



UNIVERSIDAD COMPLUTENSE MADRID

TRABAJO DE FIN DE GRADO Esd2: Cuaderno de recogida de datos para el estudio médico

Eduardo Gonzalo Montero

&

Sergio Pacheco Fernández

Profesor director: Pablo Manuel Rabanal Basalo

Codirector: Alejandro Rabanal Basalo

Curso académico: 2019-2020

Identificación asignatura: Trabajo de fin de Grado, Grado en Ingeniería del Software, Facultad de Informática



UNIVERSIDAD COMPLUTENSE MADRID

FINAL DEGREE PROJECT Esd2:Data collection notebook for a medical research

Eduardo Gonzalo Montero

&

Sergio Pacheco Fernández

Director professor: Pablo Manuel Rabanal Basalo

Co-director: Alejandro Rabanal Basalo

Academic year: 2019-2020

Subject identification: Final Degree Project, Degree in Software Engineering, Computer Science Faculty

Índice

Resumen	4
Palabras clave	5
Abstract	6
Keywords	7
1. Introducción	8
1.1. Objetivo	8
1.2. Motivación	9
1.3. Metodología de trabajo	10
1.3.1. Transparencia	10
1.3.2. Inspección	11
1.3.3. Adaptación	12
2. Tecnologías	13
2.1. Lenguajes de programación	13
2.1.1. TypeScript	13
2.1.2. HTML-5	13
2.1.3. CSS-3	14
2.1.4. Java 8	14
2.1.5. SQL	14
2.2. Entornos de desarrollo	14
2.2.1. Visual Studio Code	14
2.2.2. MySQL Workbench	14
2.2.3. PhpMyAdmin	14
2.2.4. GitHub	15
2.2.5. BitBucket	15
2.2.6. Eclipse	15
2.2.7. Overleaf	15
2.2.8. MobaXterm	15
2.3. Frameworks	16
2.3.1. Java Spring	16
2.3.2. Angular	17
3. Desarrollo	18
3.1. Arranque del desarrollo	18
3.2. Desarrollo durante los sprints	23

4. Implementación	24
4.1. FrontEnd	24
4.1.1. Generación Proyecto Angular	24
4.1.2. Arquitectura de la aplicación Angular	25
4.1.3. Inicialización del proyecto	25
4.2. BackEnd	25
4.2.1. Generación Proyecto Java Spring Boot	25
4.2.2. Arquitectura de la aplicación Java Spring Boot	26
4.2.3. Inicialización del proyecto	34
5. Despliegue	36
5.1. FrontEnd	36
5.2. BackEnd	37
5.3. Base de Datos	38
6. Mantenimiento	40
7. Conclusiones	43
8. Trabajo Futuro	44
9. Bibliografía	45

Agradecimientos

”Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Resumen

La diabetes mellitus se encuentra entre las 10 principales causas de muerte a nivel mundial (en 2017, 4 millones de personas entre 20 y 79 años fallecieron debido a ella). Se calcula que alrededor de 425 millones de personas tienen diabetes actualmente en el mundo a pesar de que muchos de los casos permanecen sin registrar. Solo en Europa alrededor del 38 % de los casos de diabetes aún están sin diagnosticar, lo que supone unos 22 millones más de afectados. En España se estima que más de 5 millones de personas padecen esta enfermedad, dándose más de 380.000 nuevos pacientes cada año.

En el caso de la diabetes mellitus tipo 2 se estima que 9 de cada 10 casos pueden atribuirse a hábitos de vida que podrían modificarse promoviendo estilos de vida saludables como el deporte o seguir una dieta equilibrada, ya que la obesidad es uno de los mayores factores de riesgo. Sin embargo en los últimos años se han planteado también otros importantes factores de riesgo, entre ellos el déficit de Vitamina D. No obstante los umbrales de niveles correctos de esta vitamina son muy controvertidos y el estudio de su impacto, por tanto, es complejo.

Este proyecto consiste en desarrollar de un portal web que dará soporte a un equipo médico recopilando datos de contraste en pacientes con diabetes mellitus tipo 2. La aplicación les permitirá recopilar datos sobre los niveles de Vitamina D entre otros factores, vía formularios para posteriormente, compararlos y extraer conclusiones que esperan ayuden a delimitar mejor la enfermedad, promover su prevención y, en general, una mejor comprensión de la misma.

El objetivo es que el portal esté en funcionamiento antes de la culminación de este Trabajo de Final de Grado, con el fin de darle soporte durante sus primeros meses útiles, mediante tareas de mantenimiento, mientras los médicos desempeñan su labor. Se busca tras la finalización del proyecto, proporcionar a los médicos una herramienta útil y ajustada que les permita recopilar datos para avanzar en su investigación.

Palabras clave

- Servicio web
- Estudio médico
- Diabetes
- Vitamina D
- API-REST
- Investigación
- Sanidad

Abstract

Diabetes mellitus is one of the 10 main causes of death worldwide (in 2017, 4 million people between 20 and 79 years passed away due to it). It is estimated that around 425 million people currently suffer diabetes in the world, despite there are a lot of cases unregistered. Only in Europe, around 38% of diabetes cases even remain undiagnosed, which means 22 million people affected. In Spain, it is estimated more than 5 million people suffer this disease, with more than 380.000 new patients every year.

In the case of diabetes mellitus type 2, it is estimated 9 in 10 cases can be attributed to lifestyle that could be changed by promoting healthy lifestyles such as sport or having a balanced diet, since obesity is one the main risk factors. Nevertheless, in recent years other important risk factors have been raised, among them the vitamin D deficit. However, the correct levels threshold of this vitamin are so controversial, and the research impact therefore is complex.

This project is to develop a web portal which will provide support to a medical team compiling contrast data in patients with diabetes mellitus type 2. The application will allow them to collect data about vitamin D levels, among other factors through forms to compare them and abstract conclusions which are expected to define the disease in a better way, to promote its prevention and, generally, a better understanding of the same one.

The objective is that the portal will be functioning before the end of this Final Work Degree, in order to provide it support during its first useful months through maintenance tasks, while doctors perform their work. After project completion, is sought to provide a useful and calibrated tool to the doctors to let them collect data to make progress in their research.

Keywords

- Web service
- Medical research
- Diabetes
- Vitamin D
- API-REST
- Study
- Health

CAPÍTULO 1

Introducción

En este capítulo se detalla la motivación que llevó a la realización del proyecto, los objetivos que este comprende y las pautas que se tomaron para lograr llevarlo a cabo

1.1. Objetivo

Este Trabajo de Final de Grado busca la implementación de un portal web sencillo y compacto que permita la recopilación de datos de interés para el estudio por parte del equipo médico. Para ello contará con formularios personalizados con los campos de interés solicitados por el mismo que serán llenados por los médicos en consultas rutinarias con los pacientes que hayan consentido participar en el proyecto. Estos datos serán luego accesibles tanto vía web como en archivos de excel descargables desde el propio portal.

Con el fin de ajustar el producto final a las necesidades del equipo la aplicación se entregará con tiempo suficiente (meses de antelación) para que pueda ser probada, revisada y modificada a medida. Esto es especialmente importante ya que el estudio se extenderá muchos meses más allá de la finalización de este TFG.

El objetivo final, y por supuesto el más importante, es que todo este desarrollo sirva de herramienta para ajustar los criterios de evaluación en los pacientes con diabetes mellitus tipo 2, permitiendo encontrar pautas en sus estados de salud subsanables que puedan en un futuro prevenir más casos de esta enfermedad.

1.2. Motivación

Nuestra primera motivación para adentrarnos en este proyecto son las tecnologías empleadas en el mismo. Ambos participantes están interesados en centrar su carrera en tecnologías web, uno de ellos trabajando ya de hecho como desarrollador full-stack. El proyecto da además libertad para ser implementado sin ningún tipo de restricciones de diseño o rendimiento, lo que plantea un escenario ideal para experimentar durante su desarrollo.

El otro motivo principal para decantarnos por este proyecto es su cercanía a un proyecto real. El cliente, la aplicación, los plazos y las necesidades del mismo no son algo creado para un ambiente académico, como otros desarrollos ya efectuados durante la carrera, sino un problema real que precisa una solución realista contando con todas las personas que lo acabarán utilizando. Además el proyecto requerirá de un mantenimiento post entrega, otro ámbito nunca explorado en la carrera que será una valiosa experiencia de cara a nuestro futuro.

Por último remarcar que uno de los miembros, Sergio, ya tiene buenas experiencias con un proyecto anterior para el ámbito médico desarrollado en solitario para una empresa en Suiza y Eduardo siempre ha tenido interés en los hábitos de vida saludables y la nutrición, posible principal remedio para la diabetes extraído de los resultados de este estudio.

1.3. Metodología de trabajo

El proyecto será planificado bajo los estándares de la metodología Scrum aunque distendiendo un poco sus cotas temporales, pues al ser solo dos miembros no es necesario hacer un hincapié tan diario en la organización para mantener el orden. Lo primero será definir cómo mantendremos los tres pilares básicos de la metodología: **transparencia, inspección y adaptación**.

1.3.1. Transparencia

Para mantener a ambos miembros al día de cualquier avance o variación en el proceso se empleará la herramienta online de Trello, que nos permitirá ir viendo en tiempo real los mismos. Este será estructurado de la siguiente forma:

- Se crearán dos columnas, una con la lista de tareas pendientes para el sprint actual y otra con las tareas actualmente en desarrollo. Además se creará una columna por cada sprint en la que se irán almacenando todas las tareas finalizadas. Cada tarea podrá encapsularse en una o más de estas categorías: despliegue, documentación, front, bug, investigación, modelo de datos, pendiente de resolver (cuestión a espera de la próxima reunión con el tutor), back, varios, resuelto (cuestiones ya solucionadas en anteriores reuniones que quedan como recordatorio). Además cada tarea llevará una o más etiquetas indicando los desarrolladores que han trabajado en ella.

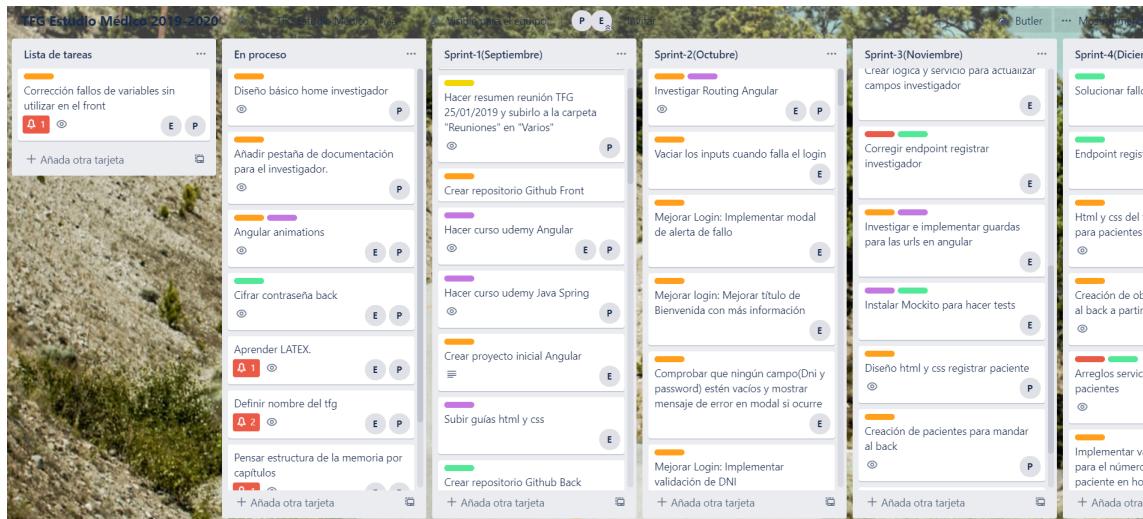


Figura 1.1: Captura de ejemplo de la herramienta Trello tomada a posteriori.

- Además todo el código del proyecto estará subido y actualizado en un repositorio, en este caso GitHub, a través del cual se podrá ver un histórico preciso de los cambios realizados en cada commit junto a comentarios explicativos del desarrollador que los haya realizado.

