UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e Implementación de una Capa de Seguridad para una plataforma IoT Health care interoperable

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alba Luengo, María del Carmen

Tutor: Lemus Zúñiga, Lenin Guillermo

Cotutor: Martínez Millana, Antonio

Curso 2019-2020

Resumen

Hoy en día, Internet se ha convertido imprescindible para la vida de las personas. Gracias a la evolución de Internet y de las tecnologías de la comunicación se ha conseguido crear una red de objetos físicos (cosas) que llevan sensores integrados, *software* y otras tecnologías. Estos elementos de la red tienen el fin de conectar e intercambiar datos en tiempo real con otros dispositivos y sistemas a través de Internet. Esta descripción de la red es lo que se llama Internet de las Cosas (*IoT*).

Uno de los problemas de esta tecnología es la falta de interoperabilidad y la escalabilidad comprometida en escenarios grandes, lo que limita su potencial aplicación a casos reales. Conscientes de esta problemática desde el instituto ITACA se ha diseñado un prototipo de un sistema de monitoreo de salud basado en la integración de *hardware* y *software* interoperable. El sistema involucra varios sensores para la transmisión remota de variables ambientales y fisiológicas.

En este TFG se propone el diseño e implementación de una capa de seguridad basada en las tecnologías de *Internet of Things.* Asimismo, la creación de una aplicación web *eHealth care* de manera que el sistema sea interoperable y transmita los datos de forma segura.

Palabras clave: IoT, interoperabilidad, salud, sensores, escalabilidad, cifrado y firma digital.

Resum

Hui dia, Internet s'ha convertit imprescindible per a la vida de les persones. Gràcies a l'evolució d'Internet i de les tecnologies de la comunicació s'ha aconseguit crear una xarxa d'objectes físics (coses) que porten sensors integrats, programari i altres tecnologies. Aquests elements de la xarxa tenen la fi de connectar i intercanviar dades en temps real amb altres dispositius i sistemes a través d'Internet. Aquesta descripció de la xarxa és el que es diu Internet de les Coses (*IoT*).

Un dels problemes d'aquesta tecnologia és la falta d'interoperabilitat i l'escalabilitat compromesa en escenaris grans, la qual cosa limita la seua potencial aplicació a casos reals. Conscients d'aquesta problemàtica des de l'institut ÍTACA s'ha dissenyat un prototip d'un sistema de monitoratge de salut basat en la integració de maquinari i programari interoperable. El sistema involucra diversos sensors per a la transmissió remota de variables ambientals i fisiològiques.

En aquest TFG es proposa el disseny i implementació d'una capa de seguretat basada en les tecnologies *d'Internet of Things*. Així mateix, la creació d'una aplicació web *eHealth care* de manera que el sistema siga interoperable i transmeta les dades de manera segura.

Paraules clau: IoT, interoperabilitat, salut, sensors, escalabilitat, xifrat i signatura digital.

Abstract

Nowadays, the Internet has become essential for people's lives. Thanks to the evolution of the Internet and communication technologies, it has been possible to create a network of physical objects (things) that carry integrated sensors, software and other technologies These elements of the network have the purpose of connecting and exchanging data in real time with other devices and systems through the Internet. This description of the network is what is called the Internet of Things (IoT).

One of the problems with this technology is the lack of interoperability and the scalability compromised in large scenarios, which limits its potential application to real cases. Aware of this problem, the ITACA institute has designed a prototype of a health monitoring system based on the integration of interoperable hardware and software. The system involves several sensors for the remote transmission of environmental and physiological variables.

This TFG proposes the design and implementation of a security layer based on Internet of Things technologies. Also creating a web application eHealth care so that the system is interoperable and send data securely.

Keywords: IoT, interoperability, health, sensors, scalability, encryption and digital signature.

Índice

Contenido

Índice de figuras

Índice de tablas

1. Introducción

A lo largo de este primer apartado vamos a definir la motivación, objetivos a cumplir, metodología para alcanzar los objetivos y estructura del proyecto presente.

1.1. Motivación

El Internet de las Cosas (IoT) es una de las tecnologías que está evolucionando a mayor velocidad y de forma acelerada. Uno de los ámbitos con evolución fuerte es el de la telemedicina, que es el empleo de las tecnologías de la información y las telecomunicaciones en la medicina. En este contexto se define el concepto de eSalud como la práctica de las tecnologías de la información y las telecomunicaciones en materia de salud. Gracias a esta evolución de la tecnología en IoT, se ha conseguido crear sistemas con dispositivos interoperables capaces de enviar y recibir información en tiempo real posibilitando el intercambio de información.

El motivo de realizar este TFG es mejorar esta interacción haciendo que la transmisión de datos entre el paciente y el médico sea segura y en caso de ser interceptada evitar que pueda ser utilizada para usos malignos, ya que *IoT* como tecnología es relativamente nueva y su nivel de seguridad sigue en continuo desarrollo.

Siguiendo este contexto, se ha visto la necesidad de crear una aplicación web para que los médicos puedan gestionar en tiempo real, los datos procedentes de los dispositivos conectados a sus pacientes. Además, una segunda motivación es la creación de una aplicación web para el ámbito sanitario pudiendo mejorar la monitorización diaria de adultos mayores.

1.2. Objetivos

El objetivo principal de este proyecto es crear e implementar un sistema de adquisición de datos médicos que garantice la seguridad de la información de los pacientes. Además, un requisito importante de este proyecto es crear un sistema escalable e interoperable para que pueda ser implantado en un hospital.

Para alcanzar este objetivo se han de cumplir los siguientes objetivos secundarios:

- Describir cuáles son las tecnologías detrás de las aplicaciones en un entorno médico.
- Asegurar que la información la reciben personas autenticadas y autorizadas (médicos).
- Garantizar la interoperabilidad y la escalabilidad del sistema.

1.3. Metodología

La metodología que se va a utilizar para este proyecto es la unión de las etapas del ciclo de vida del desarrollo de una aplicación web y las etapas del ciclo de vida de una capa de seguridad, la cual marcará la estructura y planificación del proyecto.

Etapas del ciclo de vida del desarrollo de una aplicación web

1. Análisis: análisis de los requisitos de la aplicación web, es decir, qué cuestiones queremos solventar.

2. Diseño: modelado de los requisitos y diseño de un cuadro de control (*dashboard*).

3. Implementación o desarrollo: desarrollo del sitio web, es decir, implementar el *dashboard* atendido a las reglas del diseño, y, que la API gestione los datos.

4. Validación: realizar pruebas con el fin de comprobar que la aplicación web funciona correctamente.

Etapas del ciclo de vida de una capa de seguridad

1. Análisis: análisis de los requisitos de la capa de seguridad, es decir, qué cuestiones queremos solventar.

2. Diseño: modelado de los requisitos.

3. Implementación o desarrollo: desarrollo de la capa de seguridad.

4. Validación: realizar pruebas con el fin de comprobar que la transmisión de datos segura funciona correctamente.

1.4. Estructura

Los objetivos y la metodología del proyecto, definidos anteriormente, indican la organización que va a seguir el documento, y, definen la primera sección. Las secciones que componen la memoria son las siguientes.

En la segunda sección, estudiaremos el contexto tecnológico de la informática médica, centrándonos en artículos relacionados en Seguridad de la Información. Gracias a la información recogida podremos comparar trabajos del mismo ámbito con el nuestro.

En la tercera sección, deberemos analizar las posibles soluciones que encontremos para realizar nuestro proyecto y explicar la solución final escogida, además definiremos un plan de trabajo y haremos una realización de un presupuesto. La cuarta sección, detalla la especificación de requisitos.

En la quinta sección, explicaremos el diseño de la solución en el que definiremos su arquitectura, las tecnologías a emplear, haciendo antes un análisis de estas, y el diseño estructural.

En la sexta sección, introducimos el desarrollo de la solución propuesta donde explicaremos las dificultades encontradas desde la solución propuesta a la solución final. En la séptima sección, describimos los pasos a seguir para implementar la aplicación web segura.

La octava sección detalla las pruebas realizadas para verificar que todo funciona correctamente. En la novena sección presenta las conclusiones obtenidas y la relación del proyecto desarrollado con los estudios cursados.

Y la décima sección describe trabajos futuros como ampliaciones o mejoras a desarrollar. Existe un glosario de términos y abreviaturas al final de la memoria que es recomendado visualizar antes de comenzar con la lectura.

2. Contexto tecnológico

El objetivo de este capítulo es explicar artículos y trabajos relacionados con nuestro proyecto. Para ello, el instituto ITACA nos ha proporcionado información sobre proyectos actuales y que realizan funcionalidades parecidas en el ámbito de eSalud. También definiremos las dos propuestas de arquitectura para la aplicación web y posteriormente compararemos y explicaremos la arquitectura seleccionada para realizar este proyecto.

2.1. Seguridad de la información en eSalud

Actualmente, la información sobre la salud personal se ha integrado en el mundo digital de manera beneficiosa y a una velocidad vertiginosa. Esto ha sido gracias a las Tecnologías de la Información y de la Comunicación (TIC) que han conseguido hacernos la vida más fácil y que todo esté conectado.

En referente a la digitalización en este ámbito existe la Historia Clínica Electrónica en España (HCE) [1] que supuso de tenerla implementada en Baleares y País Vasco en 2006 a tenerla en 14 comunidades autónomas en 2011, según el informe "*Las TIC en el Sistema Nacional de Salud» publicado por el Ministerio de Sanidad y Política Social*" [2].

Si la seguridad de la información en el sector sanitario no se implanta de manera eficiente y se hace un mal uso de ella puedo causar el robo o pérdida de datos sensibles de pacientes.

Primero, para evitar esos riesgos debemos garantizar que se cumplen los principios básicos de la seguridad de la información que son:

- Confidencialidad: sólo las personas autorizadas tendrán acceso a la información.
- Integridad: proteger la información de modificaciones de forma no autorizada.
- Disponibilidad: acceso a la información por personas autorizadas cuando éstas requieran.

Si uno de estos principios no se cumple estaremos poniendo en riesgo los datos personales de salud de cualquier paciente.

2.2. Arquitectura SOAP

SOAP puede considerarse el estándar de comunicación principal para servicios de red. Está definido por el W3C (*World Wide Web Consortium*) [3]. Se trata de un protocolo para el intercambio de información en un entorno descentralizado y disperso. Además, está basado en XML (*eXtended Markup Language*).

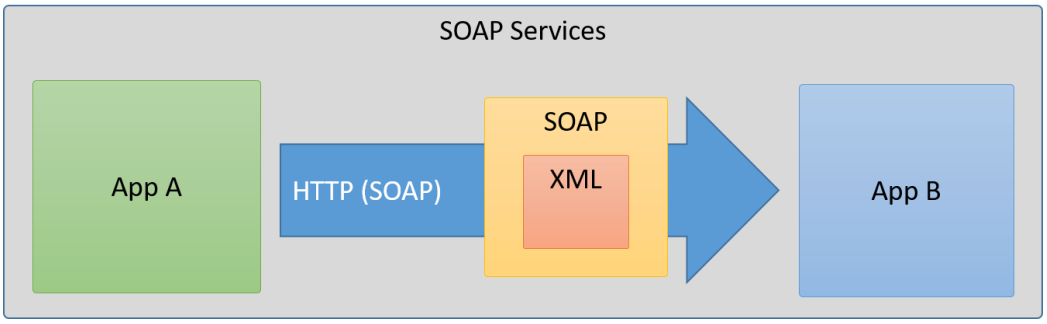

Figura 1. Comunicación SOAP.

La especificación *SOAP* ha sido ampliamente adoptada como un protocolo estándar para transportar mensajes procesados por servicios de red [4]. Los protocolos web están instalados y disponibles en todos los principales sistemas operativos.

SOAP especifica exactamente cómo codificar el encabezado HTTP y el archivo XML para que una aplicación de una computadora pueda comunicarse con la aplicación en otra computadora e intercambiar datos [5].

2.3. Arquitectura REST

Roy Fielding introdujo el término *REST* en 2000 [6], pero la primera edición de *REST* se desarrolló en 1994 mientras desarrollaba la especificación HTTP / 1.0. ***REST* es una arquitectura** cliente-servidor, en la que el cliente envía una solicitud al servidor, el servidor la procesa y devuelve una respuesta. Las solicitudes y respuestas se utilizan para enviar una representación de recursos.

El recurso está representado por un URI (Identificador Uniforme de Recursos). Se utilizan cuatro operaciones para crear, leer, actualizar y eliminar recursos: *PUT, GET,*

POST y *DELETE* [6]. Estas son las funciones estándar del protocolo HTTP (*Hypertext Transfer Protocol*).

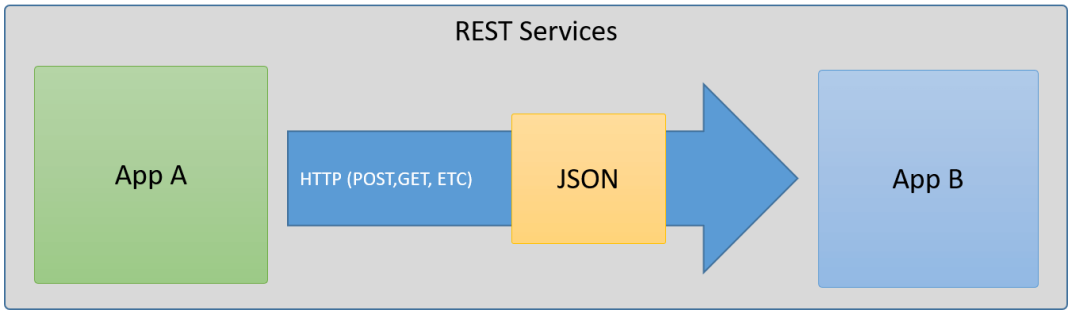

Figura 2. Comunicación REST.

La arquitectura *REST* se basa en seis reglas: arquitectura cliente-servidor, apatridia, capacidad de almacenamiento en caché, sistema en capas, interfaz uniforme y código bajo demanda. Cualquier aplicación que cumpla con los criterios anteriores en un sistema disperso tendrá características tales como rendimiento, escalabilidad, simplicidad, modularidad, portabilidad y confiabilidad.

Si el servicio no cumple ni siquiera una de las reglas enumeradas anteriormente, no puede denominarse *RESTful* [7]. Los servicios web *RESTful* son servicios web basados en la arquitectura *REST*, que se usan comúnmente para la creación de *API*s para aplicaciones basadas en la web.

2.4. Trabajos relacionados

En la actualidad, uno de los mayores problemas es implementar seguridad en dispositivos *IoT* y que, además, sean interoperables [8]. En esta sección explicaremos los trabajos que existen relacionados con eSalud e *IoT* desarrollados en colaboración con el grupo SABIEN-ITACA (Innovaciones Tecnológicas para la Salud y el Bienestar).

