatics Association,* vol. 16, nº 5, pp. 624-630, 2009.

[93] A. Martinez-Millana, G. Fico, C. Fernández-Llatas y V. Traver, «Performance assessment of a closed-loop system for diabetes management,» *Medical & biological engineering & computing,* vol. 53, nº 12, pp. 1295-1303, 2015.

[94] H. González-Vélez, M. Mier, M. Julià-Sapé, T. N. Arvanitis, J. M. García-Gómez, M. Robles, P. Lewis, S. Dasmahapatra, D. Dupplaw, A. Peet y otros, «HealthAgents: distributed multi-agent brain tumor diagnosis and prognosis,» *Applied intelligence,* vol. 30, nº 3, pp. 191-202, 2009.

[95] R. Bellazzi, «Big data and biomedical informatics: a challenging opportunity,» *Yearbook of medical informatics,* vol. 23, nº 1, pp. 8-13, 2014.

[96] G. Pryor, B. Lucey, S. Maddipatla, C. McClanahan, J. Melonakos, V. Venugopalakrishnan, K. Patel, P. Yalamanchili y J. Malcolm, «High-level GPU computing with Jacket for MATLAB and C/C++,» *Proceedings of the Modeling and Simulation for Defense Systems and Applications VI,* pp. 35-40, 2011.

[97] F. Wortmann y K. Flüchter, «Internet of things,» *Business & Information Systems Engineering,* vol. 57, nº 3, pp. 221-224, 2015.

[98] A. Whitmore, A. Agarwal y L. Da Xu, «The Internet of Things-A survey of topics and trends,» *Information systems frontiers,* vol. 17, nº 2, pp. 261-274, 2015.

[99] S. Warren, «Beyond telemedicine: Infrastructures for intelligent home care technology,» *Pre-ICADI Workshop on Technology for Aging, Disability, and Independence,* 2003.

[100] R. S. Weinstein, A. M. Lopez, B. A. Joseph, K. A. Erps, M. Holcomb, G. P. Barker y E. A. Krupinski, «Telemedicine, telehealth, and mobile health applications that work: opportunities and barriers,» *The American journal of medicine,* vol. 127, nº 3, pp. 183-187, 2014.

[101] C. H. Lin, S. T. Young y T. S. Kuo, «A remote data access architecture for home-monitoring health-care applications,» *Medical engineering & physics,* vol. 29, nº 2, pp. 199-204, 2007.

[102] T. Meneu, Á. Martínez-Romero, A. Martínez-Millana y S. Guillén, «An integrated advanced communication and coaching platform for enabling personalized management of chronic cardiovascular diseases,» *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* pp. 1563-1566, 2011.

[103] M. S. Patel, D. A. Asch y K. G. Volpp, «Wearable devices as facilitators, not drivers, of health behavior change,» *Jama,* vol. 313, nº 5, pp. 459-460, 2015.

[104] C. Orwat, A. Graefe y T. Faulwasser, «Towards pervasive computing in health care-A literature review,» *BMC medical informatics and decision making,* vol. 8, nº 1, pp. 1-18, 2008.

[105] H. F. Rashvand, V. Traver-Salcedo, E. Montón-Sánchez y D. Iliescu, «Ubiquitous wireless telemedicine,» *IET communications,* vol. 2, nº 2, pp. 237-254, 2008.

[106] P. Bower, M. Cartwright, S. P. Hirani, J. Barlow, J. Hendy, M. Knapp, C. Henderson, A. Rogers, C. Sanders, M. Bardsley y otros, «A comprehensive evaluation of the impact of telemonitoring in patients with long-term conditions and social care needs: protocol for the whole systems demonstrator cluster randomised trial,» *BMC health services research,* vol. 11, nº 1, pp. 1-12, 2011.

[107] M. Cartwright, S. Hirani, L. Rixon, M. Beynon, H. Doll, P. Bower, M. Bardsley, A. Steventon, M. Knapp, C. Henderson y otros, «Effect of telehealth on quality of life and psychological outcomes over 12 months (Whole Systems Demonstrator telehealth questionnaire study): nested study of patient reported outcomes in a pragmatic, cluster randomised controlled trial,» *BMJ,* vol. 346, 2013.

[108] S. van der Weegen, R. Verwey, M. Spreeuwenberg, H. Tange, T. van der Weijden, L. de Witte y otros, «The development of a mobile monitoring and feedback tool to stimulate physical activity of people with a chronic disease in primary care: a user-centered design,» *JMIR mHealth and uHealth,* vol. 1, nº 2, p. e2526, 2013.