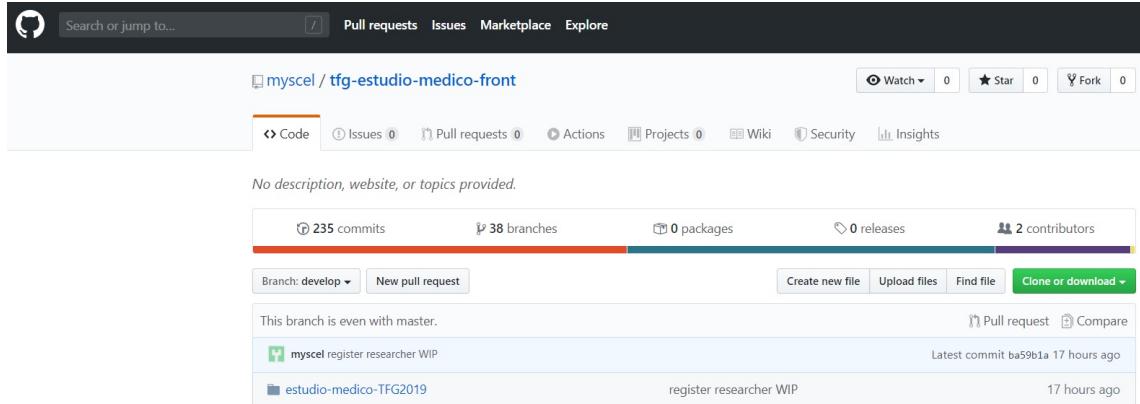


Figura 1.2: Captura de ejemplo del repositorio front en GitHub tomada a posteriori.

1.3.2. Inspección

Con el fin de mantener este principio nuestro repositorio no solo albergará código sino también un espacio separado para todos los artefactos derivados de scrum, así como la documentación que se vaya creando durante el proceso que sea relevante a efectos de esta memoria. Estos artefactos serán por tanto accesibles constantemente por ambos miembros, los cuales avisarán en caso de cualquier añadido o modificación para que su compañero pueda revisarlo.

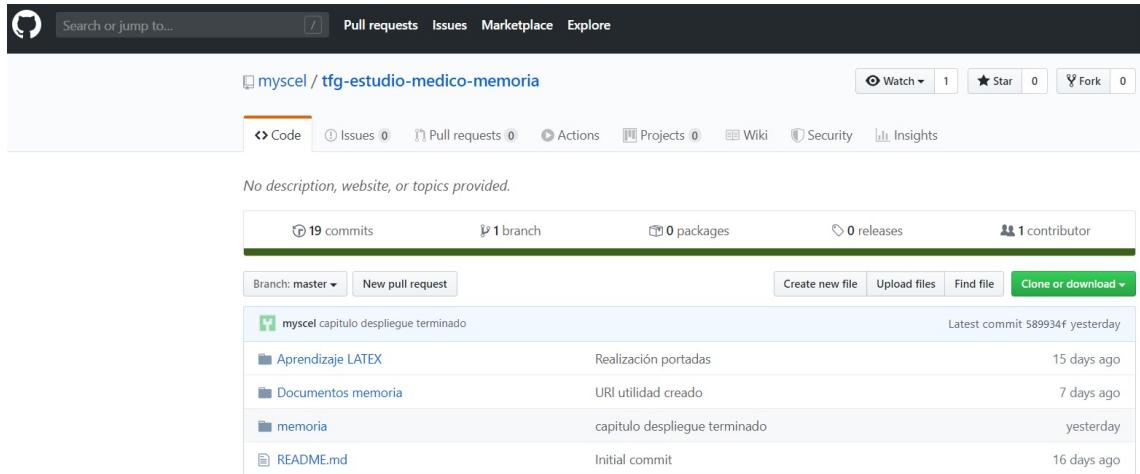


Figura 1.3: Captura de ejemplo del con todos los documentos referentes a la memoria en GitHub tomada a posteriori.

1.3.3. Adaptación

Al final de cada sprint se realizará una reunión tanto con el tutor del proyecto como con un representante del equipo médico para revisar el proceso y obtener feedback del mismo. Los errores o cambios urgentes extraídos de estas reuniones pasarán a ser las tareas más prioritarias del proyecto y su resolución será notificada de inmediato a ambos asistentes de la reunión. Además para mantener el proyecto lo más afín a las necesidades del cliente cualquier cambio significativo en mitad de un sprint será notificado por correo previamente.

Además de toda la documentación y artefactos mencionados disponibles durante el desarrollo los miembros mantendrán un contacto diario hablando de qué han hecho, qué van a hacer y los problemas que van encontrando, el cual sustituye las Daily Scrum.

En cuanto a los roles típicos de Scrum en nuestro caso no serán utilizados. El Producto Owner será suplido con la comunicación directa y constante de los miembros con el cliente que asegurara la priorización del valor para este durante el desarrollo. El Scrum Master no será preciso ya que ambos miembros del equipo tienen experiencia utilizando metodologías ágiles, tanto en la carrera con Scrum como fuera en trabajos o proyectos propios. Por tanto solo existirá el equipo de desarrollo y este suplirá las funciones necesarias del resto de roles.

Los sprints serán de una duración mensual. Se opta por esta cuantía porque al estar uno de los miembros aún con clases y otro trabajando a jornada completa no se estima que en una semana el desarrollo pueda tener un avance suficientemente significativo. Estos sprints se iniciarán con una reunión de planificación con el tutor del proyecto y el representante del equipo médico, estipulando qué se hará y cómo. Así mismo concluirán con otra reunión de misma índole que mostrará una demo funcional al representante y recogerá su feedback, para acto seguido comenzar con la reunión de planificación del siguiente sprint. Se decide aunar así el inicio y final del sprint en una sola reunión para evitar problemas de horarios difíciles de encajar por parte de los cuatro. Tras esta reunión con todos los miembros se realizará una pequeña reunión de retrospectiva solo del equipo de desarrollo para determinar si la metodología de trabajo funciona y qué se podría modificar o añadir.

CAPÍTULO 2

Tecnologías

En este capítulo se detalla todo lo relativo a los lenguajes de programación, los entornos de desarrollo y los frameworks elegidos para llevar a cabo este proyecto, así como sus características principales y la razón de su uso.

2.1. Lenguajes de programación

2.1.1. TypeScript

TypeScript es un lenguaje de programación orientado a objetos(OO) el cual es superset de JavaScript. Decimos que una tecnología es un superset de un lenguaje de programación, cuando puede ejecutar programas de la tecnología [1]. En resumen, ejecutará el código como si fuese JavaScript.

TypeScript se diferencia de JavaScript principalmente en que posee inferencia de tipos, es decir, está fuertemente tipado, además de algunas funcionalidades extra.

Este lenguaje se utiliza en el FrontEnd del proyecto, dado que el framework elegido para realizar esta parte es Angular, el cual se explicará en detalle más adelante.

2.1.2. HTML-5

HTML-5 (HyperText Markup Language) es la quinta revisión importante del lenguaje básico de la World Wide Web[2]. Se trata de un lenguaje de marcación para la elaboración del contenido de las páginas web. Hoy en día es el lenguaje estándar que aceptan la gran mayoría de los navegadores a la hora de la construcción de las páginas web.

HTML-5 se diferencia de sus versiones anteriores en que incorpora nuevas etiquetas(section, article, header, footer etc...) con las cuales se busca mejorar y estandarizar la estructura de las páginas web además de otras actualizaciones como la mejora de los formularios o la inclusión de elementos de audio y vídeo.

Se ha optado por utilizar este lenguaje de marcación debido a su popularidad y a la inclusión de nuevas etiquetas que favorecen la lectura de la página web por parte de los navegadores.

2.1.3. CSS-3

CSS (Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado[3]. Se utiliza en la mayoría de sitios web junto con html para generación de páginas web. De esta manera es mucho más sencillo generar páginas web, ya que, el diseño(CSS) se encuentra separado del contenido(HTML).

2.1.4. Java 8

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems [4]. Proviene de los lenguajes C y C++, y sus aplicaciones pueden ser ejecutadas en cualquier JVM(Java Virtual Machine).

Para este proyecto se ha utilizado la versión 8 porque esta versión es la que menos bugs tiene y la que mejora más la eficacia en el desarrollo y la ejecución de programas Java. [5] Además de estas razones, escogimos java por ser un lenguaje orientado a objetos ideal para desarrollar proyectos API-REST.

2.1.5. SQL

SQL es un lenguaje declarativo estándar internacional de comunicación dentro de las bases de datos que nos permite a todos el acceso y manipulación de datos en una base de datos [6].

Se decidió utilizar este lenguaje debido a su uso en el SGDB(Sistema de Gestión de Base de Datos) que utilizamos en el proyecto, MySQL. Además de esto, SQL es ideal para trabajar con JPA(Java Persistence API) en el BackEnd del proyecto.

2.2. Entornos de desarrollo

2.2.1. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.[7].

En este proyecto se ha utilizado este entorno de desarrollo para gestionar un proyecto Angular, ya que este editor posee gran versatilidad a la hora de instalar plugins y gestionar diferentes lenguajes de programación de manera simultánea.

2.2.2. MySQL Workbench

Se trata de un programa para gestionar, diseñar y administrar bases de datos relacionales, utilizado en nuestro proyecto a la hora de manejar datos de manera local.

Se decidió utilizar debido a que se ha usado previamente en nuestros estudios de grado en diversas asignaturas de manera productiva, además de que posee una versión gratuita.

2.2.3. PhpMyAdmin

Al igual que MySQL Worbench se trata de una herramienta de gestión de bases de datos Mysql, pero con la diferencia que el acceso a esta herramienta es vía web, alojándose en un servidor.

Esta herramienta la utiliza Hostinger[8], proveedor de alojamiento web donde se ha decidido alojar el proyecto para desplegarlo en la web.

2.2.4. GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código que funciona con git, el cual hemos utilizado para trabajar en equipo. Hemos creado tres repositorios; un repositorio para el FrontEnd del proyecto, otro repositorio para el BackEnd del proyecto y otro para guardar la memoria escrita en L^AT_EX.

Se decidió utilizar este sistema debido a la familiaridad que poseemos con el mismo, lo que se traduce en efectividad y productividad en el workflow del equipo.

2.2.5. BitBucket

BitBucket es, al igual que GitHub, un sistema de gestión de proyectos y control de versiones. No se tenía planificado utilizar este sistema, pero un cambio en la política de GitHub nos obligó a buscar otras opciones con versiones gratuitas en las cuales guardar el proyecto y gestionar sus diferentes versiones.

Tras una búsqueda de sistemas similares, nos decantamos por utilizar BitBucket debido a la posibilidad de crear repositorios privados gratuitos e ilimitados para equipos pequeños (menos de 5 personas).

2.2.6. Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones[9].

A lo largo de nuestro paso por los estudios de grado en Ingeniería del Software, hemos utilizado este programa para desarrollar código, razón por la cual lo hemos escogido para la realización de la parte Backend del proyecto.

2.2.7. Overleaf

Overleaf es un gestor online de proyectos escritos en L^AT_EX. Gracias a los cursos formativos impartidos por la Oficina de Software Libre[10] realizado en años anteriores y a los recursos puestos a disposición de los alumnos[11] nos animamos a realizar la memoria con este editor online y en lenguaje de maquetación L^AT_EX.

2.2.8. MobaXterm

MobaXterm es una herramienta muy versátil, entre sus funciones destacan la emulación de terminales o la conexión a un cliente SSH.

En este proyecto lo utilizamos con el objetivo de conectarse al servidor remoto proporcionado por Hostinger[8], en el cual albergamos el despliegue de la parte BackEnd de nuestro proyecto.

2.3. Frameworks

2.3.1. Java Spring

Spring es un framework del lenguaje de programacion java el cual nos permite desarrollar aplicaciones de manera más rápida, eficaz y corta, saltándose tareas repetitivas y ahorrando líneas de código[12].

Sus dos características principales son la inversión de control y la inyección de dependencia. Además de esto da soporte a una gran cantidad de frameworks a través de dependencias, facilitando la tarea del programador.

Hemos escogido este framework para realizar la parte BackEnd de nuestra aplicación por sus características citadas anteriormente, construyendo una API-REST. Esta API-REST actúa como un servidor, siendo consumida por un cliente.

Spring da soporte a una gran cantidad de herramientas a través de dependencias. Dichas dependencias se gestionan mediante Maven, una herramienta open-source que simplifica los procesos de build.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Figura 2.1: Instalación Maven en Java Spring

Swagger

Swagger es un proyecto open source para describir, producir, consumir y visualizar API's REST. Para este proyecto vamos a utilizarlo con código Java, con el objetivo de generar la documentación de los endpoints y de las entidades que se envían y se reciben.

Mockito

Mockito es un framework utilizado junto con JUnit en Java para realizar tests de manera sencilla. En este proyecto se va a utilizar para testear las capas de *Business* y *Controller*, con el objetivo final de conseguir un 100 % de cobertura en las clases testeadas.

JPA

JPA(Java Persistence API)), es una especificación de java para acceder, persistir y manejar datos entre Clases-Objetos de Java y bases de datos relacionales[13].