A. Proyecto COUCH[1]

Proyecto europeo cuya finalidad es desarrollar un grupo virtual de entrenadores -*coaches*-, que asistirá a personas mayores para fomentar el envejecimiento saludable.

[1] http://www.sabien.upv.es/project/couch/

COUCH implantará el consejo de entrenadores mediante *FIWARE* o *universAAL*, definiendo protocolos y definiciones de módulos de entrenamiento cambiables.

Vicente Traver, coordinador del grupo SABIEN-ITACA de la UPV, expresa que el equipo va a "crear una serie de avatares totalmente innovadores que podrán interactuar con los usuarios, ofreciéndoles información de gran interés para ayudarles a llevar un estilo de vida saludable. Cada entrenador, para una misma situación, ofrecerá una serie de recomendaciones diferentes, pautas y datos, pero huyendo del tradicional 'haz esto'". Y uno de los puntos importantes que declara es que la información es la clave.

La tecnología principal de este plan es la inteligencia artificial, la cual es la aportación principal del grupo SABIEN-ITACA. Además, integrarán el sistema con *universAAL* y FIWARE, como hemos nombrado antes, y, asimismo, con otros sensores y módulos de *IoT*.

B. INTER-IoT[2]

La mayoría de los desarrollos actuales de *IoT* existentes se basan en conceptos de "circuito cerrado", que se centran en un propósito específico y están aislados del resto del mundo. La integración entre elementos heterogéneos generalmente se realiza a nivel de dispositivo o red, y solo se limita a la recopilación de datos.

Un enfoque de varias capas que integra diferentes dispositivos, redes, plataformas, servicios y aplicaciones de *IoT* permitirá un continuo global de datos, infraestructuras y servicios que permitirán diferentes escenarios de *IoT*.

Además, se facilitará la reutilización y la integración de los sistemas *IoT* existentes y futuros, creando un ecosistema global de facto de plataformas *IoT* interoperables.

En ausencia de estándares globales de *IoT*, los resultados de INTER-IoT permitirán a cualquier empresa diseñar y desarrollar nuevos dispositivos o servicios de *IoT*, aprovechando el ecosistema existente y llevarlos al mercado rápidamente.

El grupo ITACA-SABIEN se encarga de implementar el piloto en el ámbito de eSalud. Del mismo modo, agregará la plataforma *universAAL IoT* dentro de INTER-IoT.

[2] http://www.sabien.upv.es/project/inter-iot/

C. InAdvance[3]

El proyecto InAdvance propone un nuevo plan de cuidados paliativos basado en la integración temprana y vías personalizadas dirigidas particularmente a personas mayores de 65 años con enfermedades crónicas complejas.

Las intervenciones resultantes de InAdvance deben ser sostenibles en el tiempo y replicables en diferentes contextos culturales y en diferentes vías de atención médica.

Al mismo tiempo, el proyecto tiene como objetivo abordar la principal preocupación europea en la reducción del impacto socioeconómico de las enfermedades crónicas.

D. Plataforma de monitorización eHealth basada en estándares FHIR

Este proyecto es el precedente del actual en el que se llevó a cabo el diseño e implementación de un sistema de monitorización basado en las tecnologías de *Internet of Things* de manera que sea interoperable con los estándares de referencia en intercambio de datos de salud (*FHIR*) en el marco del *Ambient Assisted Living* (*UniversAAL*).

Este trabajo fin de grado resulta en la integración de *hardware* y *software* para construir y testar un sistema con varios sensores para la emisión de variables ambientales y fisiológicas.

2.5. Crítica al contexto tecnológico

Podemos observar en el apartado 2.4. Trabajos relacionados cómo *IoT* ha conseguido incorporarse en proyectos de telemedicina y/o teleasistencia con presencia de la integración de *UniversAAL*, cosa que en nuestro proyecto eliminaremos y añadiremos esa capa de seguridad que no se nombra en los proyectos anteriores y es uno de los mayores problemas.

[3] http://www.sabien.upv.es/project/inadvance/

A continuación, realizaremos una comparación para ver la mejora entre el proyecto del año pasado y el nuevo:

Solución	Escalabilidad	Persistencia	Portabilidad	Eficiencia	Seguridad
Diseño con estándar *FHIR*	Media	Alta (SQLite)	Móvil	Acorde al dispositivo	Inexistente
Diseño actual	Alta	Muy alta (MongoDB)	Móvil	Acorde al dispositivo	Alta

Tabla 1. Comparación entre proyectos relacionados.

Así, podemos diferenciar y darnos cuenta de la notable mejora como es en la seguridad ya que la novedad de este proyecto reside en la implementación de una capa de seguridad aparte de la seguridad que proporciona ya el protocolo HTTP que hará uso nuestra aplicación web y la implantación de seguridad a la hora de autentificarse o ver el perfil con sus correspondientes datos de un paciente.

Otro punto a tener en cuenta son los servicios web *SOAP* y *REST* que hemos descrito en los puntos 2.2 y 2.3 a lo largo de esta sección. Para ello, hemos obtenido datos de ambas arquitecturas para compararlos mediante la web http://programmableweb.com, especializada en recopilar información sobre el uso de APIs.

La plataforma mantiene información detallada sobre los servicios, que incluyen: el nombre del servicio, la descripción, la categoría, la fecha de adición, la información sobre las tecnologías utilizadas, la información del autor y más. El presente estudio se basa en datos procedentes de los años 2005-2017.

Año	No. de servicios SOAP	No. de servicios REST
2005	18	35
2006	64	105
2007	109	216
2008	193	461
2009	259	767
2010	370	1318
2011	714	2478
2012	1614	4439
2013	1980	6117
2014	2078	7861
2015	2136	9634
2016	2234	11462
2017	2262	12012

Tabla 2. Números de servicios registrados usando servicios SOAP y REST.

Como podemos observar en 2015, el número de servicios *REST* es 5 veces mayor que los servicios *SOAP*. Esa notable diferencia a favor de *REST* garantiza un mayor soporte y resolución de problemas, y brinda a los programadores de servicios web la oportunidad de acceder a los servicios públicos que se ejecutan actualmente a través de una interfaz ampliamente conocida. Todo eso ayuda en términos del tiempo necesario para desarrollar e implementar nuevos servicios, y permite la reutilización del código existente.

2.6. Propuesta

A partir de los temas y trabajos realizados vistos con sus parecidos y diferencias a nuestro proyecto durante el apartado 2. Contexto tecnológico podemos decir que nuestro sistema se diferencia por su notable seguridad en *IoT*.

En visto a la comparación entre *SOAP* y *REST* en los apartados anteriores finalmente nos decidimos por una arquitectura *REST* para nuestra aplicación web ya que nos brinda las siguientes ventajas frente a la otra arquitectura:

- Es más eficiente en términos de rendimiento.
- Es más rápido en términos de tiempo de procesamiento de solicitudes.

- Tiene una implementación más simple mediante el uso de HTTP como protocolo de transporte por defecto.
- Es más adecuado para trabajar con el modelo de recursos de la red.

En conclusión, para nuestro sistema de telemedicina/teleasistencia se propone crear una aplicación *REST* de monitorización segura en tiempo real empleando hardware de bajo coste. De este modo se plantea construir un sistema basado en Arduino y Raspberry pi con toda la gama de capacidades que ofrecen: sensores de salud, posibilidad de procesar la información y enviarla a un servidor remoto.

La novedad de este enfoque reside en el uso de hardware de bajo coste, lo que permitirá escalar el sistema a muchos hogares sin costes excesivos y nos garantiza portabilidad y escalabilidad alta. Además, utilizamos MongoDB como base de datos permitiendo una persistencia alta. Y, por último, la implantación de una capa de seguridad que es un tema muy importante al implementar este proyecto en el ámbito sanitario.

3. Análisis del problema

A lo largo de esta sección definiremos y analizaremos los requisitos de seguridad, energéticos y de riesgos presentes en nuestro proyecto.

3.1. Análisis de seguridad

La aplicación guardará en el servidor datos sensibles tanto de los pacientes como de los médicos, por ejemplo, el usuario y contraseña de cada médico. Teniendo en cuenta que estamos almacenando información sensible deberemos de mantenerla privada. De esta manera, vamos a explicar los requisitos de seguridad que hemos aplicado.

Primero, la aplicación web se desplegará en una conexión segura HTTPS [9] (Protocolo Seguro de Transferencia de Hipertexto), que ofrece una capa de seguridad de red cifrando el mensaje a enviar y descifrando un mensaje una vez recibido, es decir, se encarga de la seguridad en la transmisión de datos.

Además, en este proyecto hemos implantado que para que el médico pueda ver los datos de un paciente deberá firmar un fichero con su clave privada. Para ello el médico deberá enviar el fichero firmado y sus claves públicas al servidor. Este último comprobará si el usuario existe, y si existe, confirmará que todos los archivos adjuntos son correctos y le permitirá ver esos datos.

En segundo lugar, la aplicación cuenta con un sistema de autentificación. Así, solo los médicos que estén dados de alta en el servidor podrán entrar al sistema y visualizar los datos introduciendo su usuario y contraseña.

Por último, la información se guarda en una base de datos MongoDB. Es una base de datos no relacional que dispone de un sistema de control de acceso basado en roles, que cada rol tiene privilegios. Esta autorización hace que los usuarios autentificados puedan acceder a ciertos recursos según sus privilegios.

3.2. Análisis energético

Actualmente, el sistema está desplegado en una Raspberry Pi ya que ofrece un consumo energético bajo y el proyecto sigue en versión piloto, que mirando al futuro se implementaría la aplicación en los ordenadores de los hospitales y el pulsioxímetro pasaría a ser una pulsera lo que conllevaría un coste energético bastante mayor.

Puesto que un ordenador convencional consume aproximadamente 1.1kW por hora, la Raspberry Pi 3B sólo consume 4W por hora. Pues el modelo actual Raspberry Pi ha llegado a ser hasta cuatro veces más potente que el primer modelo del miniordenador.

3.3. Análisis de riesgos

Riesgo	Impacto	Probabilidad	Medidas de mitigación
Avance de sensores en el mercado	Creación de sensores *IoT* que no se adapten al sistema de conexión actual.	Media	Implantar un protocolo de comunicación vía Bluetooth.
Raspberry Pi	Cada año salen nuevas versiones de este dispositivo, dejando obsoletas a las versiones anteriores.	Alta	Cambiar el sistema a la pulsera que está desarrollando el grupo SABIEN-ITACA.
Conflictos con sistemas operativos	No disponibilidad del sistema por no compatibilidad con la Raspberry Pi.	Alta	Estudiar los sistemas operativos compatibles e integrar el más conveniente.
Alcance	El proyecto puede llegar a implantarse en un mayor proyecto futuro.	Alta	Adaptar el sistema para que sea lo más escalable e interoperable posible.

Tabla 3. Análisis de riegos.

3.4. Soluciones

En los dos siguientes subapartados vamos a contemplar las ideas que tuvimos en cuenta antes de empezar a desarrollar el proyecto y la decisión final del proyecto que escogimos para llevarlo a cabo.

A. Soluciones posibles

Los puntos importantes de este proyecto y que debemos considerar en cómo hacerlos son: el diseño de la aplicación web y la capa de seguridad. Para ello, consideramos varias opciones que vamos a exponer a lo largo de este apartado.

Primero, el diseño de la aplicación. Hicimos varios bocetos para visualizar qué apartados necesitaba nuestra aplicación y cómo deberíamos de organizarlos para que fuese una interfaz fácil de usar e intuitiva por el usuario.

Por otra parte, nos documentamos sobre qué tipo de capa de seguridad implantar. De este modo, contemplamos varios modos como son encriptación simétrica o asimétrica, autentificación y firma digital.

B. Solución propuesta

En este capítulo vamos a exponer la solución final que hemos desarrollado. Como ya hemos dicho antes, consta de dos partes:

- Una aplicación web *RESTful* desde la cual los médicos sean capaces de monitorizar y visualizar los datos de los pacientes con facilidad. Consta de funciones de usuarios, donde los médicos autentificados podrán añadir, actualizar, leer y borrar (*CRUD*) a los pacientes correspondientes.
- La seguridad la podemos encontrar en el sistema de autentificación para poder entrar en la aplicación. Además, para visualizar los datos de un paciente los médicos deben firmar digitalmente un fichero y enviarlo junto con sus claves públicas. Esto último hace que se añada un punto se seguridad, permitiendo obtener un historial en el servidor de qué médico, cuándo solicitó el acceso, y, para qué lo solicito.

Por último, realizaremos unas pruebas de aceptación para comprobar el funcionamiento correcto del sistema.

3.5. Plan de trabajo

El trabajo a realizar está compuesto por tareas donde cada una de ellas tiene una fecha de entrega. El proyecto está dividido en las siguientes 7 tareas:

1. Documentación y revisión bibliográfica. (4 semanas)
2. Diseño de la aplicación web. (1 semana)
3. Diseño de la capa de seguridad. (3 semanas)
4. Desarrollo de la aplicación web. (3 semanas)
5. Implantación de la capa de seguridad. (3 semanas)
6. Testeo del sistema. (1 semana)
7. Redacción de la memoria. (14 semanas)

Tarea 1: Documentación y Revisión Bibliográfica	
Fecha Inicio	10/02/2020
Fecha Fin	02/03/2020
Descripción	En esta actividad la alumna se documenta sobre todas las tecnologías y herramientas que van a implementarse y ser utilizadas a lo largo del proyecto. Esta tarea incluye documentarse sobre *IoT, API RESTful*, interoperabilidad y cifrado y firma digital.
Duración	60h

Tabla 4. Tarea 1 - Documentación y Revisión Bibliográfica.

Tarea 2: Diseño de la aplicación web	
Fecha Inicio	03/03/2020
Fecha Fin	10/03/2020
Descripción	En esta actividad la alumna diseña el modelo y arquitectura de la aplicación web, tanto la parte visual del cliente como la parte interna del servidor, que se implementará posteriormente.
Duración	10h

Tabla 5. Tarea 2 - Diseño de la aplicación.

Tarea 3: Diseño de la capa de seguridad	
Fecha Inicio	11/03/2020
Fecha Fin	01/04/2020
Descripción	En esta actividad la alumna diseña el modelo y arquitectura de la capa de seguridad a implantar en el sistema.
Duración	35h

Tabla 6. Tarea 3 - Diseño de la capa de seguridad.

Tarea 4: Desarrollo de la aplicación web	
Fecha Inicio	02/04/2020
Fecha Fin	23/04/2020
Descripción	En esta actividad la alumna desarrolla la *API RESTful* diseñada en la tarea 2. Se hará uso de la base de datos MongoDB para almacenar los datos y del programa Postman para comprobar que los datos se envían y reciben correctamente.
Duración	65h

Tabla 7. Tarea 4 - Desarrollo de la aplicación web.