[109] A. Fioravanti, G. Fico, M. Arredondo, D. Salvi y J. Villalar, «Integration of heterogeneous biomedical sensors into an ISO/IEEE 11073 compliant application,» *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology,* pp. 1049-1052, 2010.

[110] A. Sagahyroon, «Remote patients monitoring: Challenges,» *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC),* pp. 1-4, 2017.

[111] D. Bender y K. Sartipi, «HL7 FHIR: An Agile and RESTful approach to healthcare information exchange,» *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems,* pp. 326-331, 2013.

[112] A. S. Oh, «A study on standardized healthcare system based on HL7,» *Journal of the Korea Institute of Information and Communication Engineering,* vol. 17, nº 3, pp. 656-664, 2013.

[113] C. Gomez, J. Oller y J. Paradells, «Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology,» *Sensors,* vol. 12, nº 9, pp. 11734-11753, 2012.

[114] K. M. Chang, S. H. Liu y X. H. Wu, «A wireless sEMG recording system and its application to muscle fatigue detection,» *Sensors,* vol. 12, nº 1, pp. 489-499, 2012.

[115] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci y S. Andreescu, «Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges,» *2015 IEEE International Conference on Services Computing,* pp. 285-292, 2015.

[116] C. Fernández-Llatas, J. Mocholí, C. Sánchez, P. Sala y J. Naranjo, «Process choreography for Interaction simulation in Ambient Assisted Living environments,» *XII Mediterranean Conference on Medical and Biological Engineering and Computing,* pp. 757-760, 2010.

[117] L. Buechley, M. Eisenberg, J. Catchen y A. Crockett, «The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education,» *Proceedings of the SIGCHI conference on Human factors in computing systems,* pp. 423-432, 2008.

[118] M. Banzi y M. Shiloh, «Getting started with Arduino: the open source electronics prototyping platform,» *Maker Media,* p. 131, 2014.

[119] «Arduino Website,» 2017. [En línea]. Available: http://www.arduino.org. [Último acceso: 5 Noviembre 2022].

[120] «eHeatlh Website,» 2017. [En línea]. Available: https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical. [Último acceso: 1 Diciembre 2019].

[121] K. Saleem, H. Abbas, J. Al-Muhtadi, M. A. Orgun, R. Shankaran y G. Zhang, «Empirical Studies of ECG Multiple Fiducial-Points Based Binary Sequence Generation (MFBSG) Algorithm in E-Health Sensor Platform,» *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops),* pp. 236-240, 2016.

[122] N. Petrellis, M. K. Birbas y F. Gioulekas, «The Front End Design of a Health Monitoring System,» *HAICTA,* pp. 426-436, 2015.

[123] M. Schmidt, «Raspberry Pi,» *Pragmatic Bookshelf: Raleigh,* 2014.

[124] V. Vujović y M. Maksimović, «Raspberry Pi as a Wireless Sensor node: Performances and constraints,» *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* pp. 1013-1018, 2014.

[125] M. Sahani, C. Nanda, A. K. Sahu y B. Pattnaik, «Web-based online embedded door access control and home security system based on face recognition,» *2015*

International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], pp. 1-6, 2015.

[126] «Windows Core IoT Website,» 2017. [En línea]. Available: https://developer.microsoft.com/en-us/windows/iot. [Último acceso: 5 Noviembre 2022].

[127] «RESTUP. Webserver for Universal Windows Platform (UWP) Apps,» 2017. [En línea]. Available: https://github.com/tomkuijsten/restup. [Último acceso: 5 Noviembre 2022].

[128] V. Vujović y M. Maksimović, «Raspberry Pi as a Sensor Web node for home automation,» *Computers & Electrical Engineering,* vol. 44, pp. 153-171, 2015.

[129] «WPF Website,» 2017. [En línea]. Available: https://msdn.microsoft.com/en-us/library/ms754130.aspx. [Último acceso: 5 Noviembre 2022].

[130] Connectivity Standards Alliance, [En línea]. Available: https://csa-iot.org/. [Último acceso: 21 Diciembre 2022].

[131] C. Fernandez-Llatas, Interactive Process Mining in Healthcare, Springer Cham, 2021.

[132] C. Fernández Llatas, J. B. Mocholí, A. Moyano y T. Meneu, «Semantic Process Choreography for Distributed Sensor Management,» *SSW,* pp. 32-37, 2010.