Gracias a los conocimientos de JPA adquiridos y desarrollados en las asignaturas de IS(Ingeniería del Software) y MS(Modelado de Software), nos decantamos por esta opción para gestionar las llamadas a las bases de datos correspondientes.

2.3.2. Angular

Angular es un framework de desarrollo de aplicaciones SPA(Single Page Applications), el cual utiliza Typescript como lenguaje de programación. Este framework tiene la ventaja de que no refresca el navegador al modificar los elementos de la página web, dando una sensación de dinamismo y de inmersión al usuario.

Nos decantamos por utilizar este framework debido a su popularidad y al gran uso que se le da a nivel empresarial.

Bootstrap

Bootstrap es una herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño[14].

Escogimos esta herramienta ya que nos proporciona resultados óptimos y limpios de manera rápida, además de que posee una documentación muy extensa a la que acudir en caso de duda[15].

CAPÍTULO 3

Desarrollo

En este capítulo se relata el proceso mes a mes que fue el desarrollo de esta aplicación web. Primero veremos la planificación inicial hablando de los diferentes artefactos que se generaron, posteriormente de como estos evolucionaron y se fueron adaptando y por ultimo del resultado final del desarrollo y lo aprendido en el camino,

3.1. Arranque del desarrollo

El primer artefacto que se generó es el Product Backlog, que recoge todas las funcionalidades deseadas por el cliente, en este caso el equipo médico, en la aplicación en forma de historias de usuario (pequeños fragmentos de texto estructurado que intentan clarificar al máximo las mismas). Este se compone de una tabla con tres columnas; la primera de ellas recoge el grupo en el que se engloba la historia de usuario y su ID, la segunda una descripción corta de la misma y la última los criterios del comportamiento que debe tener la funcionalidad en los diversos escenarios posibles.

La generación de este artefacto fue compleja pues los médicos tenían muy claras dos o tres funcionalidades que supondrán el núcleo de la aplicación como veremos posteriormente, pero del resto de la aplicación o no tenían ninguna petición o no había una opinión firme sobre ellas. Por ellos muchas de las funcionalidades de mediana o baja importancia fueron propuestas por el equipo de desarrollo o el director del TFG y el equipo médico se mostró más que contento de acogerlas prácticamente todas. A continuación se relatara y explicara cada sección de dicho Product Backlog y las modificaciones que sufrió, el documento completo en pdf para su cómoda lectura va adjunto a esta memoria.

El núcleo del que hablábamos anteriormente, y por tanto lo primero a remarcar en nuestro Product Backlog, se compone de estas tres funcionalidades:

HISTORIA DE USUARIO ID	DESCRIPCIÓN	CRITERIOS DE ACEPTACIÓN
TestMédico_RellenarDatosPaciente	Como investigador quiero llenar los campos del formulario con datos de un paciente para poder compararlos con otros a futuro y extraer conclusiones.	<ul style="list-style-type: none"> • Cuando el investigador rellene los campos del formulario, si ha rellenado todos los campos, entonces podrá pulsar en "Guardar" para confirmar y almacenar los datos. • Cuando el investigador rellene los campos del formulario, si ha rellenado los campos con valores pero alguno está en blanco, entonces podrá pulsar en "Guardar" pero se le impedirá confirmar el guardado indicándole un error.
TestMédico_ExtraerDatosTests	Como investigador quiero descargar un excell con todos los datos de los test almacenados para poder almacenarlos y compartirlos en el formato que me es más cómodo.	<ul style="list-style-type: none"> • Cuando el investigador pulse "descargar excell" desde su vista principal, si ha rellenado ya al menos un test y lo a guardado, entonces se le descargara un excell con una fila por paciente con la menos un test y ambos test uno consecutivo al otro en la misma linea. • Cuando el investigador pulse "descargar excell" desde su vista principal, si no ha rellenado ya al menos un test y lo a guardado, entonces el botón estará inhabilitado y no producirá ningún efecto además de verse más claro y sin relieve.
Login_AccesoComoInvestigador	Como investigador quiero acceder a la aplicación con mi dni y contraseña para asegurar que esta solo está siendo utilizada por mi y mis compañeros.	<ul style="list-style-type: none"> • Cuando el investigador pulse "descargar excell" desde su vista principal, si ha rellenado ya al menos un test y lo a guardado, entonces el botón estará inhabilitado y no producirá ningún efecto además de verse más claro y sin relieve. • Cuando el investigador pulse "login" desde la vista inicial de acceso a la aplicación, si ha rellenado los campos con su DNI y contraseña correctos, entonces se le indicará con un mensaje de éxito que el login ha sido exitoso. • Cuando el investigador pulse "login" desde la vista inicial de acceso a la aplicación, si no ha rellenado los campos con su DNI y contraseña correctos, entonces se le indicará con un mensaje de error que alguno de los dos campos es incorrecto.

Figura 3.1: Historias de usuario de mayor importancia, Walking Skeleton de la aplicación.

Como se puede apreciar en las historias de usuario lo principal era tener unos test donde llenar los datos de los pacientes de forma cómoda, que estos datos pudieran luego ser extraídos en un excel y que el acceso a la aplicación fuese privado a los miembros del equipo médico. Las dos primeras suponen la entrada y salida más básica de datos que permitiría al equipo médico extraer sus conclusiones y la tercera, a pesar de ser solo el login podría decirse, el acceso a la aplicación, es también de gran importancia. Uno de los requisitos principales del estudio es el control de la procedencia y gestión de los datos. Únicamente pacientes que se hayan ofrecido voluntariamente vía formulario pueden tener sus datos representados y solo los miembros oficiales del estudio pueden tener acceso a ellos. Esto se consigue mediante un login cerrado que no permite ningún tipo de registro desde el exterior y la creación de todos los perfiles de investigador manualmente por el administrador.

El siguiente bloque de funcionalidades complementa las tres primeras que en el primer prototipo emplearían perfiles introducidos a mano en la base de datos para probarse:

AdministrarPacientes_RegistrarPaciente	Como investigador quiero registrar a un paciente en el estudio para poder realizarle test y recopilar sus datos.	<ul style="list-style-type: none"> • Cuando el investigador pulse "registrar" desde su vista principal, si lo introducido en el campo de registro es un número de 8 dígitos, entonces se registrará al paciente y se habilitarán sus test para realizarse. • Cuando el investigador pulse "registrar" desde su vista principal, si lo introducido en el campo de registro no es un número de 8 dígitos, entonces se le indicará que el número de dígitos de la fecha sanitaria del paciente debe ser los últimos 8 dígitos de la fecha sanitaria del paciente.
AdministrarInvestigadores_RegistrarInvestigador	Como administrador quiero registrar manualmente cada investigador para asegurar que tengo el control del acceso a la aplicación.	<ul style="list-style-type: none"> • Cuando el administrador pulse "registrar" desde su vista principal para su perfil en a la aplicación, si ha rellenado los campos todos del formulario que lo acompaña incluyendo un DNI válido y que no estaba registrado previamente, entonces se registrara un nuevo investigador notificándole el éxito de la operación. • Cuando el administrador pulse "registrar" desde la vista principal para su perfil en a la aplicación, si no ha rellenado los campos todos del formulario que lo acompaña o no incluyendo un DNI válido y que no estaba registrado previamente, entonces se le mostrará un mensaje indicándole el error cometido para que lo subsane si lo desea.
Login_AccesoComoAdmin	Como administrador quiero acceder a la aplicación con mi DNI y contraseña para asegurar que solo yo tengo acceso a todas las funcionalidades asociadas a dicho perfil.	<ul style="list-style-type: none"> • Cuando el administrador pulse "login" desde la vista inicial de acceso a la aplicación, si ha rellenado los campos con su DNI y contraseña correctos, entonces se le redirige a su vista principal. • Cuando el administrador pulse "login" desde la vista inicial de acceso a la aplicación, si no ha rellenado los campos con su DNI y contraseña correctos, entonces se le indicará con un mensaje de error que alguno de los dos campos es incorrecto.
AdministrarInvestigadores_ModificarContraseñaInvestigador	Como administrador quiero acceder a la aplicación con mi DNI y contraseña para asegurar que solo yo tengo acceso a todas las funcionalidades asociadas a dicho perfil.	<ul style="list-style-type: none"> • Cuando el administrador pulse "login" desde la vista inicial de acceso a la aplicación, si ha rellenado los campos con su DNI y contraseña correctos, entonces se le redirige a su vista principal. • Cuando el administrador pulse "login" desde la vista inicial de acceso a la aplicación, si no ha rellenado los campos con su DNI y contraseña correctos, entonces se le indicará con un mensaje de error que alguno de los dos campos es incorrecto.

Figura 3.2: Historias de usuario sobre la inclusión del perfil de administrador y los diversos registros.

En este bloque se recogían los registros tanto de pacientes como investigadores así como el perfil de administrador y una funcionalidad que quizás sorprenda encontrarse tan pronto en la escala de prioridades, la modificación de contraseñas. Aparentemente por lo explicado de parte del equipo médico es un problema usual la perdida de contraseña o el filtrado de las mismas por las condiciones de su ambiente de trabajo, y tener un acceso cómodo a su modificación es de gran importancia para ellos. Los desarrolladores lo veían mas como algo secundario pero en este artefacto manda el cliente y así es como se recogió.

Las funcionalidades que prosiguieron en la escala de valor para el cliente fueron las siguientes:

AdministrarInvestigadores_ListarInvestigadores	Como administrador quiero listar todos los investigadores registrados en la aplicación para poder visualizarlos y modificarlos.	<ul style="list-style-type: none"> • Cuando el administrador navegue a su página principal de investigadores, si existe al menos un investigador registrado, entonces se le mostrará una lista con todos los investigadores registrados mostrando su DNI, nombre y género. • Cuando el administrador navegue a su página principal de investigadores, si no existe al menos un investigador registrado, entonces se le mostrará una lista vacía y se le indicará que aún no hay investigadores registrados.
Perfiles_ModificarContraseña	Como investigador/administrador quiero poder modificar mi contraseña para colocar una que me sea más fácil de recordar y solo yo conozca por seguridad.	<ul style="list-style-type: none"> • Cuando el investigador/administrador pulse el icono "Modificar contraseña" desde la vista de su perfil, si ha rellenado todos los campos, su antigua contraseña coincide con la que tenía y la nueva es válida, entonces se le pide confirmación y se realiza el cambio de contraseña de ser afirmativa. • Cuando el investigador/administrador pulse "Modificar contraseña" desde la vista de su perfil, si no ha rellenado todos los campos, o su antigua contraseña no coincide con la que tenía o la nueva no es válida, entonces se le comunica el error mediante un mensaje para que lo subsane si lo deseas.
TestMédico_VisualizarDatosTest	Como investigador quiero visualizar todos los campos de un test ya guardado para comprobar que sean correctos y poder recordarlos cuando quiera.	<ul style="list-style-type: none"> • Cuando el investigador pulse el icono de "visualizar" del test de una cita en su tabla de citas completadas, si ha rellenado esa cita anteriormente, entonces se le redirigirá a una página con todos los campos de la cita acompañados de su valor en la cual no podrá modificar nada. • Cuando el investigador pulse el icono de "visualizar" del test de una cita en su tabla de citas completadas, si no ha rellenado esa cita anteriormente, entonces el botón le redirigirá en su lugar al formulario para la realización de dicha cita.
TestMédico_ModalesDeConfirmaciónyErrores	Como investigador quiero obtener feedback mediante modales al intentar guardar un test para saber en que me he equivocado o no guardar sin estar seguro de quererlo.	<ul style="list-style-type: none"> • Cuando el investigador pulse el icono de "guardar" del test que está realizando, si ha rellenado todos los campos con datos de los tipos esperados, entonces se mostrará un modal pidiéndole confirmación del guardado. • Cuando el investigador pulse el icono de "guardar" del test que está realizando, si ha rellenado todos los campos con datos de los tipos esperados, entonces se mostrará un modal indicándole que hay errores de tipología o algún campo vacío, referenciéndole el primero que se detecte. • Cuando el administrador pulse el icono de "visualizar" en un test realizado dentro de su vista de citas, si dicho test está completado, entonces le redirigirá a una vista con los datos previos del test y campos en los que pueda introducir valores nuevos para todos, alguno o ningún campo que luego para guardar. • Cuando el administrador pulse el icono de "modificar" en un test realizado dentro de su vista de citas, si dicho test no está completado, entonces no aparecerá ninguna opción en la lista. • Cuando el administrador navegue a su página de pacientes, si existe al menos un paciente registrado, entonces se le mostrará una lista con todos los pacientes registrados mostrando su identificador. • Cuando el administrador navegue a su página de pacientes, si no existe al menos un paciente registrado, entonces se le mostrará una lista vacía y se le indicará que aún no hay pacientes registrados. • Cuando el administrador navegue a su página de citas, si existe al menos una cita registrada, entonces se le mostrará una lista con todas las citas realizadas mostrando su identificador de la cita, paciente e investigador implicados. • Cuando el administrador navegue a su página de citas, si no existe al menos una cita registrada, entonces se le mostrará una lista vacía indicándole que aún no se ha realizado el test para ninguna cita.
AdministrarPacientes_ModificarCitaRealizada	Como administrador quiero modificar los campos del test de una cita ya realizada para subsanar errores que se hayan cometido en su creación.	<ul style="list-style-type: none"> • Cuando el administrador pulse el icono de "modificar" en un test realizado dentro de su vista de citas, si dicho test está completado, entonces le redirigirá a una vista con los datos previos del test y campos en los que pueda introducir valores nuevos para todos, alguno o ningún campo que luego para guardar. • Cuando el administrador pulse el icono de "modificar" en un test realizado dentro de su vista de citas, si dicho test no está completado, entonces no aparecerá ninguna opción en la lista. • Cuando el administrador navegue a su página de pacientes, si existe al menos un paciente registrado, entonces se le mostrará una lista con todos los pacientes registrados mostrando su identificador. • Cuando el administrador navegue a su página de pacientes, si no existe al menos un paciente registrado, entonces se le mostrará una lista vacía y se le indicará que aún no hay pacientes registrados. • Cuando el administrador navegue a su página de citas, si existe al menos una cita registrada, entonces se le mostrará una lista con todas las citas realizadas mostrando su identificador de la cita, paciente e investigador implicados. • Cuando el administrador navegue a su página de citas, si no existe al menos una cita registrada, entonces se le mostrará una lista vacía indicándole que aún no se ha realizado el test para ninguna cita.
AdministrarPacientes_ListarPacientes	Como administrador quiero listar todos los pacientes registrados en la aplicación para poder visualizarlos y modificarlos.	<ul style="list-style-type: none"> • Cuando el administrador navegue a su página de pacientes, si existe al menos un paciente registrado, entonces se le mostrará una lista con todos los pacientes registrados mostrando su identificador. • Cuando el administrador navegue a su página de pacientes, si no existe al menos un paciente registrado, entonces se le mostrará una lista vacía y se le indicará que aún no hay pacientes registrados.
AdministrarPacientes_ListarCitasRealizadas	Como administrador quiero listar todas las citas realizadas en la aplicación para poder visualizarlas y modificarlas.	<ul style="list-style-type: none"> • Cuando el administrador navegue a su página de citas, si existe al menos una cita registrada, entonces se le mostrará una lista con todas las citas realizadas mostrando su identificador de la cita, paciente e investigador implicados. • Cuando el administrador navegue a su página de citas, si no existe al menos una cita registrada, entonces se le mostrará una lista vacía indicándole que aún no se ha realizado el test para ninguna cita.