Tarea 5: Implantación de la capa de seguridad	
Fecha Inicio	24/04/2020
Fecha Fin	15/05/2020
Descripción	En esta actividad la alumna implanta la capa de seguridad en la aplicación desarrollada en la tarea anterior. Consta de una capa de autentificación y una transmisión de datos segura mediante cifrado y firma digital.
Duración	70h

Tabla 8. Tarea 5 - Implantación de la capa de seguridad

Tarea 6: Testeo del sistema	
Fecha Inicio	18/05/2020
Fecha Fin	25/05/2020
Descripción	En esta actividad la alumna realizará las pruebas de validación necesarias para el funcionamiento correcto del sistema.
Duración	15h

Tabla 9. Tarea 6 - Testeo del sistema.

Tarea 7: Redacción de la memoria	
Fecha Inicio	18/03/2020
Fecha Fin	24/06/2020
Descripción	En esta actividad la alumna realizará la redacción de la memoria del proyecto. Esta actividad se hará en paralelo con las tareas anteriores.
Duración	85h

Tabla 10. Tarea 7 - Redacción de la memoria.

Cronograma de actividades					
ACTIVIDADES	Febrero	Marzo	Abril	Mayo	Junio
Tarea 1					
Tarea 2					
Tarea 3					
Tarea 4					
Tarea 5					
Tarea 6					
Tarea 7					

Tabla 11. Cronograma de actividades.

3.6. Presupuesto

En este último apartado expondremos todo el coste del material hardware y software necesario para llevar a cabo el TFG junto a las horas-hombre empleadas. Hemos incluido el hardware que fue necesario el año pasado para la creación del sistema ya que es el que vamos a utilizar nosotros también. El presupuesto se divide en:

- Componentes: Material hardware necesario para implantar el sistema.
- Personal: Cantidad de horas para realizar el proyecto.

Presupuesto	Nombre	Precio Unitario (€)	Cantidad	Subtotal
Componentes	*Kit* Raspberry Pi 3B	120,00	1	120,00€
	Arduino UNO	20,00	1	20,00€
	Sensor de temperatura *IoT*	19,95	1	19,95€
	Pulsioxímetro *IoT*	20,70	1	20,70€
	Módulo *e-Health*	900,00	1	900,00€
	Pantalla táctil para Raspberry	72,50	1	72,50€
Personal	Horas de desarrollador	25,00	65	1625,00€
	Horas de Ingeniero Informático	90,00	275	24750,00€
	Horas del tutor	150,00	50	7500,00€
TOTAL				35028,15€

Tabla 12. Presupuesto.

4. Especificación de requisitos

A lo largo de esta sección definiremos los requisitos específicos de funcionalidad, de interfaces y de rendimiento.

4.1. Propósito

Los siguientes apartados se definen con el fin de informar el desarrollo de la aplicación por medio de la definición de las funcionalidades y las propiedades de la interfaz.

4.2. Ámbito del sistema

La aplicación web mostrará los datos de los pacientes en tiempo real de modo que los usuarios –médicos- puedan observar sus modificaciones y actuar como administradores.

El médico podrá realizar las siguientes acciones:

- Registro de pacientes.
- Modificación en los datos de los pacientes.
- Eliminar un paciente.
- Solicitar ver los datos de un paciente.
- Ver los datos de un paciente.
- Visualizar una tabla con todos los pacientes registrados.
- Observar todos los datos y gráficas disponibles en la pestaña principal.

4.3. Funciones del producto

Con el objetivo de entender el usuario del sistema y qué acciones puede realizar, hemos realizado un diagrama de casos de uso mostrado a continuación.

Figura 3. Diagrama de casos de uso del usuario médico.

En esta tabla podemos visualizar los casos de usos presentes en la Figura 3.

Casos de uso	Descripción
CU01	Autentificarse
CU02	Registrar un paciente nuevo
CU03	Visualizar todos los pacientes
CU04	Ver datos de un paciente
CU05	Eliminar un paciente
CU06	Observar datos y gráficas generales
CU07	Modificar un paciente
CU08	Enviar firma digital

Tabla 13. Casos de usos.

4.4. Características de los usuarios

En este caso, sólo hay un rol de usuario: el médico. Este actúa como administrador del sistema. Se caracteriza por tener el control del sistema ya que es el gestor de la aplicación y el cual ve los datos médicos de los pacientes registrados. Este usuario debe tener experiencia en análisis de gráficas y datos médicos para comprender el significado de estos.

4.5. Restricciones

Las restricciones son las condiciones del sistema. Estas condiciones son las siguientes:

A. Restricciones de desarrollo

- Implantación del servidor de base de datos en un ordenador.
- La estructura de los datos debe tener un formato JSON.

4.6. Supuestos y dependencias

Una modificación en la estructura de los datos necesitaría un análisis y cambio en la estructura del sistema.

4.7. Requisitos específicos

A continuación, vamos a definir y especificar los requisitos que se deben abordar en el desarrollo del sistema.

A. Interfaces externas

Este subapartado define los requisitos referidos a la interfaz de usuario.

RI01: Implantación del sistema	
Descripción	El sistema se integrará junto a la base de datos.
Prioridad	Alta.
Casos de uso	CU01, CU02, CU03, CU04, CU05, CU06 y CU07.

Tabla 14. Requisito de interfaz - Implantación del sistema.

RI02: Navegabilidad de la aplicación	
Descripción	La aplicación será de uso fácil e intuitiva para los usuarios.
Prioridad	Media.
Casos de uso	CU02, CU03, CU04, CU05, CU06 y CU07.

Tabla 15. Requisito de interfaz - Navegabilidad de la aplicación.

RI03: Administración del sistema	
Descripción	El usuario administrador podrá modificar y consultar los datos del sistema.
Prioridad	Alta.
Casos de uso	CU02, CU03, CU04, CU05, CU06 y CU07.

Tabla 16. Requisito de interfaz - Administración del sistema.

RI04: Representación de datos	
Descripción	Los datos serán presentados en gráficas por el sistema.
Prioridad	Alta.
Casos de uso	CU03, CU04 y CU06.

Tabla 17. Requisito de interfaz - Representación de datos.

B. Requisitos de funcionalidad

Este subapartado define los requisitos referidos a los servicios que ofrece el sistema.

RF01: Verificación de usuario	
Descripción	El usuario tendrá acceso al sistema si los datos introducidos se verifican con los introducidos en la base de datos.
Prioridad	Alta.
Casos de uso	CU01.

Tabla 18. Requisito de funcionalidad - Verificación de usuario.

RF02: Registro datos del paciente	
Descripción	El sistema facilitará un formulario en el que el médico deberá rellenar los datos del paciente.
Prioridad	Alta.
Casos de uso	CU02.

Tabla 19. Requisito de funcionalidad - Registro datos del paciente.

RF03: Análisis pacientes registrados	
Descripción	El sistema facilitará una visualización en forma de tabla con los datos registrados de cada paciente.
Prioridad	Media.
Casos de uso	CU03.

Tabla 20. Requisito de funcionalidad - Análisis pacientes registrados.

RF04: Análisis historial paciente	
Descripción	El sistema facilitará una visualización de los datos registrados de un paciente y, además, sus constantes fisiológicas.
Prioridad	Media.
Casos de uso	CU04 y CU08.

Tabla 21. Requisito de funcionalidad - Análisis historial paciente.

RF05: Eliminar paciente	
Descripción	El usuario administrador podrá eliminar pacientes de la base de datos.
Prioridad	Alta.
Casos de uso	CU05.

Tabla 22. Requisito de funcionalidad - Eliminar paciente.

RF06: Análisis datos hospital	
Descripción	El sistema facilitará una visualización de los médicos, pacientes y gráfica con los datos de los pacientes registrados en el hospital correspondiente.
Prioridad	Media.
Casos de uso	CU06.

Tabla 23. Requisito de funcionalidad - Análisis datos hospital.

RF07: Modificación datos paciente	
Descripción	El usuario administrador podrá modificar los datos de pacientes en la base de datos.
Prioridad	Alta.
Casos de uso	CU07.

Tabla 24. Requisito de funcionalidad - Modificación datos paciente.

RF08: Verificación firma médico	
Descripción	El sistema deberá pedir al usuario que envíe su usuario, claves públicas y un fichero firmado por este mediante sus claves cuando intente acceder a los datos de un paciente específico.
Prioridad	Alta.
Casos de uso	CU04 y CU08.

Tabla 25. Requisito de funcionalidad - Verificación firma médico.

C. Requisitos de rendimiento

RR01: Número de ordenadores	
Descripción	El sistema se implantará en un único equipo, el cual deberá estar implantada la base de datos.
Prioridad	Alta.
Casos de uso	-

Tabla 26. Requisito de rendimiento - Número de ordenadores.

RR02: Tiempo de reacción	
Descripción	El sistema visualizará y analizará los datos en menos de un minuto.
Prioridad	Alta.
Casos de uso	-

Tabla 27. Requisito de rendimiento - Tiempo de reacción.

5. Diseño de la solución

En este capítulo explicaremos tanto el diseño de nuestro sistema como los componentes que lo forman.

5.1. Arquitectura hardware del sistema

El objetivo principal del sistema es la comunicación entre los sensores y el servidor. Por ello, vamos a definir los aspectos importantes en nuestra arquitectura hardware para conseguir finalmente un sistema interoperable con los componentes del sistema.

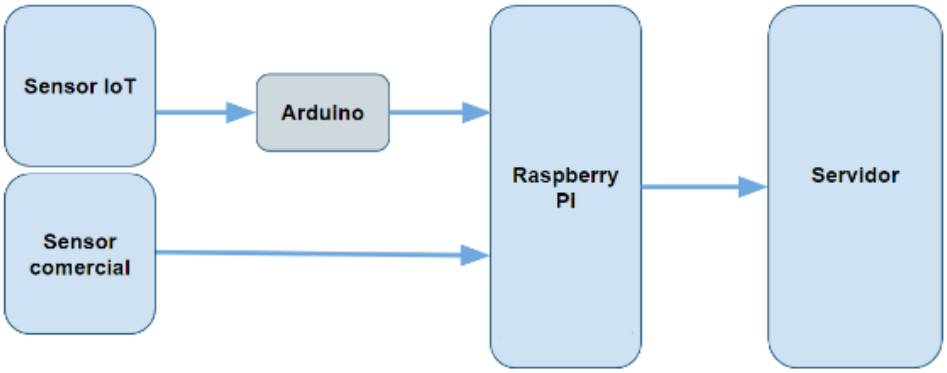

Figura 4. Esquema general de la arquitectura hardware.

La comunicación nombrada anteriormente sigue los siguientes pasos:

1. Los sensores envían los datos correspondientes.
2. El Arduino tramita los datos recibidos y los encapsula en un JSON.
3. La Raspberry Pi administra el Arduino, desencripta el mensaje, identifica y almacena en la base de datos los datos de cada sensor.
4. Los datos se envían por medio de un método POST.

5.2. Representación arquitectónica

Esta sección describe la arquitectura por medio del modelo “4+1” haciendo uso de sus visiones definidas.

Vista de casos de uso	
Audiencia	Todos los *stakeholders* del sistema (usuarios finales incluidos).
Área	Define la funcionalidad del sistema mediante el conjunto de casos de uso y/o escenarios. Detalla los actores y describe las necesidades de los usuarios, a nivel de diseño.

Tabla 28. Vista de casos de uso.

Vista lógica	
Audiencia	Diseñadores.
Área	Requisitos funcionales. Define el modelo de objeto de diseño, los casos de uso más relevantes y requerimientos del sistema.

Tabla 29. Vista lógica.

Vista de datos	
Audiencia	Gestión de la base de datos.
Área	Persistencia. Define los componentes arquitectónicos en el modelo de datos, como los flujos de datos del sistema.

Tabla 30. Vista de datos.

Vista física	
Audiencia	Administrador de despliegue.
Área	Define la topología de componentes de software en el hardware mostrando los aspectos distribuidos del sistema. Detalla las estructuras de despliegue, así como los escenarios de despliegue y sus interacciones con el sistema.

Tabla 31. Vista física.

5.3. Objetivos y condiciones arquitectónicas

En referencia a la arquitectura del sistema encontramos unos requisitos claves y condiciones que debemos destacar. Son los siguientes:

- El proyecto está diseñado con una visión de futuro donde pueda obtener la máxima interoperabilidad y escalabilidad. Por ello, los futuros arquitectos y diseñadores son los principales *stakeholders* del sistema y proyecto, que con

la ayuda de este documento podrán llevar a cabo una correcta mantenibilidad.

- El sistema se desarrollará mediante tecnologías de Microsoft .NET, usando una base de datos local (MongoDB) para la persistencia de datos. Gracias a la persistencia alta que nos ofrece esta base de datos nos permitirá aumentar la portabilidad del sistema en un futuro.
- El sistema se debe de comunicar con la *API RESTful*. De esta manera, una de las preocupaciones de esta arquitectura es la comunicación del sistema y la interacción con sistemas externos.
- Definir la comunicación entre los sensores y el sistema. Hay dos formas: los sensores actúan como un sistema activo, enviando constantemente datos (*PUSH*); o de manera pasiva, solicitando los datos cuando sean necesarios desde el sistema (*PULL*).

5.4. Diseño detallado

Nuestro proyecto está basado en un *API RESTful* que se comunica con la base de datos (MongoDB) por medio de la capa de lógica que explicaremos más en detalle posteriormente. Con esta breve introducción el esquema del sistema es el siguiente:

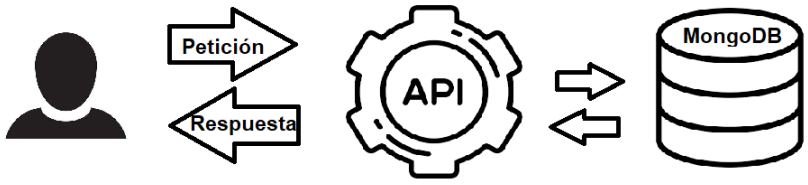

Figura 5. Esquema del sistema a desarrollar.

A. Capa de lógica

En esta sección nos centraremos en la lógica de la aplicación, donde los datos tienen un propósito. También definiremos los componentes del sistema e interfaces, las cuales se utilizan para la comunicación entre los componentes.

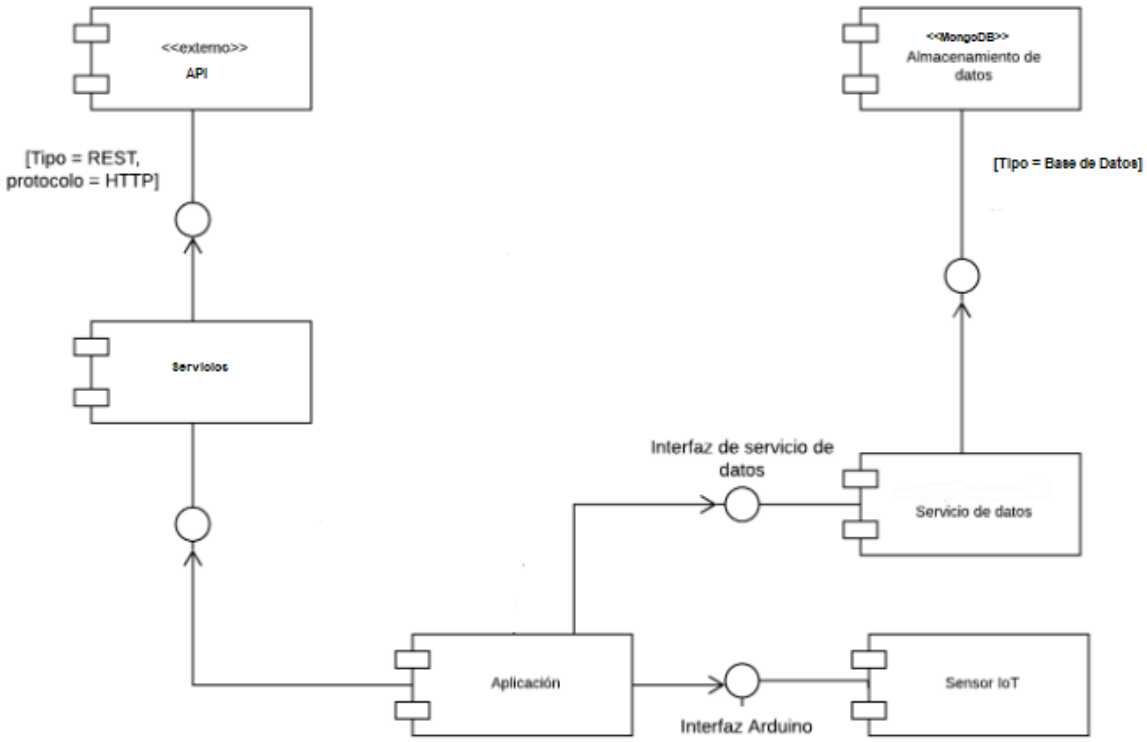

Figura 6. Diagrama de lógica de componentes.

Componente	Responsabilidad
API	• Gestiona todas las comunicaciones por medio de una *API RESTful*. • Facilita los métodos para la correcta comunicación cliente-servidor entre la aplicación y los servicios web.
Servicios	• Hace uso de los métodos *POST, GET, UPDATE* y *DELETE* para el envío de datos entre la aplicación y el servidor.
Almacén de datos	• Proporciona a la aplicación persistencia alta ya que usamos una base de datos local en MongoDB.
Servicio de datos	• Aporta las librerías para el empleo de métodos en relación con la persistencia de datos.
Sensor *IoT*	• Recoge los datos de los pacientes.

Tabla 32. Responsabilidades de los componentes lógicos.

B. Definición de interfaces

Esta sección se encuentra detallada en el proyecto antecesor a este en el apartado 4.4.2 cuyo enlace adjunto aquí.

C. Capa de acceso a datos

Esta capa se encarga de almacenar la información de los usuarios. Aunque la base de datos es NoSQL, MongoDB, podemos observar la representación de la capa en la siguiente figura.

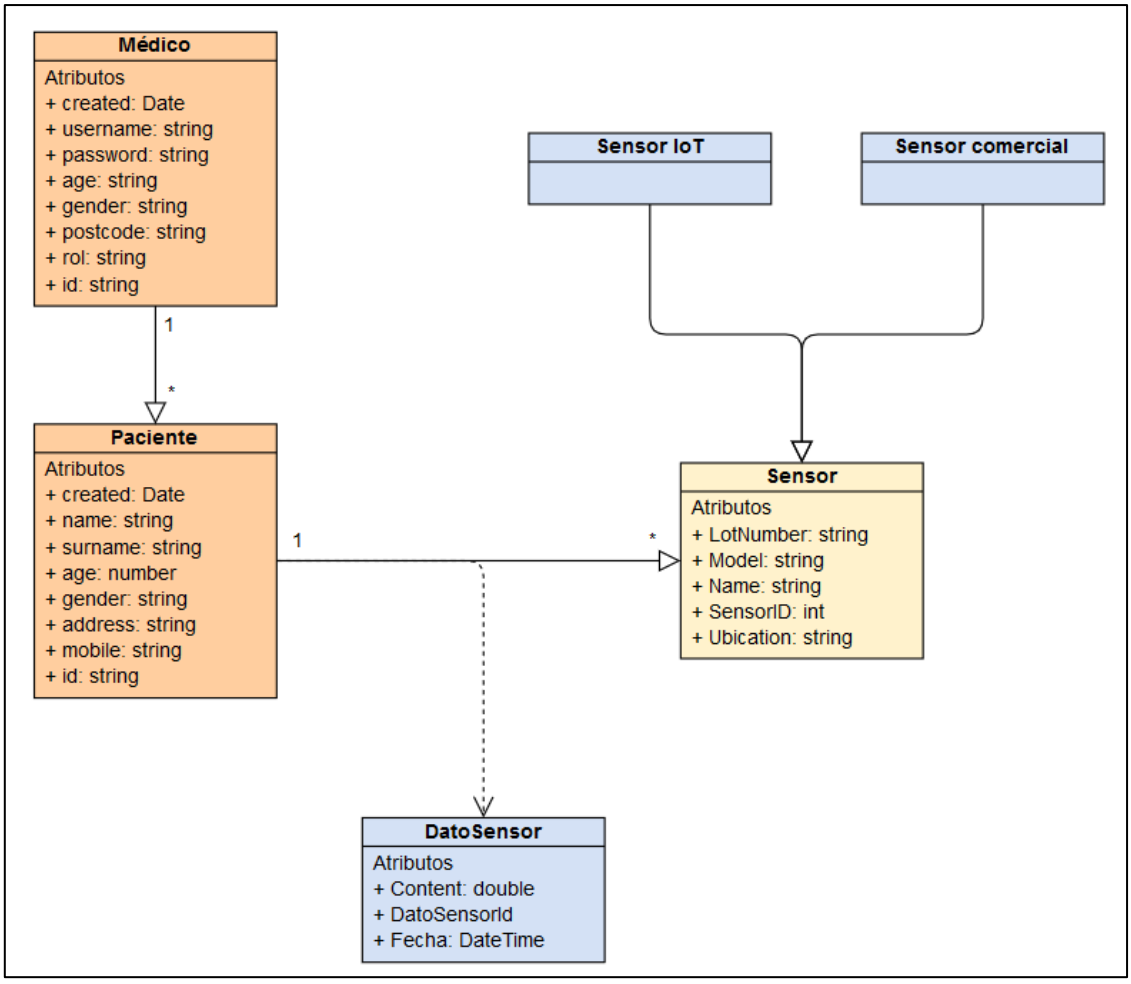

Figura 7. Estructura de clases del sistema.

- Las clases Sensor comercial e *IoT* heredan de la clase Sensor. Se diferencian en que los datos procedentes del sensor *IoT* está conectado al sistema, y, el segundo, sensor comercial, son datos introducidos por el paciente.
- DatoSensor es una clase de tipo asociación entre Paciente y Sensor.

Las nuevas clases introducidas en este proyecto son las de Médico y Paciente que se gestionan en la API que estamos desarrollando.

5.5. Componentes del sistema

En esta sección describiremos el hardware con el que se ha construido el proyecto.

A. Raspberry Pi 3B[4]

Figura 8. Raspberry Pi 3B.

Raspberry Pi es una placa computadora portable de bajo coste. Se usa ampliamente en proyectos de investigación, como el monitoreo de variables ambientales y fisiológicas en telemedicina, que es nuestro caso. Este modelo cuenta con conexión LAN inalámbrica y Bluetooth. También lleva integrado un procesador *Broadcom Quad Core* de 1.2GHz y una memoria de 1GB de RAM.

En este proyecto lleva implantada la versión de 2019 de Windows *IoT*. La API se despliega cuando el sistema operativo ya está iniciado.

[4] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

B. Arduino UNO[5]

Arduino UNO es una placa de microcontrolador basado en el Microchip ATmega328P. Integra una memoria flash de 32KB y una memoria de 2KB de SRAM. Marca el protocolo entre la Raspberry Pi y los sensores *IoT* debido a su puerto USB de tipo B.

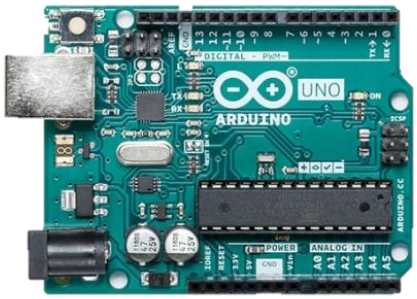

Figura 9. Arduino UNO.

Envía la información recibida de los sensores *IoT* al Arduino a la velocidad que establezcamos en el código. También envía cada segundo los datos a la Raspberry.

C. Pulsioxímetro[6]

El pulsioxímetro nos indica la cantidad de oxígeno disuelto en la sangre, es decir, la saturación de oxígeno arterial. Esto se lleva a cabo mediante impulsos fotoeléctricos no impulsivos.

Se debe colocar en una articulación con buen flujo sanguíneo, como son los dedos.

Figura 10. Pulsioxímetro eSalud.

[5] https://store.arduino.cc/arduino-uno-rev3
[6] https://www.cooking-hacks.com/pulse-and-oxygen-in-blood-sensor-spo2-ehealth-medical.html

D. Sensor de temperatura[7]

Figura 11. Sensor de temperatura corporal.

Este sensor nos permite medir la temperatura corporal. Capta de 0 a 500 grados.

Para el envío de los datos primero debemos conectarlo al módulo e-*Health* del Arduino. De esta manera, el Arduino codifica los datos, en formato JSON, procedentes del sensor y los envía a la Raspberry Pi para su monitorización y análisis.

E. Módulo e-Health[8]

Figura 12. E-Health Kit.

Gracias al módulo e-*Health* que ofrece Libellium podemos conectar al Arduino sensores médicos y biométricos, los cuales nos permiten la monitorización de pacientes en tiempo real.

Para poder hacer uso de los sensores *IoT* debemos acoplar este módulo a los puertos PIN del Arduino. Además, tiene añadida una librería con ejemplos para el uso de cada sensor integrado en el kit.

5.6. Tecnologías utilizadas

En esta sección explicaremos las diferentes herramientas y tecnologías empleadas a lo largo de nuestro proyecto.

[7] https://www.cooking-hacks.com/body-temperature-sensor-ehealth-medical.html
[8] http://www.libelium.com/libeliumworld/ehealth/

A. MongoDB

Es una base de datos documental o NoSQL que ofrece las siguientes características:

- No estructurado, es decir, almacena en ficheros JSON aportando una gran flexibilidad.
- Escalabilidad horizontal. Aporta una escalabilidad alta.
- No relacional.
- Gran eficacia en acceso a datos y su debido análisis.
- Indexación.
- Añadir datos en tiempo real.
- Fácil de usar.
- Gratuito.

Los datos a almacenar en la base de datos son sobre los pacientes y médicos de un hospital. Estos son los ya expuestos en la sección anterior.

MongoDB [10] almacena documentos BSON, es decir, una representación binaria de JSON. Como este proyecto se implantará, en un futuro, en hospitales necesitamos que la base de datos tenga una velocidad de almacenamiento alta y los datos se estructuren en documentos JSON, donde la estructura se puede modificar ya que cada documento puede tener diferentes campos. Estos dos aspectos lo cumplen el sistema. Está compuesto por colecciones de documentos.

En conclusión, como la información que debemos guardar es no relacional se ha escogido MongoDB como base de datos del sistema. Además, aporta una alta escalabilidad que es uno de los aspectos más importantes que buscamos para la realización de este proyecto.

B. API RESTful

Una *API RESTful* utiliza solicitudes HTTP mediante los métodos: *GET* para obtener un recurso, *PUT* para cambiar el estado o actualizar un recurso, *POST* para crear un recurso y *DELETE* para eliminarlo. Permite que dos programas se comuniquen entre ellos, es decir, detalla la forma correcta para que un desarrollador programe un programa solicitando servicios de una aplicación.

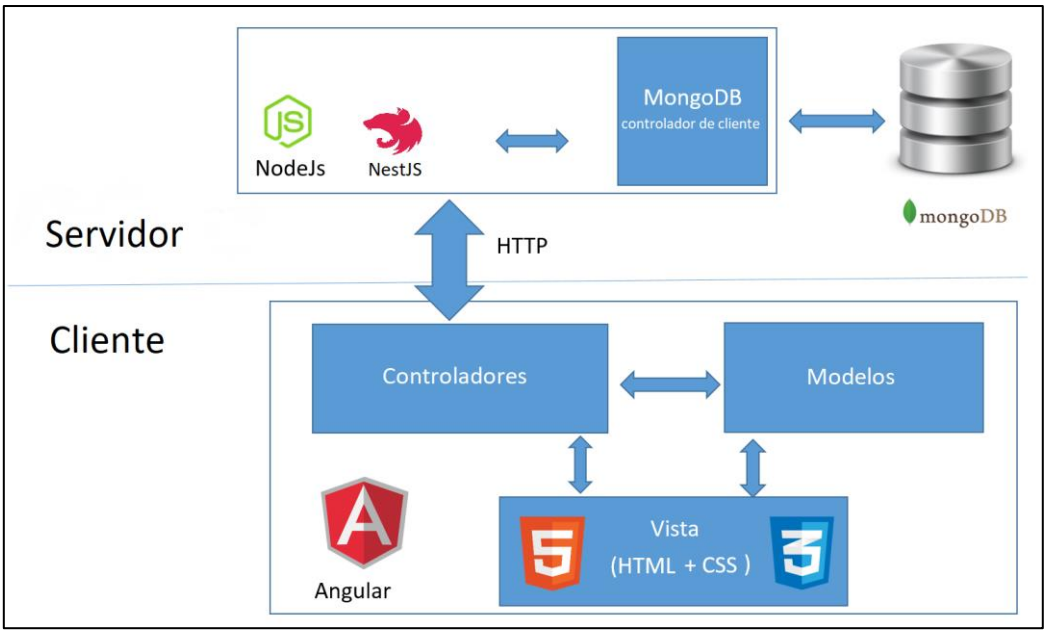

Figura 13. Arquitectura REST del proyecto.

Los servicios web *RESTful* se basan en la transferencia de estado representacional (*REST*), un estilo arquitectónico y un enfoque de las comunicaciones, que se suele utilizar en el desarrollo de servicios web. Estos servicios deben cumplir las siguientes condiciones:

- Uso de interfaz uniforme (UI).
- Basado en cliente-servidor.
- Operaciones sin estado. Todas las operaciones cliente-servidor deben ser sin estado, y cualquier gestión de estado que se requiera debe llevarse a cabo en el cliente, no en el servidor.
- Caché de recursos *RESTful*.
- Sistema de capas.
- Código bajo demanda.

Para concluir, hemos escogido hacer una *API RESTful* a una *SOAP* porque usa menos ancho de banda, lo que la hace más adecuada para un uso eficiente de Internet. Y es la más adecuada ya que necesitamos que el *Front-End* y el *Back-End* se comuniquen para el correcto funcionamiento del sistema.