Figura 3.3: Historias de usuario sobre listado de usuarios y diversas modificaciones.

Estas funcionalidades se componen de dos tablas que listen de forma limpia tanto investigadores para el perfil de administrador como pacientes para el de investigador. La funcionalidad de modificar contraseña de nuevo aparece, pero esta vez es para que cada investigador pueda cambiar su propia clave. Esta funcionalidad es principalmente para que el administrador pueda crear sus perfiles con contraseñas que ellos mismos puedan modificar al obtener las cuentas a algo fácil de recordar para ellos mismos. Además se añade la posibilidad de visualizar es una plantilla los datos de un test realizado, ahorrando tener que extraer el excel y consultarla cada vez cuando solo se quiere revisar el test de una cita en concreto. Se añaden modales para indicar errores y confirmación en los test, importante ya que estos no pueden ser modificados una vez guardados a no ser que lo haga el administrador manualmente. Esta posibilidad de modificar una cita por el administrador es la ultima funcionalidad no comentada de este bloque junto con su listar particular para facilitar el trabajo de este.

El penúltimo bloque de funcionalidades que se extrajo al principio del desarrollo empieza a corresponder ya a elementos de gestión necesarios pero no prioritarios que en su mayoría fueron propuestos por el equipo de desarrollo como posibles añadidos prácticos:

AdministrarInvestigadores_EliminarInvestigador	Como administrador quiero eliminar a un investigador para privarle de acceso a la aplicación.	<ul style="list-style-type: none"> Cuando el administrador pulse el icono de "eliminar" en un investigador dentro de su vista de investigadores, si dicho investigador existe y no tiene pacientes asociados, entonces se le pedirá confirmación de su intención y en caso afirmativo se eliminará al investigador. Cuando el administrador pulse el icono de "eliminar" en un investigador dentro de su vista de investigadores, si dicho investigador no existe, entonces no aparecerá siquiera en la lista. Cuando el administrador pulse el icono de "eliminar" en un investigador dentro de su vista de investigadores, si dicho investigador tiene pacientes asignados, entonces no indicara en un mensaje de error que no se pueden eliminar investigadores con pacientes actualmente a su cargo.
AdministrarInvestigadores_FiltrarInvestigadoresPorDNI	Como administrador quiero filtrar los investigadores existentes en la aplicación por DNI para encontrarlos más fácilmente.	<ul style="list-style-type: none"> Cuando el administrador pulse el icono de "buscar" en su vista principal, si ha rellenado el recuadro adjunto con un DNI válido y que está registrado en la aplicación, entonces le filtrará la lista de investigadores a una única tupla con el DNI buscado. Cuando el administrador pulse el icono de "buscar" en su vista principal, si no ha rellenado el recuadro adjunto con un DNI válido, entonces se le indicará en un mensaje de error que este solo acepta DNI en formato completo. Cuando el administrador pulse el icono de "buscar" en su vista principal, si ha rellenado el recuadro adjunto con un DNI válido pero que no está registrado en la aplicación, mostrara una lista vacía sin resultados, indicando que ningún investigador a sido registrado con dicho DNI.
AdministrarInvestigadores_FiltrarInvestigadoresEnOrdenAlfabético	Como administrador quiero filtrar los investigadores existentes en la aplicación por orden alfabético para encontrarlos más fácilmente.	<ul style="list-style-type: none"> Cuando el administrador pulse el icono de "ascendente" sobre la lista de investigadores en su vista principal, si existen al menos dos investigadores registrados en la aplicación, entonces recolocarán de menor a mayor según su DNI. Cuando el administrador pulse el icono de "descendente" sobre la lista de investigadores en su vista principal, si existen al menos dos investigadores registrados en la aplicación, entonces recolocarán de mayor a menor según su DNI. Cuando el administrador pulse el icono de "descendente" o "ascendente" sobre la lista de investigadores en su vista principal, si no existen al menos dos investigadores registrados en la aplicación, entonces la lista de investigadores no sufrirá ninguna modificación.
AdministrarPacientes_EliminarPaciente	Como administrador quiero eliminar a un paciente para sacarle del estudio médico.	<ul style="list-style-type: none"> Cuando el administrador pulse el icono de "eliminar" en un paciente dentro de su vista de pacientes, si dicho paciente existe, entonces se le pedirá confirmación de su intención y en caso afirmativo se eliminará al paciente y sus test asociados. Cuando el administrador pulse el icono de "eliminar" en un paciente dentro de su vista de pacientes, si dicho paciente no existe, entonces no aparecerá siquiera en la lista.

Figura 3.4: Historias de usuario sobre eliminación de perfiles y agilidad en las búsquedas.

Como se puede observar este bloque recogía principalmente la eliminación de perfiles, necesaria si algún paciente o investigador salia del proyecto. Esta funcionalidad aunque simple realmente no se añadió a este artefacto en un primer momento porque generaba dudas por la naturaleza del estudio. Los datos extraídos de la aplicación, para ser validos, deben ser privados, de distintos periodos temporales y completos. La eliminación en este caso de un paciente podía dejar datos incompletos o la de un investigador pacientes con citas a medio completar y cualquiera de estos escenarios invalidaría el estudio. Por ello en primera instancia incluso se pensó en no permitir ningún tipo de eliminación, pero tras debatir con el equipo médico este mismo aseguro que las eliminaciones serian pocas o prácticamente nulas y solo se usarían para subsanar erratas a la hora de crear perfiles nuevos sin afectar los datos. Tras ello se decidió añadirlas no sin delimitar claramente en sus criterios de aceptación las condiciones para poder realizar las eliminaciones.

La otra funcionalidad que apareció fueron los filtrados, los cuales habían sido implementados en otras aplicaciones por el equipo de desarrollo así que no suponían un gran esfuerzo y podían ser moderadamente útiles. Los médicos no se mostraron entusiastas con la idea pues apenas habría 15 investigadores pero ciertamente era una funcionalidad que podrían usar eventualmente así que se decidió añadirla.

El ultimo grupo de funcionalidades que cerraba este artefacto eran las menos prioritarias. Entre ellas un filtrado también para pacientes, la muestra de datos para cada investigador en su propio perfil para los olvidadizos y la inclusión de estadísticas. Esta ultima funcionalidad lamentablemente nunca pudo llevarse al desarrollo pues se prefirió cerrar la aplicación para poder tenerla funcionando y ajustarla a las necesidades del equipo médico a embarcarse en esta funcionalidad que nunca despertó mucho interés en el equipo:

AdministradorPacientes_FiltrarPacientesPorIdentificador	Como administrador quiero filtrar los pacientes por identificador para encontrarlos más rápido.	<ul style="list-style-type: none"> Cuando el administrador pulse el icono de "buscar" en su vista de pacientes, si ha rellenado el recuadro adjunto con un identificador de paciente válido y que está registrado en la aplicación, entonces le filtrará la lista de pacientes a una única tupla con el identificador buscado. Cuando el administrador pulse el icono de "buscar" en su vista de pacientes, si no ha rellenado el recuadro adjunto con un identificador de paciente válido, entonces se le indicará en un mensaje de error que este solo acepta identificadores de pacientes formados por los últimos 8 dígitos de sus tarjetas sanitarias. Cuando el administrador pulse el icono de "buscar" en su vista de pacientes, si no está registrado en la aplicación, entonces mostrará una lista vacía sin resultados indicando que ningún paciente a sido registrado con dicho identificador. Cuando el investigador navegue a su pestaña de perfil, si no ha habido ningún problema con su registro, entonces mostrará una ventana con los datos de su cuenta y la opción de modificar su contraseña si lo desea. Cuando el investigador navegue a su pestaña de perfil, si ha habido ningún problema con su registro, entonces ni siquiera tendrá acceso a la navegación en la aplicación pues no podrá hacer login y deberá ponerte en contacto con el administrador. Cuando el investigador navegue a su pestaña de estadísticas, si ha realizado al menos un test, entonces se le mostrará una gráfica que engloba las incidencias totales de cada campo entre todos sus test. Cuando el investigador navegue a su pestaña de estadísticas, si no ha realizado al menos un test, entonces se le mostrará un mensaje indicando que aún no ha realizado ningún test así que no hay datos con los que alimentar las gráficas.
Perfiles_PerfilInvestigadorConDatos	Como investigador quiero ver los datos de mi perfil para comprobar que son correctos y recordar como los rellene.	
Estadisticas_EstadisticasGlobalesTestRealizados	Como investigador quiero unas graficas que engloben los resultados de todos los test que he realizado para poder tener una referencia visual del estado actual de sus datos.	

Figura 3.5: Historias de usuario sobre filtrado de pacientes, perfiles de investigador y estadísticas.