C. NestJS

NestJS [11] es un *framework* progresivo para NodeJS, el cual nos permite crear servicios web de tipo *API RESTful* de forma eficiente. Además, ofrece una arquitectura muy sólida, escalable, fácil de mantener y fácil de testear.

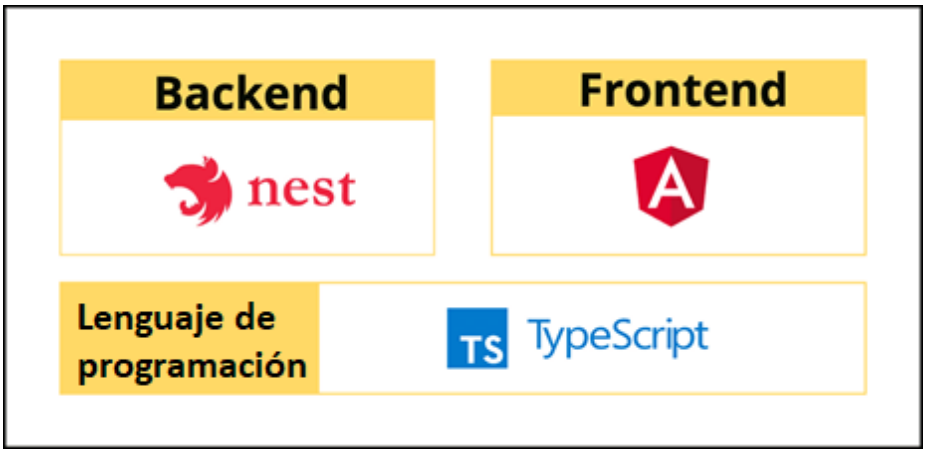

Figura 14. Estructura con NestJS como Back-End.

Hemos integrado este *framework* a nuestro proyecto, especialmente para la parte de *Back-End* ya que cuenta con una gran cantidad de librerías para llevar a cabo la parte del servidor.

D. NodeJS

NodeJS [12] es un entorno de tiempo de ejecución de JavaScript asíncrono controlado por eventos. Está diseñado para construir fácilmente aplicaciones rápidas y escalables.

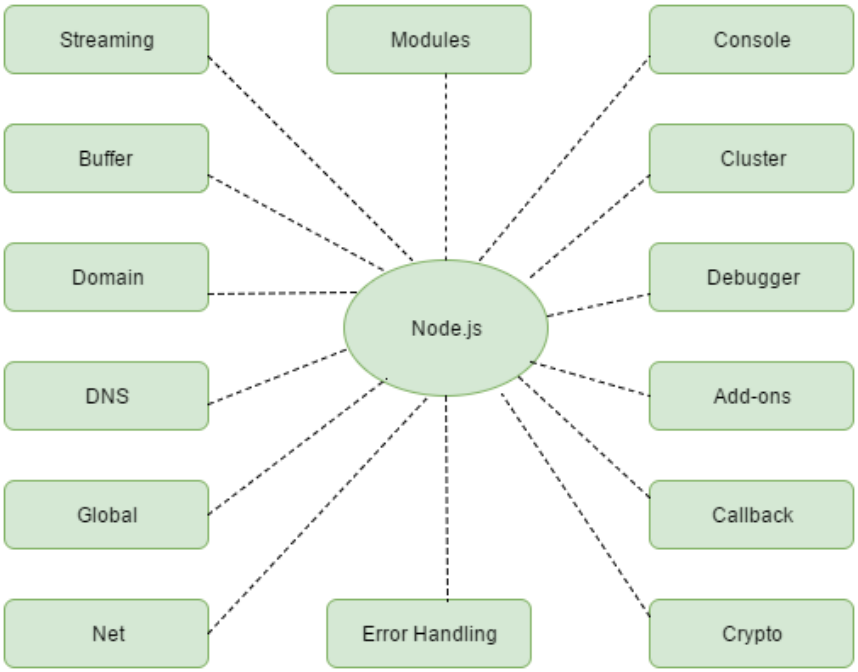

Figura 15. Componenetes NodeJS.

Las funciones en este entorno no suelen realizar directamente E/S, por lo que el proceso nunca se bloquea. Esto lo hace ligero y eficiente; un entorno perfecto para un sistema escalable como es el nuestro.

E. OpenPGP

OpenPGP es un protocolo no propietario que por medio de criptografía de clave pública encripta la comunicación mediante correo electrónico. Este protocolo define los formatos estándar para mensajes cifrados, firmas y certificados para el intercambio de claves públicas.

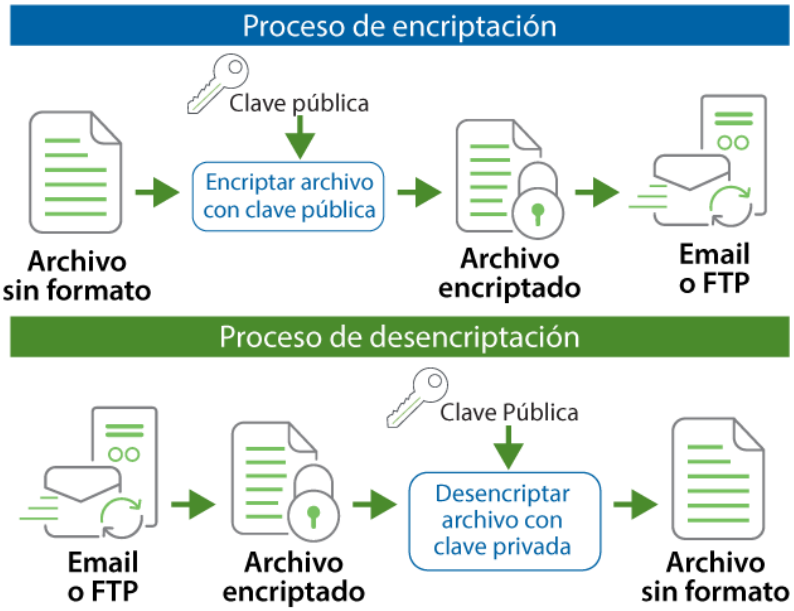

Figura 16. Proceso de encriptación y desencriptación con OpenPGP.

En el proyecto presente, utilizaremos este programa tanto para la parte del cliente como del servidor. Para ello el médico firmará un documento digitalmente, mediante una aplicación como es Kleopatra [13], y lo enviará junto a su clave pública. Seguidamente, el servidor recibirá los dos archivos y los autentificará, dando acceso a los datos del paciente solicitado en caso positivo.

F. Postman

Postman [14] es una herramienta de desarrollo de software. Permite el envío de peticiones HTTP *REST* sin deber de tener un cliente. Para ello, creamos los *end-points* de la parte del servidor con el mismo nombre en Postman. La información nos la devuelve por la consola que tiene esta misma herramienta.

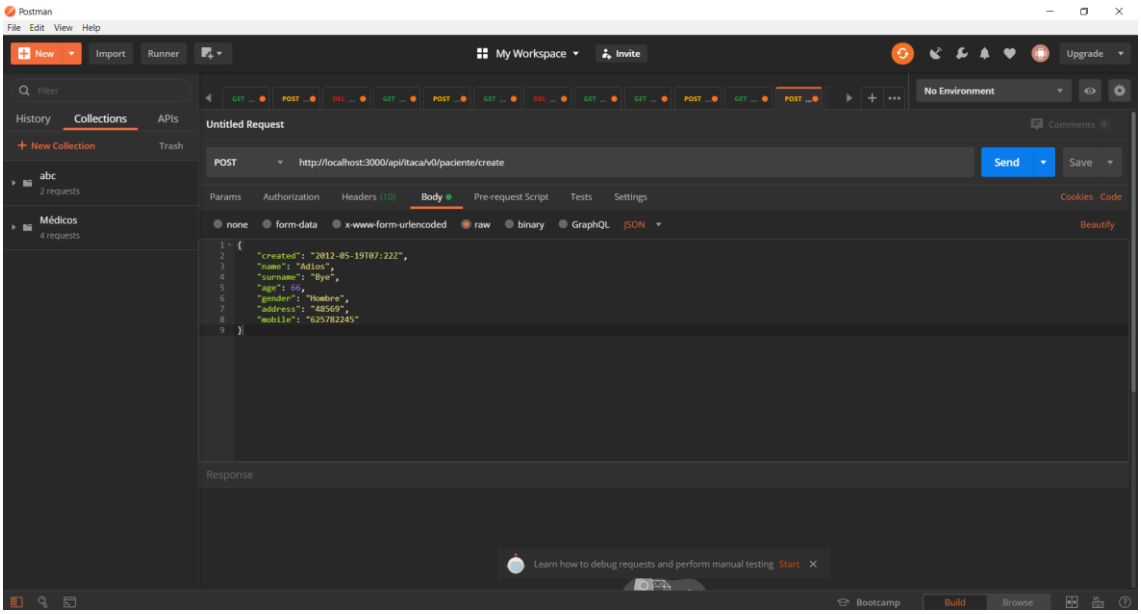

Figura 17. Ejemplo de uso - Añadir paciente en Postman.

Hemos hecho uso de esta herramienta para testear que los métodos que íbamos creando funcionarán correctamente, pudiendo añadir, consultar, actualizar y eliminar datos que llamamos en los métodos. Además, es una herramienta fácil de usar e intuitiva.

G. Angular

Angular [15] es un *framework* para el diseño de aplicaciones web dinámicas. Nos permite usar HTML para la estructura visual. Su objetivo es aumentar las aplicaciones web con la capacidad Modelo-Vista-Controlador (MVC) y reducir la cantidad de código necesarios para que la aplicación funcione.

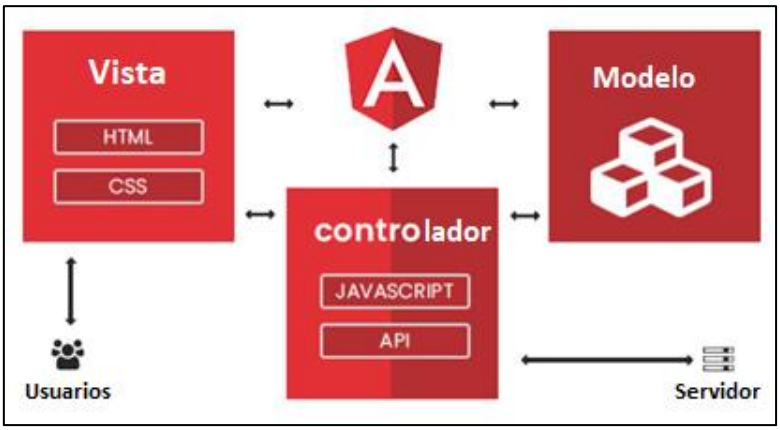

Figura 18. Arquitectura MVC Angular.

En el proyecto presente, utilizamos Angular para la parte de *Front-End*. También hemos incorporado una librería para mejorar los estilos visualmente llamada mdbootstrap[9].

Esta librería incorpora iconos a agregar en nuestra página, tablas para la visualización de datos, cómo hacer formularios para el *Front-End* y más.

H. Git

Git [16] es un sistema de control de versiones para rastrear cambios en archivos de computadora y coordinar el trabajo en esos archivos entre varias personas. Se utiliza principalmente para la gestión del código fuente en el desarrollo de software, pero se puede utilizar para realizar un seguimiento de los cambios en cualquier conjunto de archivos.

Para subir nuestro proyecto para que las demás personas puedan bajárselo debemos de ejecutar las siguientes órdenes: git *add*, git *commit* y git *push* en este mismo orden. Para la persona que desee ver el proyecto subido debe hacer la orden git *pull* y actualizará su código actual.

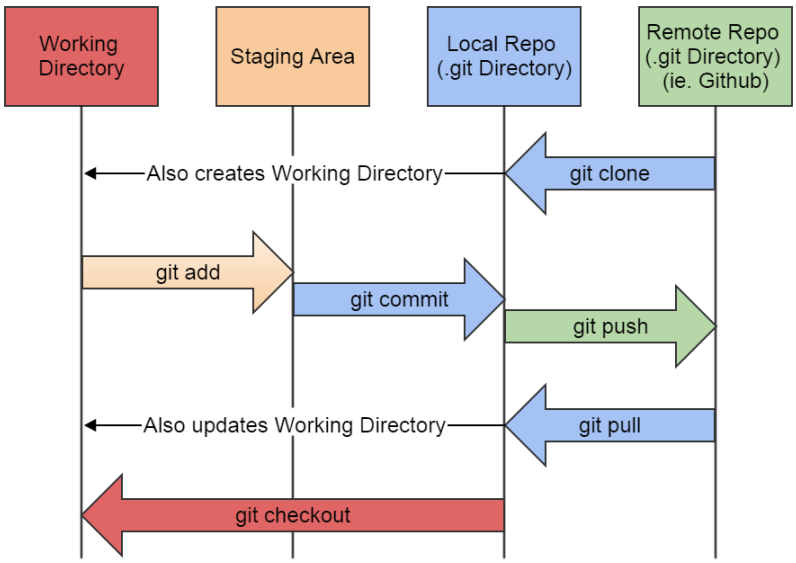

Figura 19. Acciones Git.

[9] https://mdbootstrap.com/

En nuestro proyecto este software ha sido clave para la coordinación del proyecto a la hora de visualizar el código ya sea por añadir, cambiar o eliminar código ya que hemos tenido que trabajar en remoto.

6. Desarrollo de la solución propuesta

En esta sección explicaremos la interacción con el sistema. Se mostrará el código de las funciones que son complejas. Nuestra aplicación está dividida en dos partes: *Front-End*, el cliente o parte visual, y *Back-End*, el servidor, que está dividido en dos carpetas: paciente y médico.

El funcionamiento de la aplicación realizada durante este proyecto es el siguiente: al encender el servidor se accede a una dirección web, accediendo a la vista del *Front-End*, esta vista, cuando sea necesario, se comunicará con el servidor y la capa de lógica para obtener datos de la base de datos que se usarán para su visualización en la página HTML, la cual las renderizará, o crear y/o actualizar los datos de pacientes en los formularios.

6.1. Población de MongoDB

En un principio la base de datos se iba a inicializar con datos recogidos de los sensores, pero, debido al COVID-19, no se ha podido implementar nuestra aplicación junto al equipo hardware desarrollado el año pasado.

De este modo, la base de datos se inicializa vacía sin tener médicos ni pacientes. Como la población de la base de datos es una clave importante, el usuario administrador debe añadir un *.csv* con los datos de las variables fisiológicas de pacientes, ya que esta aplicación es un modelo piloto y se pueden crear datos no reales.

Después, para el ingreso de médicos y pacientes, el usuario encargado podrá crear los debidos mediante la herramienta Postman, siguiendo la estructura de los datos en formato JSON establecidos. La colección de estos se guarda en la base de datos.

6.2. Pantalla principal y autentificación

La vista de la pantalla principal es un formulario donde se debe de introducir el usuario y contraseña del médico. Para poder pasar al *Home* de la aplicación debemos recibir un *true* por parte del servidor para verificarnos que existe tal usuario junto a esa contraseña. Esta verificación se hace mediante el método *medicoLogin()*, donde primero

busca mediante el método *findOne()* el usuario introducido y si lo encuentra verifica la contraseña, devolviendo *true* en caso de que exista o false en caso negativo.

Una vez el usuario se ha autentificado accede a la pestaña principal de la aplicación. En esta pestaña se encontrará con una aplicación estilo *dashboard*. Un *dashboard* es una interfaz de usuario gráfica donde se puede visualizar a primera vista todos los elementos importantes de la página. Esta pestaña es la denominada *Home* donde se visualiza el número de médicos y pacientes en tiempo real ingresados en la base de datos, el número de pacientes con tensión alta y baja, y, gráficas diferenciando hombres y mujeres en cuanto a los pacientes.

6.3. Lista de pacientes

Esta es la segunda opción en el menú lateral. Esta ventana muestra una tabla con todos los pacientes del hospital junto a sus datos generales. El usuario administrador podrá añadir un paciente a la base de datos mediante un formulario sencillo, que al darle a enviar se enviarán los datos al servidor. El método encargado de crear el paciente es *createUser(),* que a la vez tiene otro método *pacienteExists()* para verificar si el paciente ya existe. En caso de que no exista, se incorporan los datos a la base de datos satisfactoriamente y se aumenta el número de pacientes en la tabla y *Home*.

También, el usuario administrador puede editar los datos y eliminar pacientes. La primera acción abre un formulario con los datos presentes del paciente y el usuario puede modificar el dato que desee. El método encargado de esto se encuentra en la carpeta de paciente en el servidor llamado *update*(). Éste encuentra el id del paciente que se quiere modificar y actualiza los datos. Para la acción de eliminar, el método *delete*() encuentra el id del paciente a eliminar y lo borra de la base de datos.

Por último, se puede acceder al perfil del paciente donde podemos observar todos los datos, incluyendo sus variables fisiológicas. Para ello, el usuario primero deberá rellenar un formulario introduciendo su usuario y adjuntando los ficheros: claves públicas y documento firmado digitalmente. Una vez que el servidor reciba estos datos verificará que el intento de acceso sea de un médico ingresado en la base de datos. En caso positivo, accederá a la vista de datos del paciente.

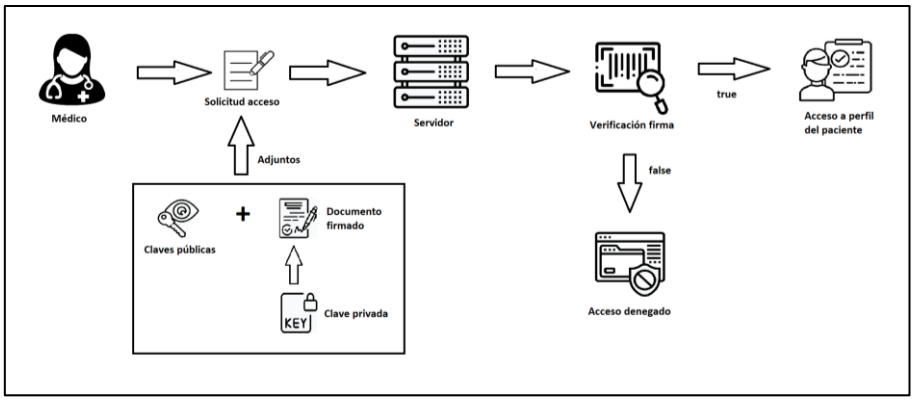

Figura 20. Seguridad en acceso a datos.

6.4. Visualización perfil paciente

Esta vista HTML está dividida en dos columnas: la primera se encuentra la izquierda, donde se puede visualizar los datos básicos del paciente, también visibles en la tabla de pacientes que es la vista anterior; y, la segunda se encuentra en la derecha, donde se puede observar las variables de un paciente en tiempo real.

El código está compuesto por *scripts* que dibujan gráficas y tablas haciendo uso de mdbootstrap para una visualización más detallada a primera vista. Estas gráficas deberían recibir los datos reales de pacientes mediante los sensores, pero como ya he explicado antes, se hará uso de datos introducidos mediante archivos *.csv*.

6.5. Dificultades encontradas durante el desarrollo

A la hora del diseño e implementación de la capa de seguridad, tuvimos que ir implantando varias soluciones por incompatibilidades encontradas y las herramientas que tendría que utilizar el usuario para enviar el fichero encriptado o firmado.

Otro de los desafíos ha sido no poder implantar la aplicación junto a la capa de seguridad con el *hardware,* ya que la realización de este proyecto empezó en febrero de 2020, y, nos encontramos con la situación del COVID-19 y el cierre de los laboratorios.

7. Implementación

Una vez iniciado el servidor se podrá desplegar la aplicación en local, pudiendo los usuarios hacer uso de sus funcionalidades. En esta sección vamos a observar las funciones principales que puede hacer el usuario.

Empezamos con la primera ventana que aparece al iniciar la aplicación, el *login*. Se compone de un formulario simple a introducir el usuario y contraseña.

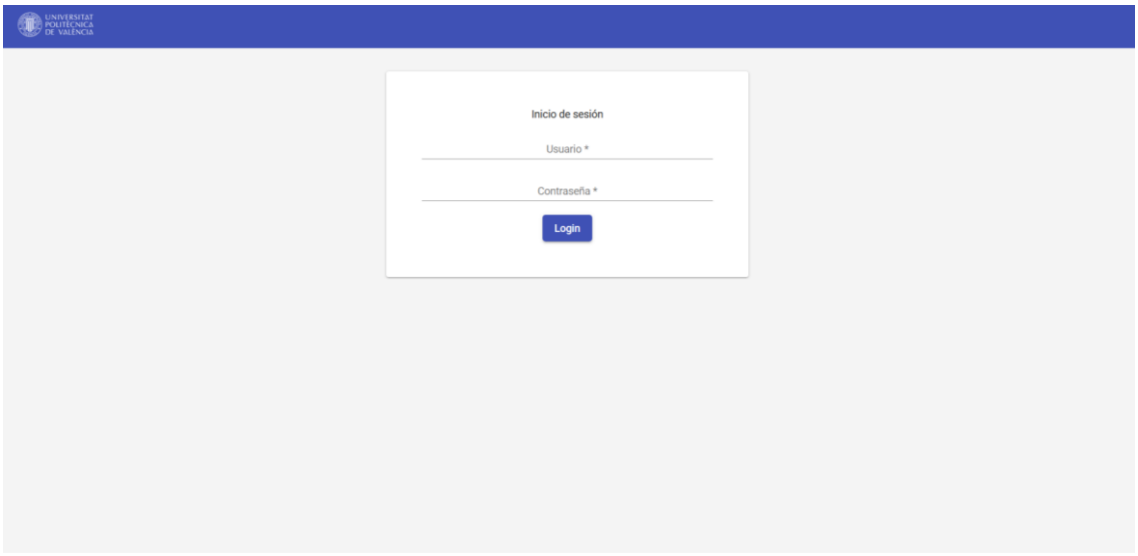

Figura 21. Login médico.

Una vez el usuario ha iniciado sesión, visualizará la ventana principal, *Home*. En la Figura 22 podemos observar la vista que tendrá el usuario. Está compuesta por 6 *cards*: la primera, el número de pacientes en el hospital conectados al dispositivo; la segunda, los médicos; la tercera, el número de pacientes con tensión alta; la cuarta, el número de pacientes con tensión alta; la quinta, una gráfica de barras horizontales que indican la edad de los pacientes; y, por último, una gráfica con el pulso de los pacientes.

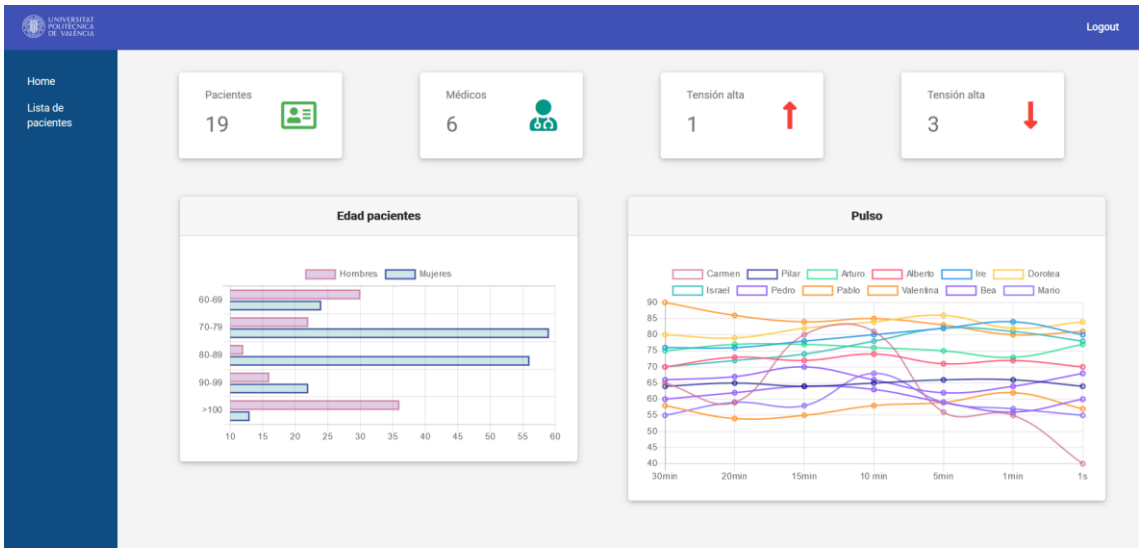

Figura 22. Home.

La segunda opción del menú nos ofrece la lista de los pacientes. En esta ventana podremos hacer las siguientes funciones: añadir un paciente, editar los datos de un paciente, eliminarlo y acceder a los datos médicos específicos del paciente.

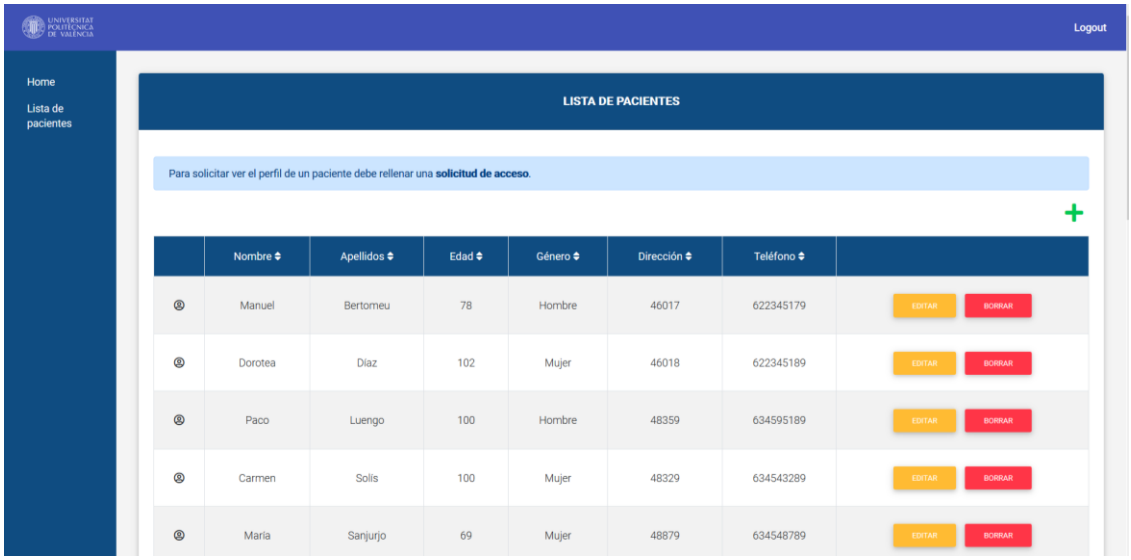

Figura 23. Vista: Lista de pacientes.

Para acceder a los datos fisiológicos deberemos de clicar en el icono de la izquierda con forma de usuario de un paciente. Entonces mediante un *dialog* nos pedirá que rellenemos un formulario. Esto corresponde a la capa de seguridad que hemos insertado. El usuario deberá de introducir su usuario, un documento firmado digitalmente y sus claves públicas para su verificación y acceso a los datos.

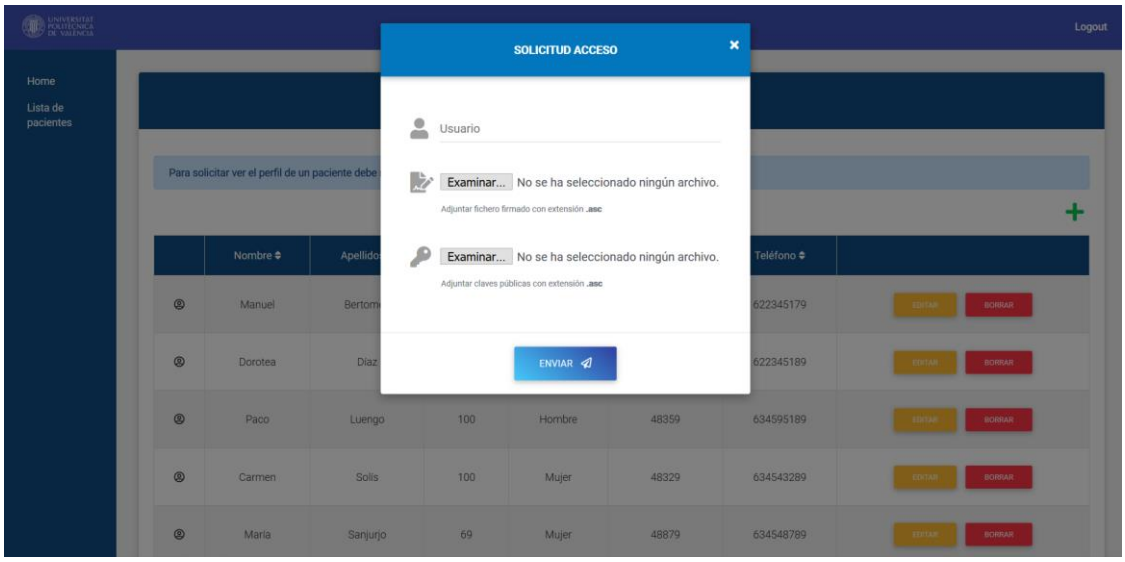

Figura 24. Solicitud de acceso a datos de un paciente.

Otra de las funciones que hemos nombrado es la de añadir un paciente. Para ello, clicaremos en el icono verde en forma de suma. Y aparecerá otro *dialog* con un formulario indicándonos los datos a rellenar para este nuevo paciente.