Estas fueron todas las funcionalidades añadidas al Product Backlog al principio del desarrollo. Junto a este artefacto se creo también un User Story Map, que permite visualizar mejor todas las historias comentadas anteriormente y además marca un flujo de uso de las mismas de izquierda a derecha y secciona las funcionalidades en las releases mensuales en las que esperábamos tenerlas:

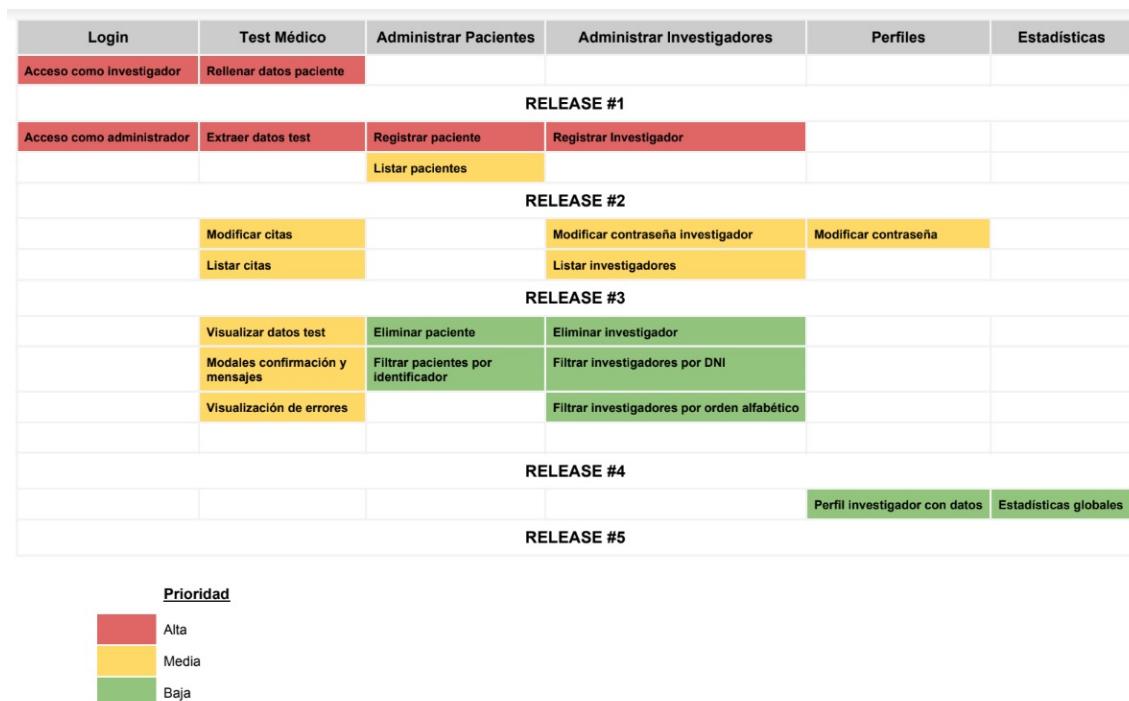


Figura 3.6: Mapa de historias de usuario.

3.2. Desarrollo durante los sprints

Con nuestros artefactos listos y la primera reunión fijada para principios de Octubre comenzó nuestro primer sprint en Septiembre. Nuestro objetivo como se puede apreciar en el User Story Map anterior era el acceso a la aplicación como investigador y la posibilidad de llenar test de pacientes. La idea era empezar con el modelo de datos de los test y de paso pensar en su extracción y después hacer el login, pero faltaban detalles sobre la estructura y limitaciones de los test así que intercambiamos el orden a la espera de información mas precisa. Visto lo cual comenzamos con el acceso a la aplicación.

A pesar de haber estado durante el mes de agosto viendo tutoriales y trasteando con las tecnologías al ponernos de verdad sobre el proyecto vimos lo verdes que estábamos. Eduardo por suerte en el trabajo que había comenzado ese mismo verano había tocado parte de las herramientas aunque no fuese en profundidad, pero Sergio no había oido casi hablar de herramientas como Angular o Spring Boot hasta hace un par de semanas. El primer mes se nos escapo de las manos prácticamente en configuración del software necesario y de los proyectos tanto en front como back, en la conexión de todo el sistema y el diseño de la simple pagina de login. Para cuando llego Octubre el consumo de tareas iba mucho mas lento de lo esperado y la primera reunión prácticamente solo sirvió para cerrar el diseño de la aplicación en lineas generales, la paleta de colores,el tipo de letra, el espaciado y tamaño, etc. Este retraso en las tareas y desajuste del orden de desarrollo se mantendría lamentablemente para el siguiente sprint aunque se iría ajustando en los siguientes. A continuación se recoge una captura del login, el cual inicialmente tenia una imagen distinta de fondo pero con la misma temática de tener al equipo médico en su ambiente de trabajo, los mismo colores claros entre blanco y azul y la misma letras negra grande.

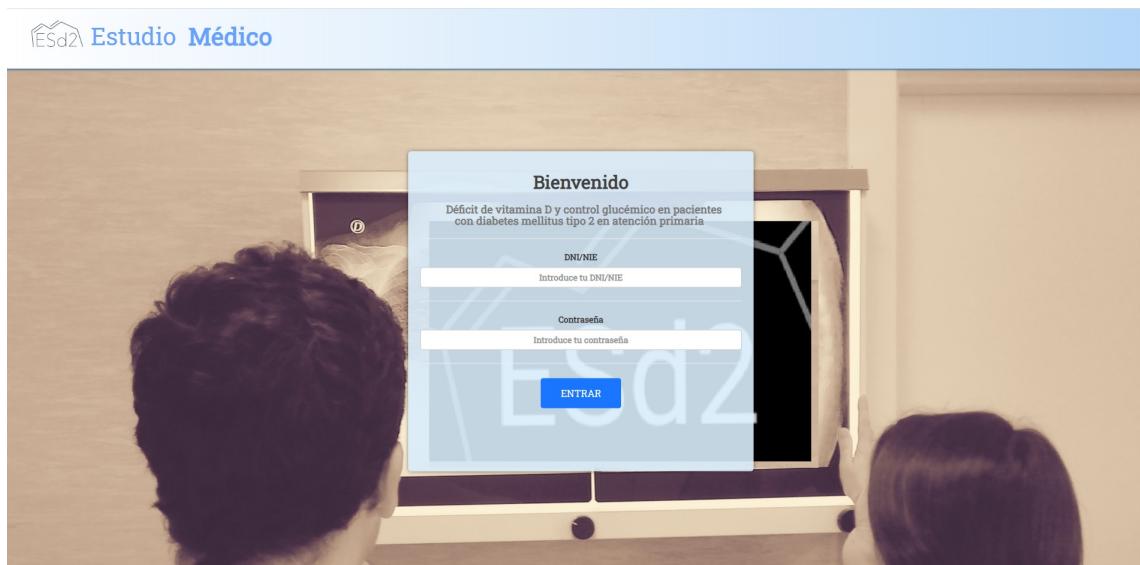


Figura 3.7: Captura del login de la aplicación.

CAPÍTULO 4

Implementación

En este capítulo se explican los detalles de la implementación de las partes FrontEnd y BackEnd de nuestro proyecto, tanto de la generación del proyecto inicial como de la infraestructura del mismo.

4.1. FrontEnd

En esta sección se va a hablar del proyecto Angular el cual compone la parte FrontEnd de nuestra aplicación. Este framework nos ofrece un desarrollo más rápido con respecto a otros frameworks, ya que se estructura en diferentes componentes web y genera una página web dinámicamente.

4.1.1. Generación Proyecto Angular

Para la generación de un proyecto base en Angular se utilizó *Angular Cli*, un herramienta de línea de comandos que facilita la creación y gestión de proyectos Angular y de sus diferentes componentes.

Angular Cli, al tratarse de una herramienta NodeJS, requiere tenerlo instalado en nuestro sistema operativo. Para ello la primera tarea a realizar es instalar NodeJS a través de su página oficial[?], escogiendo la versión que soporte nuestro sistema operativo.

A continuación, a través del terminal que nos proporciona la herramienta de desarrollo *Visual Studio Code*, ejecutamos el comando `npm install -g @angular/cli@latest`, el cual nos instala *Angular Cli* en la última versión disponible.



Figura 4.1: Versión Angular Cli

Una vez instalada la herramienta de comandos *Angular Cli*, procedemos a crear nuestro proyecto base Angular ejecutando el comando `ng new estudio-medico-tfg2019`, generando un proyecto completo Angular junto con toda su estructura de carpetas.

4.1.2. Arquitectura de la aplicación Angular

4.1.3. Inicialización del proyecto

Local

Remoto

4.2. BackEnd

En esta sección se va a hablar del proyecto Java Spring Boot el cual compone la parte BackEnd de nuestra aplicación. La aplicación es una API-REST, la cual será consumida por el FrontEnd mediante llamadas en protocolo HTTP.

4.2.1. Generación Proyecto Java Spring Boot

Para la generación del proyecto inicial se utilizó Spring Initializr[?], a través de su página web. Escogimos gestionar las dependencias del proyecto con Maven dado el alto grado de familiaridad que poseemos con esta tecnología. A continuación se escogió el lenguaje Java para implementar la aplicación debido a sus características intrínsecas (polimorfismo y herencia). Para finalizar se añadieron los paquetes de Web y JPA, necesarios para gestionar las conexiones externas y para la capa de persistencia.

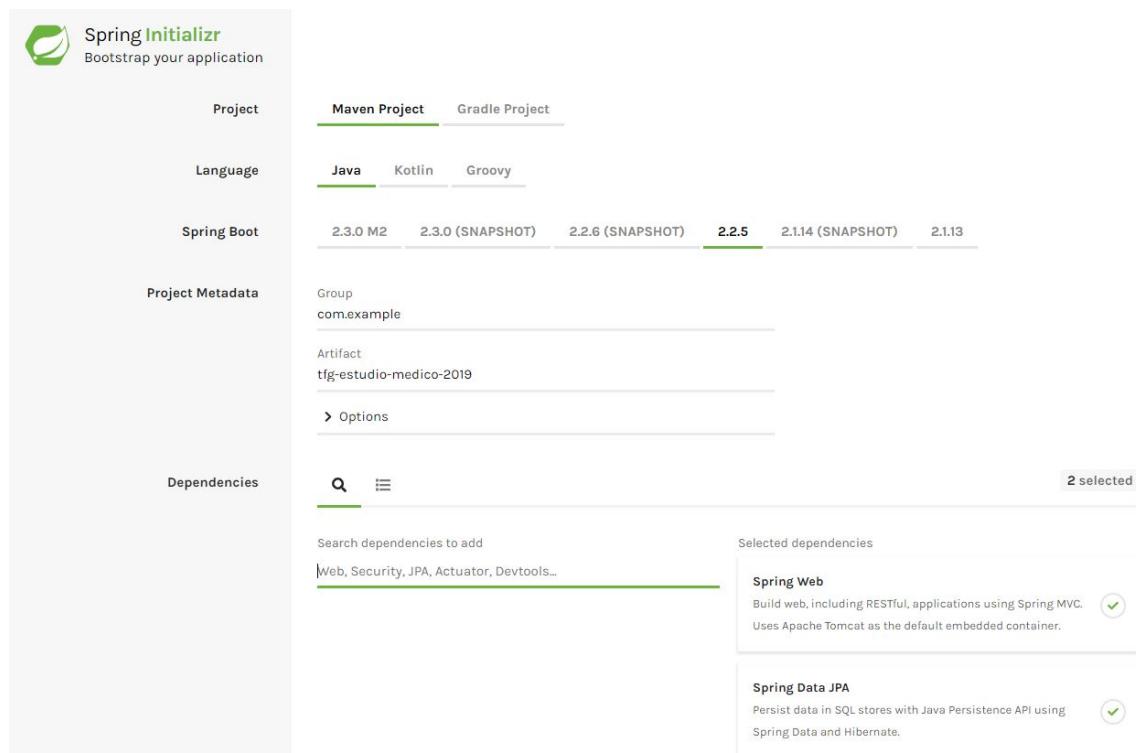


Figura 4.2: Proyecto inicial Spring Boot

Una vez generado el proyecto Spring Boot, hubo de importarse el mismo en el entorno de desarrollo Eclipse.

4.2.2. Arquitectura de la aplicación Java Spring Boot

La arquitectura de la aplicación Spring Boot desarrollada se divide principalmente en los apartados de código, documentación y tests.

Código

El proyecto BackEnd de la aplicación está escrito en Java 8, y dividido en los siguientes bloques o paquetes, los cuales se describen a continuación.

■ Controladores

Son los elementos del código encargados de recibir las diferentes peticiones HTTP y mapear los objetos enviados en las mismas, ya sea en el cuerpo de la petición en caso de ser una petición de tipo POST, o en la propia url si es una petición de tipo GET. Además de esto, son los encargados de realizar la lógica de navegación, llamando a la capa de Negocio y, con el resultado obtenido, generar una respuesta a la parte FrontEnd de la aplicación.

Para identificar los diferentes controladores de nuestra aplicación, la nomenclatura de los mismos siempre terminará con la palabra *Controller*.

En nuestra aplicación disponemos de 3 controladores: El *UserController*, encargado principalmente del proceso de login de la aplicación. El *AdminController*, encargado de obtener todos los recursos que demande el usuario de tipo administrador en nuestra aplicación. Finalmente el *ResearcherController*, encargado de obtener los recursos que demanden los usuarios de tipo investigador en nuestra aplicación.

Todos los controladores se han implementado con el patrón de Diseño *Interface*, separando los métodos de su implementación y haciendo al código más legible y mantenible de cara al futuro.