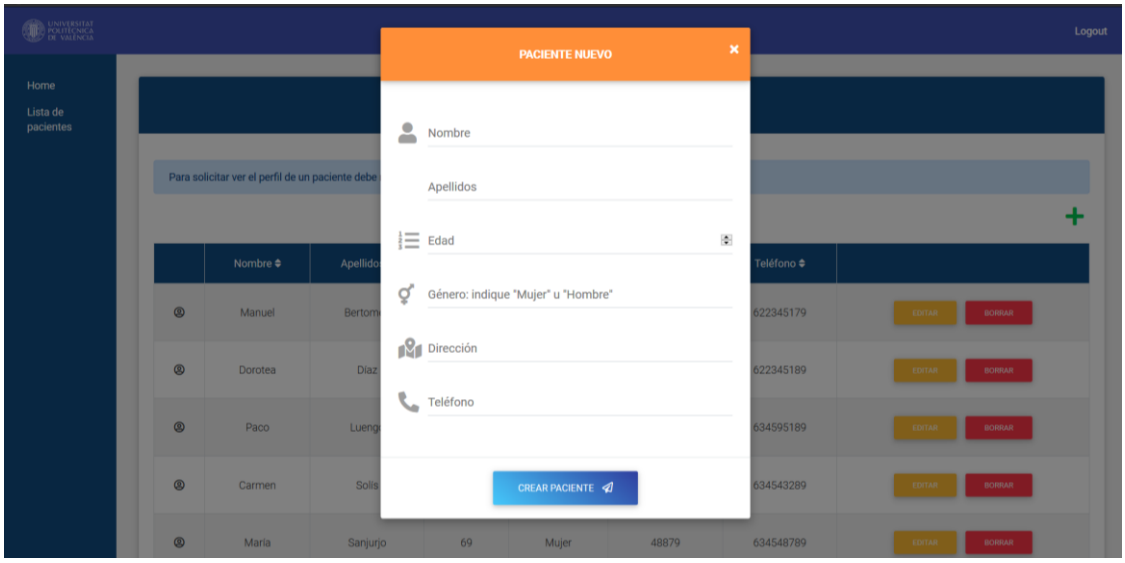

Figura 25. Añadir paciente.

En continuación a la Figura 23, al pasar la verificación nos llevará al perfil del paciente, como es la Figura 26. Se visualizarán los datos de la ficha personal ya disponibles en la tabla anterior, y, además, sus variables fisiológicas expuestas en varios tipos de gráficas.

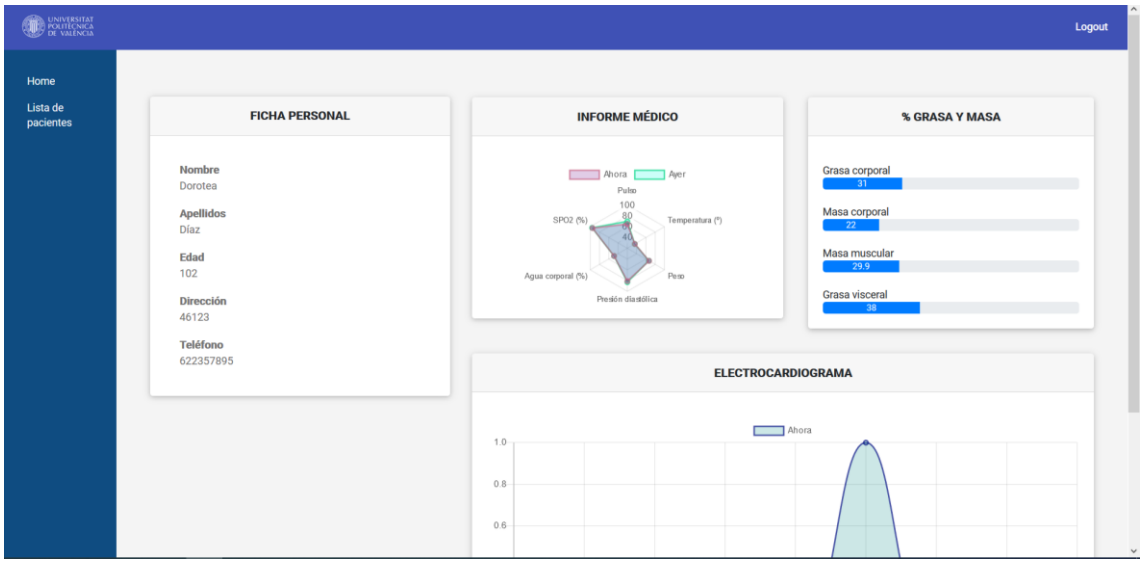

Figura 26. Perfil paciente.

8. Pruebas

En esta sección vamos a comprobar que el sistema cumple con los requisitos mediante una serie de pruebas, y, que la aplicación funciona correctamente.

8.1. Pruebas modulares

Por medio de estas pruebas comprobaremos que los casos de uso se llevan a cabo de forma correcta.

Prueba 1	
Casos de uso	CU01 Autentificarse.
Prueba	Rellenar *login*. En caso de no rellenarlo o no estar los datos introducidos en la base de datos debe dar error.
Resultado	Los datos introducidos han sido validados. Al dejar los campos vacíos o ser erróneos se lanza un error visible en la Figura siguiente.

Tabla 33. Prueba 1.

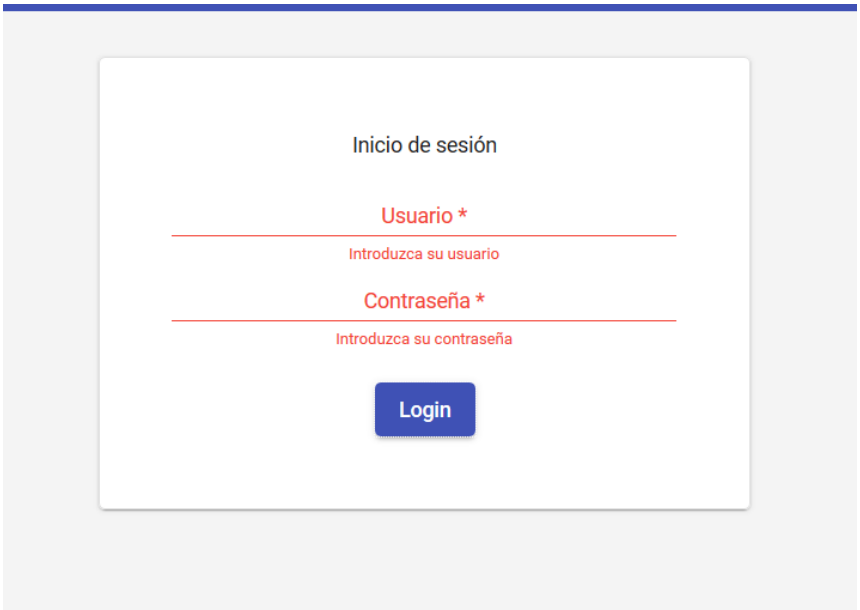

Figura 27. Error campos erróneos y/o vacíos.

Prueba 2	
Casos de uso	CU02 Registrar un paciente nuevo, CU07 Modificar un paciente.
Prueba	Rellenar formulario. Si se dejan campos vacíos no insertará el paciente en la tabla ni en la base de datos.
Resultado	El paciente se inserta satisfactoriamente en la base de datos, se puede observar en la Figura 29. En caso de no rellenar algún campo se lanza un error como el de la Figura 28.

Tabla 34. Prueba 2.

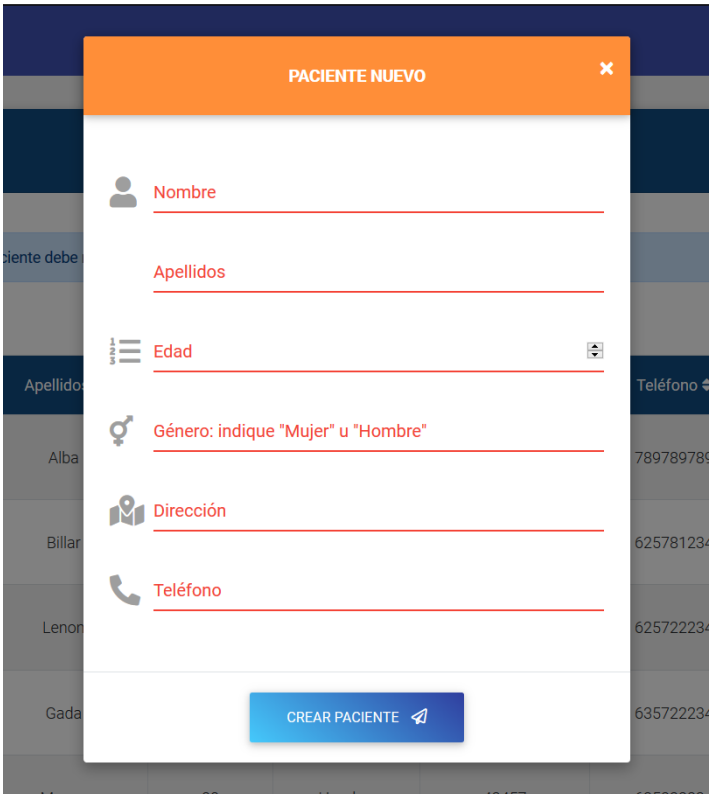

Figura 28. Crear paciente - Error campos vacíos.

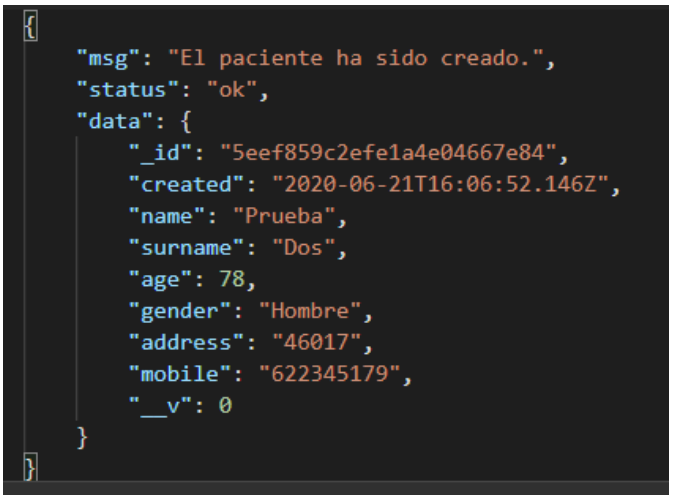

Figura 29. Resultado paciente creado.

Prueba 3	
Casos de uso	CU03 Visualizar todos los pacientes, CU06 Observar datos y gráficas generales.
Prueba	Visualización correcta de los datos en tabla de pacientes y *Home*.
Resultado	Tras iniciar sesión, añadir, modificar y/o eliminar un paciente se puede visualizar el *Home* y la lista de pacientes con datos correctos (Figura 23).

Tabla 35. Prueba 3.

Prueba 4	
Casos de uso	CU04 Ver datos de un paciente.
Prueba	Visualización perfil de paciente.
Resultado	Tras verificar el usuario del médico, se puede observar la ficha médica junto a sus variables fisiológicas del paciente específico (Figura 26).

Tabla 36. Prueba 4.

Prueba 5	
Casos de uso	CU05 Eliminar un paciente.
Prueba	Eliminar paciente del sistema.
Resultado	El paciente se elimina satisfactoriamente en la base de datos, se puede observar en la Figura 30; y se actualiza la tabla de pacientes.

Tabla 37. Prueba 5.

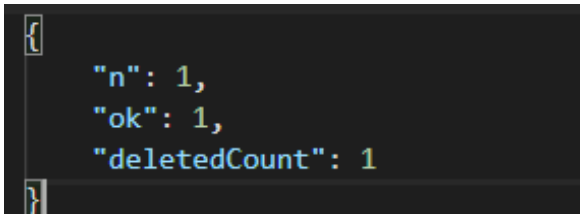

Figura 30. Paciente eliminado.

Prueba 6	
Casos de uso	CU08 Enviar firma digital.
Prueba	Envío y verificación del médico para acceso a datos sensibles del paciente.
Resultado	El usuario se verifica correctamente. En caso de dejar algún campo vacío se lanza un error como el de la Figura 31.

Tabla 38. Prueba 6.

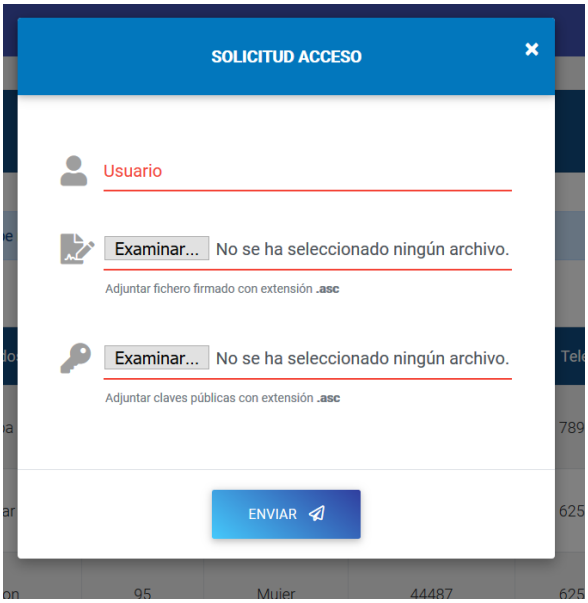

Figura 31. Solicitud acceso - Error campos vacíos.

```
gpg: cifrado con clave de 2048 bits RSA, ID 2C6F90269EEEDC04, creada el 2020-05-20
      "Carmen <maallue@inf.upv.es>"
El fichero 'carmen.txt' ya existe. ¿Sobreescribir? (s/N) N
Introduzca nuevo nombre de fichero: desencriptado
gpg: Firmado el 05/20/20 18:33:56 Hora de verano romance
gpg:                usando RSA clave 962BF02A249096C6D6EDDD26D68B5DCEF07170B1
gpg: Firma correcta de "Carmen <maallue@inf.upv.es>" [absoluta]
```

Figura 32. Verificación correcta firma.

8.2. Informe de usabilidad de la aplicación web

Uno de los objetivos de la aplicación web es que el usuario se relacione con facilidad con la interfaz y su navegación. Para llevarlo a cabo comprobaremos que se cumplen unos criterios de usabilidad.

En este apartado, vamos a comprobar mediante una lista si se cumplen los diez principios de la usabilidad establecidos por Jakob Nielsen [17].

1. **Visibilidad del estado del sistema.**
   Este principio se cumple ya que la aplicación muestra al usuario lo que está pasando y dónde se encuentra en cada momento.
2. **Adecuación entre el sistema y el mundo real.**
   Este principio se cumple ya que la aplicación tiene establecido como idioma el castellano, que es el idioma nativo de los usuarios. Además, utiliza un vocabulario general y sencillo. A su vez, hace uso de iconos intuitivos, por ejemplo, asociar la acción de añadir paciente clicando en el icono verde de sumar.
3. **Libertad y control por el usuario.**
   Este principio se cumple ya que en todo momento en la aplicación puedes navegar a las opciones de menú mediante un solo clic.
4. **Coherencia y estándares.**
   Este principio se cumple ya que la aplicación web sigue un patrón igual en sus pantallas, y así, no confundir a los usuarios.
5. **Prevención de errores.**
   Este principio se cumple ya que en los formularios donde se debe introducir datos, si son erróneos o están vacíos la aplicación avisa al usuario mientras los escribe.
6. **Reconocer mejor que recordar.**
   Este principio se cumple al haber botones con su función escrita encima de ellos, poniendo también colores reconocibles según sus acciones.
7. **Flexibilidad y eficiencia de uso.**
   La aplicación web cumple con este requisito al poder ser utilizada tanto por nuevos usuarios como expertos, ya que es intuitiva y su interacción es fácil de usar y secuencial.
8. **Estética y diseño minimalista.**

La aplicación cumple este requisito al tener un diseño minimalista para favorecer los elementos realmente importantes.

9. **Ayuda al usuario en errores.**

   Este principio se cumple al indicar al usuario en tiempo real cuando está cometiendo errores y se indica lo que debe hacer.

10. **Ayuda.**

    Este principio se cumple al tener avisos en la misma pantalla para realizar acciones más complejas, indicando lo que debe hacer.

9. Conclusiones

En este trabajo de final de grado se ha diseñado e implementado una aplicación web segura que consta de dos capas bien separadas: *Front-End* y *Back-End*.

El *Back-End* proporciona una *API RESTful*, a través de la cual cualquier sistema agnóstico se puede conectar y poder garantizar la interoperabilidad con otros sistemas.

La aplicación web diseñada garantiza que los médicos puedan acceder a la información de los pacientes de forma segura. La información a la que actualmente se accede es a los datos clínicos de adultos mayores, dichos datos proceden de un sensor, pero fácilmente se puede extender a la captación de datos procedentes de más sensores. Asimismo, hemos conseguido un sistema escalable gracias al uso de *MongoDB* como base de datos y la implementación de una *API RESTful,* garantizando su extensibilidad.

La filosofía de diseño de la aplicación web, ha sido la de contribuir a mejorar la atención médica a adultos mayores, que necesiten de una monitorización en tiempo real, utilizando para ello las tecnologías detrás de Internet, garantizando la seguridad, no sólo de los datos adquiridos, sino también del acceso por parte del personal médico a dichos datos y garantizando el acceso al sistema desde cualquier sistema *Front-End* implementado con tecnologías que sean capaces de acceder a una *API RESTful*.

En otras palabras, la aplicación web desarrollada cumple los principios de la seguridad de la información, i) la confidencialidad mediante el sistema de autentificación y verificación de usuarios, ii) la disponibilidad al tener la información disponible siempre que el usuario autorizado lo requiera, por último, iii) la integridad al asegurar los datos por medio de la seguridad implantada en el sistema para evitar su manipulación por usuarios no autorizados.

Este proyecto ha sido un desafío en diferentes niveles, tanto a nivel tecnológico como a nivel de aprendizaje. Si bien se han empleado conocimientos adquiridos durante los estudios (mencionados en el siguiente apartado) pero también ha sido necesario aprender nuevas metodologías, tecnologías y conceptos de los que no tenía conocimiento previo.

Uno de los obstáculos fue el desarrollo de la aplicación web. El uso del *framework NestJS*, *Angular* y *MongoDB* era nuevo para mí y requirió un tiempo de aprendizaje antes de empezar la aplicación del trabajo.

Otra dificultad ha sido diseñar una capa de seguridad mediante *OpenPGP* en un sistema con dispositivos *IoT*. Primero, debía comprender qué son las tecnologías *IoT* en el entorno *eSalud*, que tras investigar sobre estas he aprendido mucho y me he dado cuenta de la cantidad de utilidades que tiene. Y, seguidamente, implantar seguridad en el sistema, la cual fue el motivo por el que decidí hacer este proyecto a pesar de no tener un gran conocimiento sobre este.

A modo de conclusión, gracias a este proyecto he descubierto y aplicado nuevas tecnologías y sus aplicaciones en el ámbito médico, el cual quería descubrir su informática más a fondo, como es el Internet de las Cosas. A su vez, el desarrollo de una aplicación con una base de datos no relacional, *MongoDB*, ha sido gustoso ya que no se imparte en las asignaturas del grado; y he conseguido adquirir más conocimiento y especializarme en la integración de aplicaciones, que está relacionado con mi rama de la carrera. Además, he aprendido a gestionar el uso de mi tiempo.

Por otro lado, puedo afirmar, que la colaboración con dos grupos de investigación "Tecnologías de la información contra el cambio climático" (ICTvsCC) y el grupo SABIEN ambos perteneciente al Instituto ITACA me ha servido para trabajar con investigadores con conocimientos en *IoT* y aplicaciones médicas, los cuales me han ayudado a poder realizar la aplicación.

Por último, he descubierto cómo aplicar los conocimientos y tecnologías aprendidas en el grado, los cuales vamos a detallar en la siguiente sección.

9.1. Relación del trabajo desarrollado con los estudios cursados

Para la realización de este trabajo han sido necesarios los conocimientos provenientes de los estudios cursados en el "Grado de Ingeniería Informática", teniendo en cuenta tanto las asignaturas obligatorias como las optativas de la rama de Tecnologías de la Información. A continuación, se describen las asignaturas de grado que han sido más útiles:

- Para la planificación del proyecto, incluyendo el plan de trabajo, especificación de requisitos y presupuesto entre otros, se han empleado los conocimientos adquiridos en la asignatura de "Gestión de proyectos".

- Para los conocimientos de legislación para la seguridad y protección de los datos, se han empleado los conocimientos adquiridos en la asignatura de “Deontología y profesionalismo”.
- Para el diseño e implementación de la aplicación, se han empleado los conocimientos adquiridos en las asignaturas de “Interfaces persona-computador”, “Desarrollo centrado en el usuario” y “Desarrollo web”.
- Para los conocimientos de aplicaciones *REST* y *SOAP*, se han empleado los conocimientos adquiridos en la asignatura de “Integración de aplicaciones”.
- Para los conocimientos de base de datos, aunque la asignatura se centra en base de datos *SQL*, se han empleado los conocimientos adquiridos en la asignatura de “Base de datos”.
- Para el diseño de diagramas necesarios, se han empleado los conocimientos adquiridos en la asignatura de “Ingeniería del Software”.
- Para el análisis de posibles soluciones de seguridad, se han empleado los conocimientos adquiridos en las asignaturas de “Seguridad en redes y sistemas informáticos”, “Hacking ético” y “Ciberseguridad en dispositivos móviles”.

En cuanto a las competencias transversales podemos destacar:

CT-01. Comprensión e integración: ha sido necesario la adquisición de conocimiento de diversas fuentes y materia de las asignaturas.

CT-02. Aplicación y pensamiento práctico: ha sido necesario la realización de un análisis exhaustivo e interpretar los datos para hacer la implementación posteriormente.

CT-03. Análisis y resolución de problemas: esta competencia se ha cumplido y desarrollado en los apartados Análisis del problema y Diseño de la solución de este TFG.

CT-04. Innovación, creatividad y emprendimiento: este trabajo es una idea innovadora indudable. Aunque sea un proyecto piloto, tiene una visión futura de mercado.

CT-05. Diseño y proyecto: han sido necesarias técnicas de gestión de proyectos para el diseño y desarrollo de éste.

CT-06. Trabajo en equipo y liderazgo: esta competencia se ha cumplido al tener reuniones y desarrollar el trabajo con el grupo ITACA.

CT-07. Responsabilidad ética, medioambiental y profesional: se ha discutido el análisis ético y profesional sobre los datos sensibles y su seguridad.

CT-08. Comunicación efectiva: durante la redacción del trabajo se ha respetado los diferentes formatos sugeridos.

CT-09. Pensamiento crítico: se ha realizado un análisis del estado del arte como del estado actual del trabajo, argumentando la toma de decisiones.

CT-10. Conocimiento de problemas contemporáneos: la seguridad en dispositivos *IoT* y el desarrollo de aplicaciones en eSalud es un tema actual.

CT-11. Aprendizaje permanente: ha sido necesario adquirir nuevos conocimientos sobre las tecnologías utilizadas.

CT-12. Planificación y gestión del tiempo: ha sido necesario definir un plan de trabajo para llevar a cabo el proyecto.

CT-13. Instrumental específica: ha sido necesario el uso de tecnologías específicas, que han requerido un esfuerzo para su integración.

10. Trabajos futuros

En esta sección vamos a analizar mejoras y/o ampliaciones a realizar dentro del proyecto para un trabajo futuro.

Una mejora, sería aumentar la cantidad de sensores *IoT* que el sistema maneja. Esto se podría realizar mediante un protocolo de datos Bluetooth o vía el *Kit e-Health*.

También, otra mejora, sería implementar el sistema *hardware* actual del proyecto para que funcionase en una pulsera de tipo *smartwatch*, con su debida conexión Bluetooth. De esta manera, aumentaría la portabilidad del sistema.

Además, otra mejora, sería testear el sistema con un grupo de personas en un hospital. Este dato ha sido aportado por el grupo SABIEN-ITACA, que ya colabora con el grupo de Hospitalización Domiciliaria del Hospital La Fe.

Una última mejora, sería agregar al sistema registros de actividad de los médicos. Es decir, guardar QUÉ, QUIÉN y CUÁNDO solicita información de los pacientes.

Agradecimientos

En primer lugar, quiero agradecer al instituto ITACA, especialmente los departamentos de Salud y Tecnologías de la Información, de la Universidad Politécnica de Valencia y en especial a Lenin Guillermo Lemus Zúñiga por confiar en mí para llevar a cabo este proyecto y por la paciencia a la hora de guiarme y resolver dudas que ha habido durante este proyecto. A su vez, agradecer a todas las personas del departamento como Antonio Martínez Milana, Jorge E. Luzuriaga y Sara Macián Marí.

Y, por último, a mi familia y amigos, por haberme apoyado en momentos bajos que ha habido durante la carrera, y, haberme animado a estudiar lo que de verdad me gustaba y motivaba.

11. Referencias

[1] V INFORME SEIS - De la historia clínica a la historia de salud electrónica. Informe de la Sociedad Española de Informática de la Salud, 2003. [En línea]. [Consultado 26 Abril 2020]. [En línea]. Disponible en: http://www.conganat.org/seis/informes/2003/

[2] Ministerio de Sanidad y Política Social. Las TIC en el Sistema Nacional de Salud (SNS): El programa Sanidad [en línea]. Madrid 2010. [Consultado 4 Junio 2020]. Disponible en: http://www.ontsi.red.es/ontsi/es/estudios-informes/las-tic-en-el-sistema-nacional-de-salud-ed-2010

[3] DAVIS, DAN; PARASHAR, MANISH P. *Latency performance of SOAP implementations*. En *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*. IEEE, 2002. p. 407.

[4] KANKANAMGE C.; *Web services testing with SoapUI. Packt Publishing*, Birmingham, 2012.

[5] KOPNIAK P.; *SOAP system integration with web services*. Varia Informatica 2011, *Polish Information Processing Society*, 2011, 147–163.

[6] FIELDING R.T., TAYLOR R.N.; *Principled design of the modern web architecture*. *ACM Transactions on Internet Technology* 2(2), 2002. 115–150.

[7] RICHARDSON, LEONARD y RUBY, SAM; *RESTful web services*. Beijing: O'Reilly Media, 2007.

[8] La Historia Clínica Electrónica (HCE) en España, 2020. *Clinic Cloud* [en línea]. [Consultado 6 Junio 2020]. Disponible en: https://clinic-cloud.com/blog/historia-clinica-electronica-hce-espana/.

[9] *The Cost of the "S" in HTTPS | Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*: diciembre de 2014. pp. 133-140. Dl.acm.org [en línea]. [Consultado 10 Junio 2020]. Disponible en: https://dl.acm.org/doi/abs/10.1145/2674005.2674991.

[10] ¿Qué es MongoDB?, 2020. MongoDB [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://www.mongodb.com/es/what-is-mongodb.

[11] *Documentation* | NestJS - *A progressive Node.js framework,* 2020. NestJS [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://docs.nestjs.com/.

[12] *About* | Node.js, 2020. Node.js [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://nodejs.org/en/about/.

[13] Kleopatra, 2020. OpenPGP [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://www.openpgp.org/software/kleopatra/.

[14] Postman – *API Documentation,* 2020. Postman [En línea]. [Consultado 15 Junio 2020]. Disponible en: https://www.postman.com/api-documentation-tool/.

[15] Angular, 2020. Angular.io [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://angular.io/docs.

[16] Git – *Documentation,* 2020. Git [en línea]. [Consultado 15 Junio 2020]. Disponible en: https://git-scm.com/doc.

[17] NIELSEN, JAKOB, *Designing web usability: The practice of simplicity*. Indianapolis: New Riders, 1999.

Glosario

API: acrónimo del término inglés *Application Programming Interface*, Interfaz de Programación de Aplicaciones en castellano. Conjunto de funciones, comandos y protocolos informáticos para implementar la aplicación en un sistema.

Back-End: es el servidor, también conocido como la capa de acceso a datos.

Bluetooth: tecnología para redes inalámbricas.

Cliente: ordenador o aplicación informática que se conecta a otro programa, llamado servidor.

Dashboard: interfaz gráfica de usuario.

End-Point: el extremo de un canal de comunicación. Cuando una API interactúa con otro sistema, los puntos de contacto de esta comunicación se consideran puntos finales.

HTML: siglas del término ingles *HyperText Markup Language*, Lenguaje de Marcas de Hipertexto en castellano. Estándar implantado en la visualización de páginas web.

FHIR: del inglés *Fast Healthcare Interoperability Resources*. Estándar diseñado para la web.

Framework: entorno de trabajo.

Front-End: es el cliente, también conocido como la capa de presentación.

IoT: del inglés *Internet of Things*. Red digital de objetos conectados por Internet.

JSON: del inglés *JavaScript Object Notation*. Formato de texto para el intercambio de datos.

RAM: del inglés *Random Access Memory*. Memoria de trabajo.

REST: del inglés *Representational State Transfer*. Estilo de arquitectura *software*.

RESTful API: API que hace uso de operaciones HTTP (*GET*, *PUT*, *DELETE*, *POST*) para obtener datos.

RF: Requisito Funcional.

RI: Requisito de Interfaz.

RR: Requisito de Rendimiento.

Servidor: aplicación encargada de responder las peticiones de un cliente.

TIC: Tecnologías de la Información y de la Comunicación.

SOAP: del inglés *Simple Object Access Protocol*. Estilo de arquitectura software.

universAAL: plataforma de código abierto que ofrece interoperabilidad entre dispositivos a gran escala.

UPV: Universidad Politécnica de Valencia.