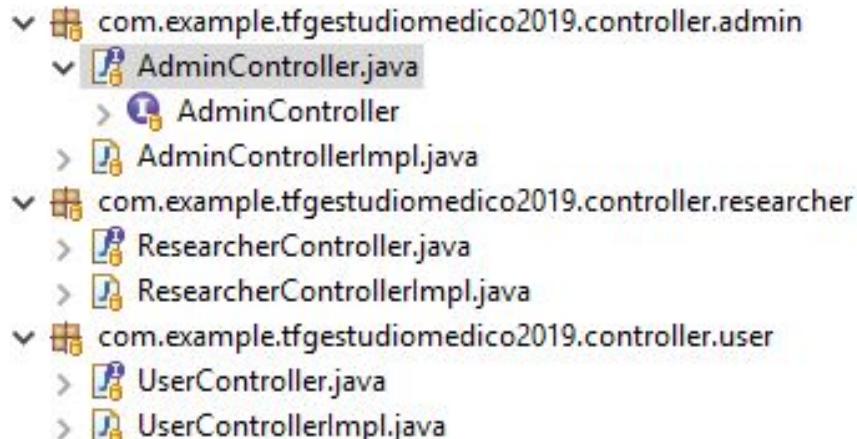


Figura 4.3: Controladores del proyecto BackEnd

```

15 /**
16 * Controller for All API Users.
17 */
18 @RequestMapping("/api/user")
19 public interface UserController {
20
21     @ApiOperation(value = "Login user")
22     @ApiResponses(value = {
23         @ApiResponse(code = 200, message = "Successfully logued"),
24         @ApiResponse(code = 409, message = "Fail login"),
25         @ApiResponse(code = 409, message = "Internal server error")
26     })
27     @PostMapping(path = "/login", produces = "application/json")
28     public ResponseEntity<?> login(@RequestBody UserToLoginDto userToLoginDto);
29 }
30
31

```

Figura 4.4: Ejemplo Interfaz Controlador UserController

- **Servicios de la aplicación**

Son los elementos del código encargados de centralizar la lógica principal de la aplicación. Los controladores los llaman, estos llaman a los repositorios para obtener los recursos solicitados, y devuelven dichos recursos al controlador.

Para identificar los diferentes servicios de de nuestra aplicación, la nomenclatura de los mismos siempre terminará con la palabra *Business*.

En nuestra aplicación disponemos de 3 servicios de aplicación: El *UserBusiness*, encargado principalmente de gestionar a los diferentes usuarios de la aplicación. El *ResearcherBusiness*, encargado de gestionar las diferentes funcionalidades que puede realizar un usuario de tipo investigador en nuestra aplicación. Por último el *SubjectBusiness*, encargado de gestionar a los pacientes y a sus respectivas citas y cuestionarios.

Al igual que los controladores, todos los servicios de aplicación se han implementado con el patrón de diseño *Interface*

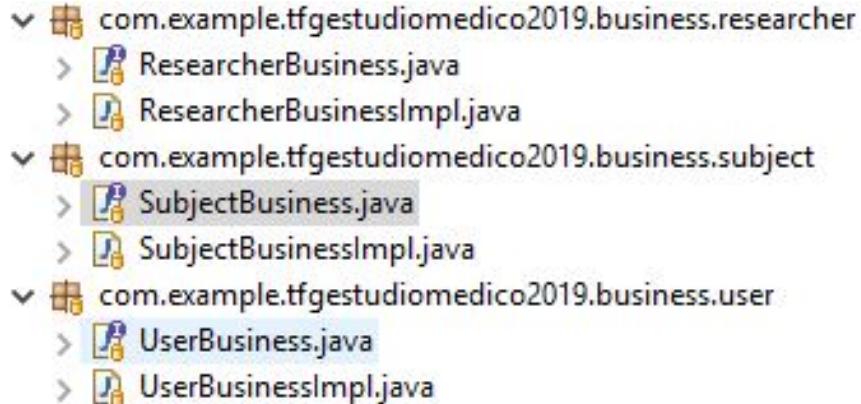


Figura 4.5: Servicios de Aplicación del proyecto BackEnd

```

6
7  /**
8  * User business.
9 */
10 public interface UserBusiness {
11     public UserEntity findByUsernameAndPassword(String username, String password);
12     public UserEntity saveUser(UserEntity user);
13     public UserEntity findByUsername(String username);
14     public UserEntity findById(Integer id);
15     public List<UserEntity> getAllResearchers();
16     public List<UserEntity> getAllAdmins();
17     public boolean deleteResearcher(String username);
18     public UserEntity updateUser(UserEntity user);
19 }
20

```

Figura 4.6: Ejemplo Interfaz Servicio de aplicación UserBusiness

■ **Repositorios**

Son los elementos encargados de comunicarse con la base de datos correspondiente, ya sea guardando recursos, actualizar, listar, borrar etc...

Todos los repositorios extienden de la clase *JpaRepository*, una clase perteneciente a Spring-Data, la cual implementa el código necesario para realizar una query a la base de datos. De esta manera no ha sido necesario implementar dichas clases.

Para identificar los diferentes repositorios de nuestra aplicación, la nomenclatura de los mismos siempre terminará con la palabra *Repository*.

Nuestra aplicación dispone de 4 repositorios: El *InvestigationDetailsRepository*, encargado de gestionar los detalles de los cuestionarios realizados a los pacientes. El *InvestigationRepository*, encargado de gestionar los cuestionarios realizados a los pacientes. El *UserRepository*, encargado de gestionar a los diferentes usuarios administradores e investigadores de la aplicación. Y finalmente, el *SubjectRepository*, encargado de gestionar a los pacientes del estudio.

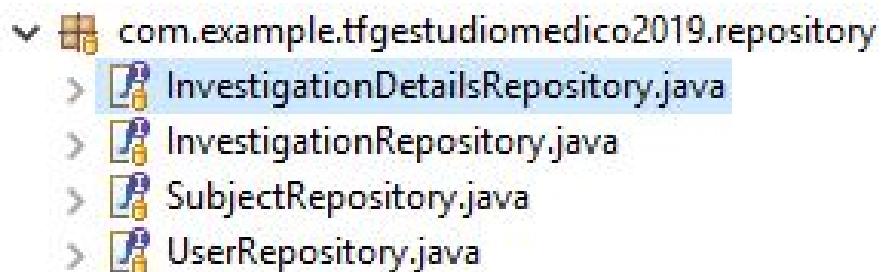


Figura 4.7: Repositorios del proyecto BackEnd

```

10 /**
11  * Interface to generate User queries.
12 */
13 @Repository
14 public interface UserRepository extends JpaRepository<UserEntity, Long> {
15
16     UserEntity findByUsernameAndPassword(String username, String password);
17     UserEntity findByUsername(String username);
18     UserEntity findById(Integer id);
19     List<UserEntity> findByRole(String role);
20     Long deleteByUsername(String username);
21 }

```

Figura 4.8: Interfaz Repositorio UserRepository

- **Modelos**

Son los elementos encargados de almacenar la información de las bases de datos y de las peticiones HTTP, delimitando el dominio de las mismas.

Hay 3 tipos de modelos en nuestra aplicación:

- Dominio.
- Son los elementos que definen un rango de posibles valores. En nuestra aplicación disponemos del enumerado *Role* para definir los tipos de usuarios que pueden acceder a la aplicación.

```

3 /**
4  * Enum to define the API roles.
5 */
6 public enum Role {
7     ADMIN, RESEARCHER
8 }
9

```

Figura 4.9: Ejemplo Dominio Enumerado Role

- Entidades.

Son los elementos encargados de interactuar con las bases de datos de nuestra aplicación. Estas entidades JPA, también llamadas POJO's(Plain Old Java Objects), se encuentran mapeadas mediante anotaciones y representan las tablas de nuestra base de datos relacional.

Para identificar las diferentes entidades de nuestra aplicación, la nomenclatura de las mismas siempre terminará con la palabra *Entity*.

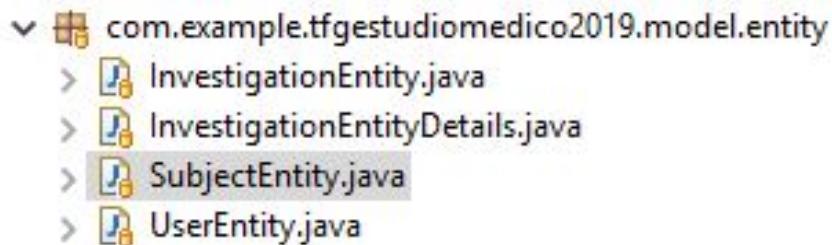


Figura 4.10: Entidades del proyecto BackEnd

```

15 /**
16  * Entity for investigation table.
17 */
18 @Entity
19 @Table(name = "investigation")
20 public class InvestigationEntity {
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Integer id;
25
26     @ManyToOne
27     @JoinColumn(name = "idsubject")
28     private SubjectEntity subject;
29
30     @Column(name = "numberinvestigation")
31     private Integer numberInvestigation;
32
33     @Column(name = "completed")
34     private Boolean completed;
35
36     @OneToOne(cascade = {CascadeType.ALL})
37     @JoinColumn(name="idinvestigationdetails")
38     private InvestigationEntityDetails investigationEntityDetails;
39

```

Figura 4.11: Ejemplo Entidad InvestigationEntity

- Dtos(Data Transfer Objects).

Son los elementos encargados de mapear los objetos de tipo JSON enviados en las peticiones HTTP para ser tratados en la aplicación. También se encargan de devolver la información necesaria mediante una respuesta HTTP.

Para identificar a los dtos(Data Transfer Objects) de nuestra aplicación, la nomenclatura de los mismos siempre terminará con la palabra *Dto*.

De esta manera, los controladores de la aplicación tratarán los dtos y los mapearán a entidades para ser tratadas en la capa de Negocio, proporcionando seguridad y robustez a la aplicación, evitando posibles inyecciones de código y protegiendo los campos sensibles que no deban mostrarse en el exterior.

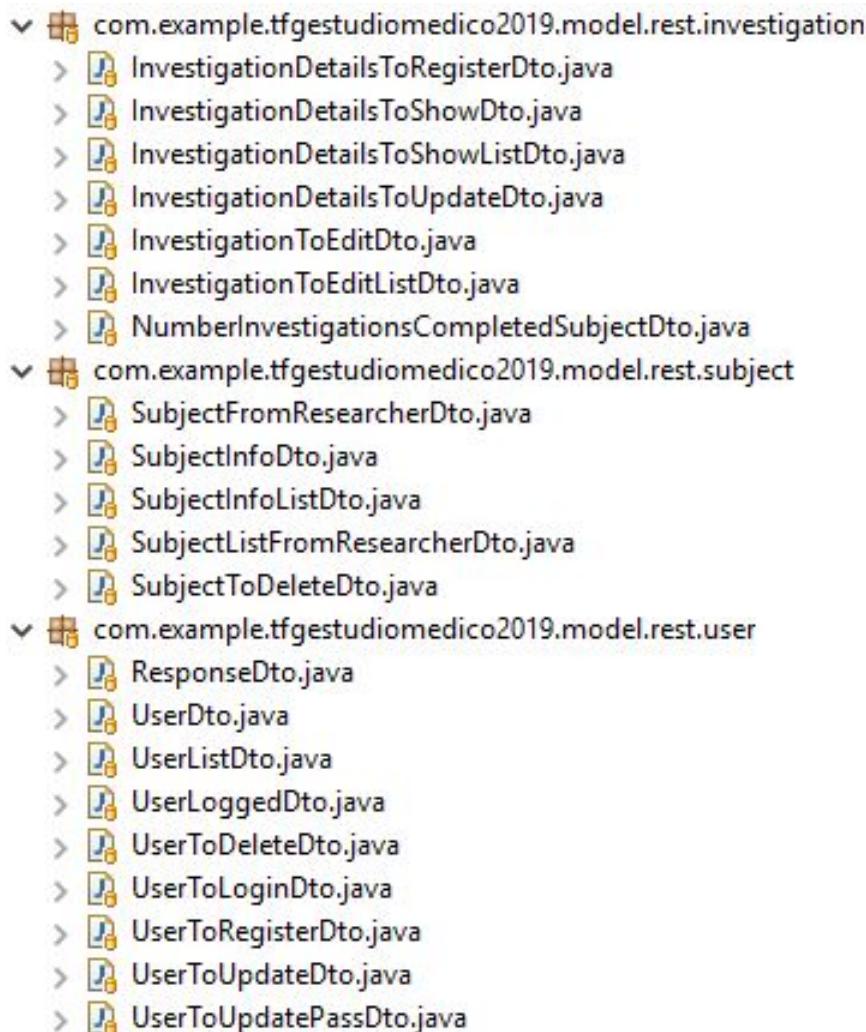


Figura 4.12: Dtos del proyecto BackEnd

```

5  /**
6   * User dto general.
7   *
8   */
9 public class UserDto {
10
11     @ApiModelProperty(value = "The username of the user", example = "12345678A", dataType = "java.lang.String")
12     private String username;
13     @ApiModelProperty(value = "The name of the user", example = "Sergio", dataType = "java.lang.String")
14     private String name;
15     @ApiModelProperty(value = "The gender of the user", example = "hombre", allowableValues="hombre, mujer",
16                 dataType = "java.lang.String")
17     private String gender;
18     @ApiModelProperty(value = "The id of the user", example = "23", dataType = "java.lang.Integer")
19     private Integer id;
20     @ApiModelProperty(value = "The surname of the user", example = "Pacheco Fernández", dataType = "java.lang.String")
21     private String surname;

```

Figura 4.13: Ejemplo dto UserDto

Documentación

A la hora de documentar la aplicación BackEnd, por una parte, se utilizó *Javadoc* al principio de cada clase Java, y por otro lado, se utilizó *Swagger*.

Para utilizar el framework *Swagger* en la aplicación, hubo de añadir las correspondientes dependencias en el archivo *pom.xml*.

```
53
54      <!-- SWAGGER -->
55@      <dependency>
56          <groupId>io.springfox</groupId>
57          <artifactId>springfox-swagger2</artifactId>
58          <version>2.9.2</version>
59      </dependency>
60
61@      <dependency>
62          <groupId>io.springfox</groupId>
63          <artifactId>springfox-swagger-ui</artifactId>
64          <version>2.9.2</version>
65      </dependency>
66
```

Figura 4.14: Dependencias Swagger añadidas en pom.xml

Swagger nos permitió documentar tanto los endpoints como los dtos de nuestra aplicación a través de anotaciones, las cuales se añadieron en las respectivas clases implicadas.

```
54@      @ApiOperation(value = "Register a researcher")
55@      @ApiResponses(value = {
56@          @ApiResponse(code = 201, message = "Successfully user registered", response= UserDto.class),
57@          @ApiResponse(code = 409, message = "User already exists", response= ResponseDto.class),
58@          @ApiResponse(code = 500, message = "Server error", response= ResponseDto.class)
59@      })
60@      @PostMapping(path = "/registerResearcher", produces = "application/json")
61@      public ResponseEntity<?> registerResearcher(@RequestBody UserToRegisterDto user);
```

Figura 4.15: Ejemplo Documentación Swagger en el endpoint Registrar Investigador

Swagger posee una interfaz llamada *swagger.ui* a la cual se puede acceder una vez arrancado el proyecto BakEnd para realizar pruebas y comprobar la documentación generada, como se muestra continuación. Dicha interfaz documenta tanto la información general de la aplicación, como los endpoints y los dtos.



Figura 4.16: Interfaz swagger.ui

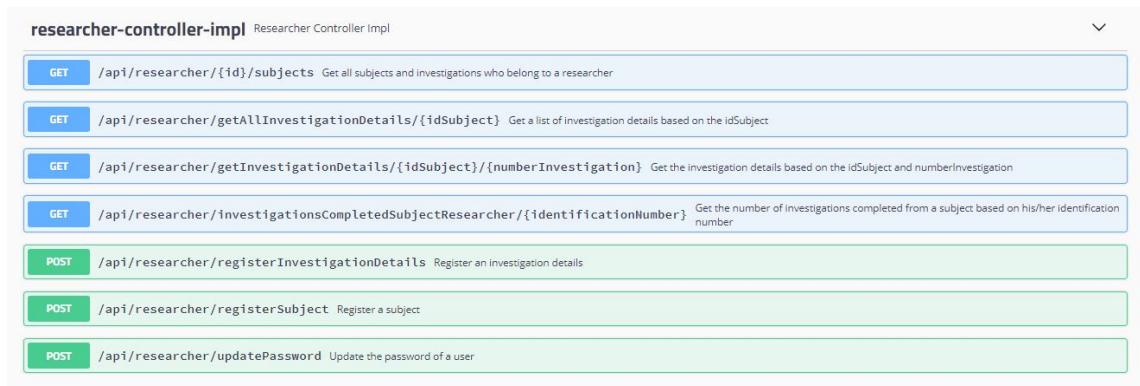


Figura 4.17: Ejemplo Documentación Swagger Researcher Controller

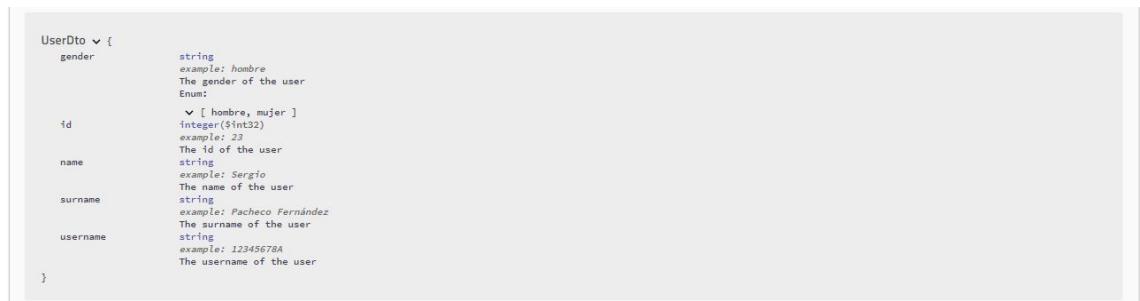


Figura 4.18: Ejemplo Documentación Swagger UserDto

Tests

A la hora de garantizar la calidad del software, se han realizado tests con Mockito y JUnit. Se han testeado todos los métodos de todas las clase que poseen lógica(Controladores y Servicios de Aplicación), probando todos los caminos y excepciones posibles, así como el camino feliz.

En la aplicación hay un total de 157 tests realizados, los cuales se ejecutan cada vez que se instala la aplicación, garantizando así el correcto funcionamiento de la misma.

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running business.researcher.ResearcherBusinessTest
[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.633 s - in business.researcher.ResearcherBusinessTest
[INFO] Running business.subject.SubjectBusinessTest
[INFO] Tests run: 24, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.063 s - in business.subject.SubjectBusinessTest
[INFO] Running business.user.UserBusinessTest
[INFO] Tests run: 17, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.01 s - in business.user.UserBusinessTest
[INFO] Running controller.admin.AdminControllerTest
[INFO] Tests run: 62, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.273 s - in controller.admin.AdminControllerTest
[INFO] Running controller.researcher.ResearcherControllerTest
[INFO] Tests run: 40, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.161 s - in controller.researcher.ResearcherControllerTest
[INFO] Running controller.user.UserControllerTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in controller.user.UserControllerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 157, Failures: 0, Errors: 0, Skipped: 0
```

Figura 4.19: Tests realizados en la aplicación

Para garantizar que se han testeado todos los caminos posibles, se ha conseguido un 100 % de cobertura en todas las clases con tests.

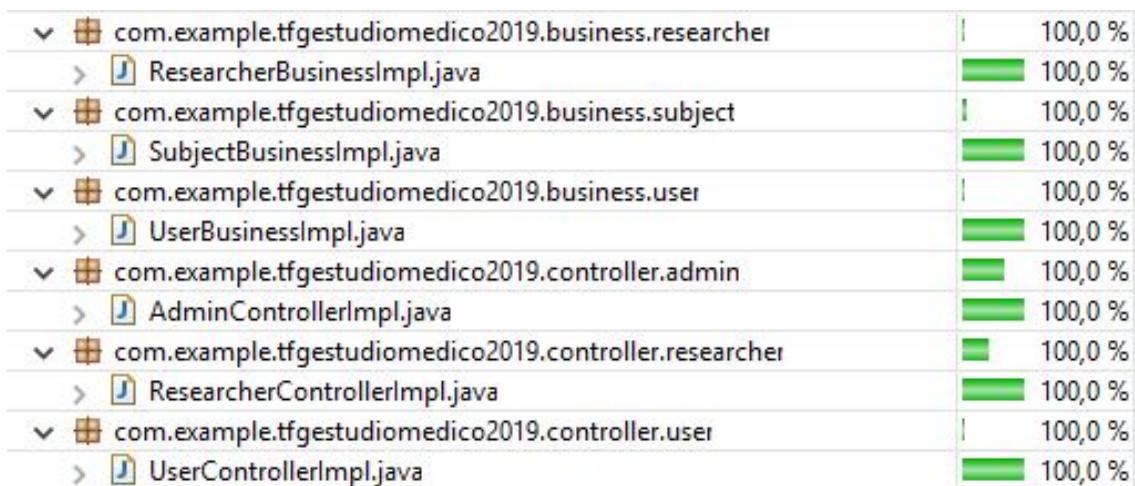


Figura 4.20: Cobertura de las clases testeadas

4.2.3. Inicialización del proyecto

A la hora de inicializar la aplicación BackEnd, al tratarse de un proyecto Maven, lo primero que hay que ejecutar son los siguientes comandos:

1. mvn clean.

Elimina los archivos generados anteriormente y descarga las librerías añadidas como dependencias en el archivo pom.xml

2. mvn install.

Genera el archivo .jar definido en el pom.xml

Si queremos ejecutar la aplicación BakEnd de manera local, lo único que tenemos que hacer es ejecutar la clase **Main** de la aplicación en el entorno de desarrollo elegido, en este caso, Eclipse.

Si queremos ejecutar la aplicación BakEnd de manera remota, solo tenemos que almacenar el archivo .jar generado anteriormente en el servicio de Hosting escogido, en este caso, en Hostinger[8].

CAPÍTULO 5

Despliegue

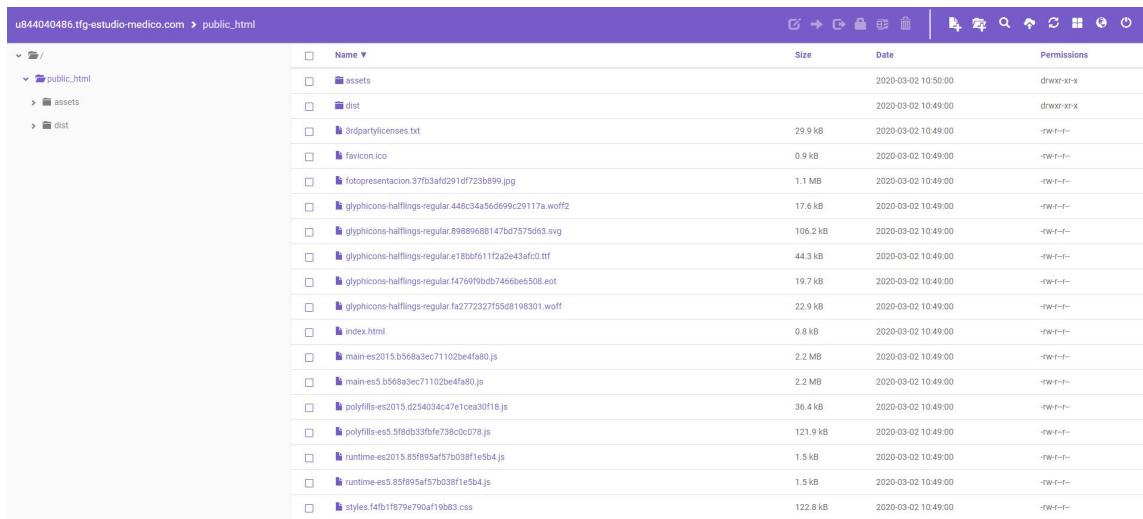
En este capítulo se detalla como, una vez terminada una versión final de la aplicación, se comienza con el proceso de habilitar el uso de la aplicación desde la web. Para ello decidimos utilizar la pataforma Hostinger[8], y todos los servicios que la misma proporciona a sus usuarios.

El despliegue de la aplicación consta de tres partes; el despliegue de la aplicación FrontEnd, el despliegue de la aplicación BackEnd, y el despliegue de la base de datos.

5.1. FrontEnd

Para desplegar la aplicación Angular que representa el FrontEnd de nuestra aplicación, se utilizó la herramienta “Administrador de archivos” que nos ofrece Hostinger y que nos proporciona una interfaz de usuario para administrar archivos y directorios en nuestro dominio *tfg-estudio-medico.com*.

En dicho administrador, se procedió a subir la carpeta *dist* generada por el proyecto Angular.



The screenshot shows a file manager interface with the following details:

Name	Size	Date	Permissions
assets		2020-03-02 10:50:00	drwxr-xr-x
dist		2020-03-02 10:49:00	drwxr-xr-x
3rdpartylicences.txt	29.9 kB	2020-03-02 10:49:00	-rw-r--r--
favicon.ico	0.9 kB	2020-03-02 10:49:00	-rw-r--r--
fotopresentacion.37fb3af2d723b899.jpg	1.1 MB	2020-03-02 10:49:00	-rw-r--r--
glyphicon-halflings-regular.44bc34a56d699c29117a.woff2	17.6 kB	2020-03-02 10:49:00	-rw-r--r--
glyphicon-halflings-regular.698899a88147bd7575d53.svg	106.2 kB	2020-03-02 10:49:00	-rw-r--r--
glyphicon-halflings-regular.e188bf611f2a2e43af0.ttf	44.3 kB	2020-03-02 10:49:00	-rw-r--r--
glyphicon-halflings-regular.f769f90db7466be6508.eot	19.7 kB	2020-03-02 10:49:00	-rw-r--r--
glyphicon-halflings-regular.fa2772327f55d8198301.woff	22.9 kB	2020-03-02 10:49:00	-rw-r--r--
index.html	0.8 kB	2020-03-02 10:49:00	-rw-r--r--
main-es2015.b568a3ec71102be4fa80.js	2.2 MB	2020-03-02 10:49:00	-rw-r--r--
main-es5.b568a3ec71102be4fa80.js	2.2 MB	2020-03-02 10:49:00	-rw-r--r--
polyfills-es2015.d254034c47e1cea30f18.js	36.4 kB	2020-03-02 10:49:00	-rw-r--r--
polyfills-es5.5f86d935fbef739cc0c078.js	121.9 kB	2020-03-02 10:49:00	-rw-r--r--
runtime-es2015.85f095af57b03dfe1e5b4.js	1.5 kB	2020-03-02 10:49:00	-rw-r--r--
runtime-es5.85f095af57b03dfe1e5b4.js	1.5 kB	2020-03-02 10:49:00	-rw-r--r--
styles.44fb1fb79e790a1f9083.css	122.8 kB	2020-03-02 10:49:00	-rw-r--r--

Figura 5.1: Administrador de archivos de Hostinger

5.2. BackEnd

Para desplegar la aplicación Java Spring que constituye el BackEnd de nuestra aplicación, se utilizó un servidor proporcionado por Hostinger[8]. En dicho servidor instalamos un sistema operativo Ubuntu 18.04 64bit.

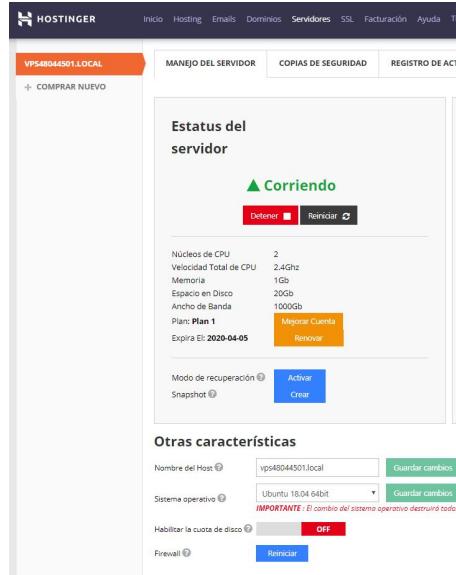


Figura 5.2: Servidor en Hostinger

Para conectarnos al servidor utilizamos el programa MobaXterm, conectándonos al servidor a través de SSH. Nuestro objetivo era mantener el archivo .jar ejecutándose en la máquina incluso si no estamos conectados a la misma, para ello generamos un *servicio*, esto es, un script que se mantiene arrancado incluso si cerramos la sesión.

Lo primero que tenemos que hacer es generar el archivo ejecutable de nuestro proyecto BackEnd. Una vez generado el archivo SNAPSHOT .jar de nuestro proyecto Java Spring, nos disponemos a subirlo a la carpeta /root/back.

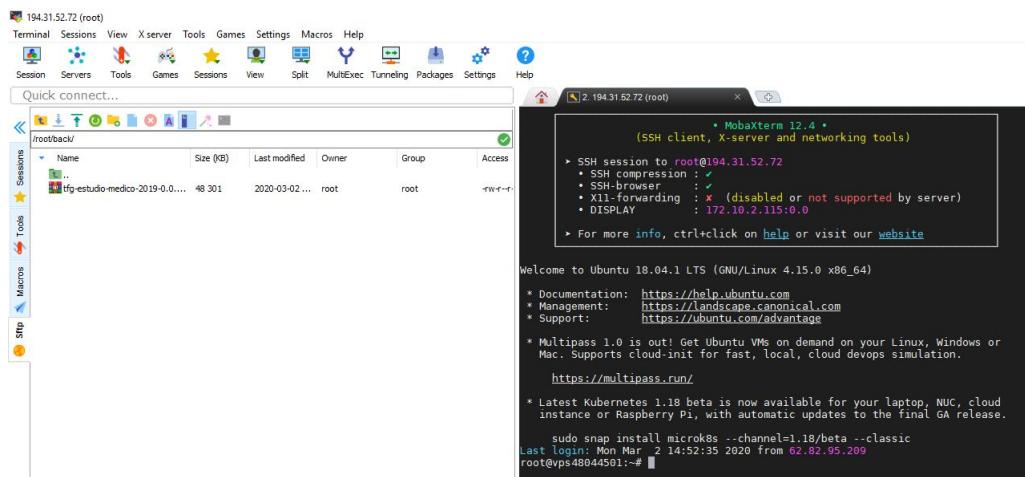


Figura 5.3: Jar desplegado en MobaXterm

A continuación, generamos un script llamado *start_back.sh* que simplemente ejecuta el comando *java -jar* para arrancar el ejecutable .jar, el cual vamos a guardar en el directorio /usr/local/bin. Lo guardamos en dicho directorio debido a que este directorio es accesible por todos los usuarios.

```
java -jar /root/back/tfg-estudio-medico-2019-0.0.1-SNAPSHOT.jar
```

Figura 5.4: Script start_back.sh desplegado en MobaXterm

Después hemos generado un *servicio* llamado *daemonback.service* en la carpeta /etc/systemd/system, que es el directorio indicado en sistemas Linux para almacenar y gestionar demonios. Este servicio se encarga de ejecutar en segundo plano el script *start_back.sh*.

```
[Unit]
Description = Init BACK service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/local/bin/start_back.sh

[Install]
WantedBy=multi-user.target
```

Figura 5.5: Servicio daemonback.service desplegado en MobaXterm

Por último tan solo tenemos que ejecutar el servicio generado previamente con el comando **sudo service daemonback start**. Para ver el estado del .jar, ejecutamos el comando **sudo service daemonback status** y nos muestra lo siguiente:

```
● daemonback.service - Init BACK service
   Loaded: loaded (/etc/systemd/system/daemonback.service; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-03 09:52:11 UTC; 1min 34s ago
     Main PID: 23864 (bash)
       Tasks: 1 (since 09:52:11)
      CGroup: /system.slice/daemonback.service
              └─23864 /bin/bash /usr/local/bin/start_back.sh

Mar 03 09:52:27 vps48044501.local bash[23864]: 2020-03-03 09:52:27,07 INFO 23865 ... [main] o.h.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.0.4.Final}
Mar 03 09:52:30 vps48044501.local bash[23864]: 2020-03-03 09:52:30,03 INFO 23865 ... [main] prg.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Mar 03 09:52:32 vps48044501.local bash[23864]: 2020-03-03 09:52:32,215 INFO 23865 ... [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
Mar 03 09:52:33 vps48044501.local bash[23864]: 2020-03-03 09:52:33,015 WARN 23865 ... [main] webConfigurationsJpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed even if they are not explicitly required. A typical case is an intercepting filter that calls Method#hasMethod (public org.springframework.web.ResponseMethod)
Mar 03 09:52:37 vps48044501.local bash[23864]: 2020-03-03 09:52:37,106 INFO 23865 ... [main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
Mar 03 09:52:37 vps48044501.local bash[23864]: 2020-03-03 09:52:37,209 INFO 23865 ... [main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
Mar 03 09:52:38 vps48044501.local bash[23864]: 2020-03-03 09:52:38,089 INFO 23865 ... [main] d.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
Mar 03 09:52:38 vps48044501.local bash[23864]: 2020-03-03 09:52:38,097 INFO 23865 ... [main] c.e.t.TfgEstudioMedico2019Application : Started TfgEstudioMedico2019Application in 25.691 seconds (JVM running for 27.005)
```

Figura 5.6: Estado servicio

5.3. Base de Datos

Para desplegar la base de datos en la web, se utilizó un servicio que ofrece Hostinger para desplegar bases de datos MySQL. A través de PhpMyAdmin y con la opción de generar la base de datos automáticamente al ejecutar un proyecto Java Spring con JPA, se generó la siguiente base de datos:

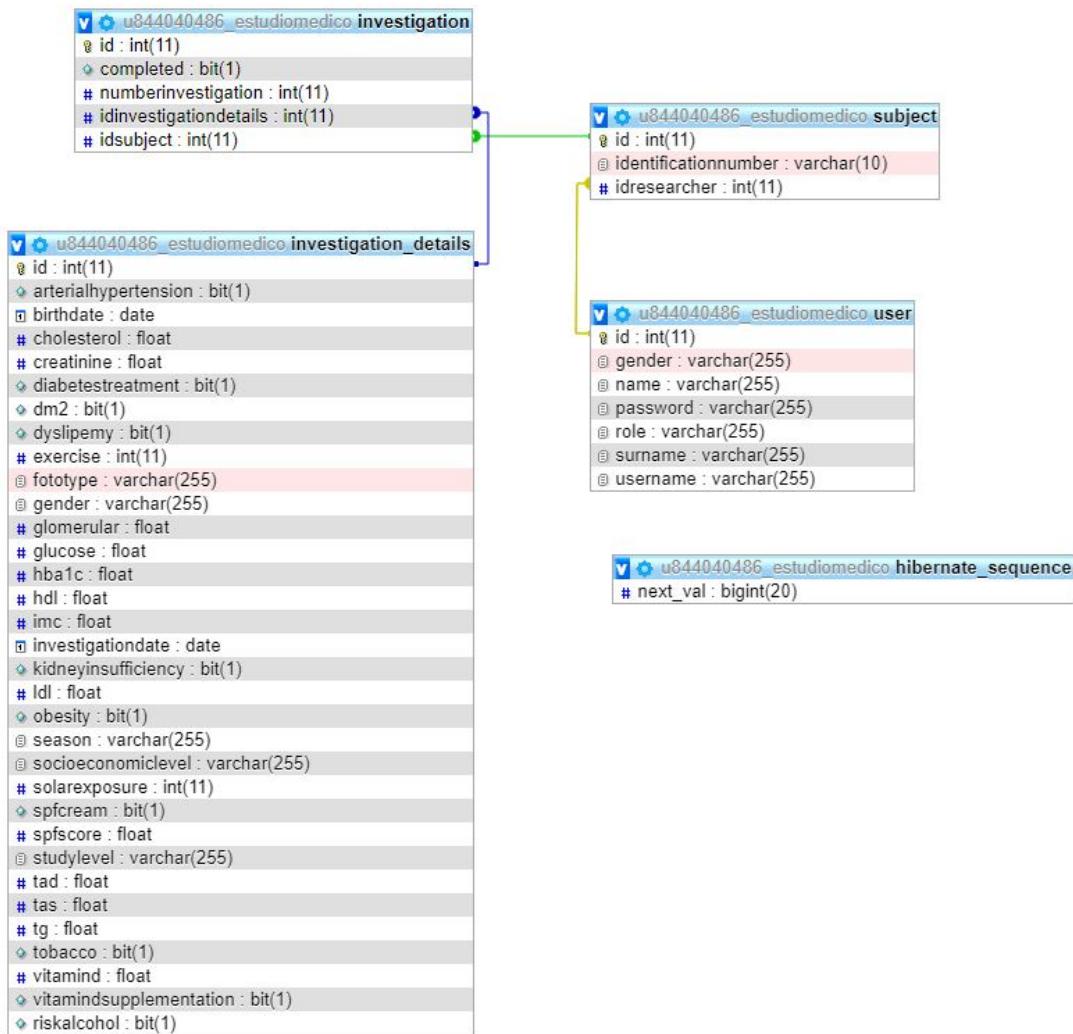


Figura 5.7: Modelo de datos generado en PhpMyAdmin

CAPÍTULO 6

Mantenimiento

En este capítulo se van a desarrollar todas las tareas de mantenimiento realizadas, ya que, una vez desplegada la aplicación, hay que dar soporte a la misma, acercándose de esta manera a como sería un proyecto real.

El mantenimiento es una parte vital del TFG y uno de los motivos por los cuales nos decantamos por escoger el mismo. Al hablar de mantenimiento nos referimos tanto a nuevas funcionalidades previamente no establecidas, corrección de errores y bugs, actualizaciones de funcionalidades ya implementadas, mejoras etc...

Toda aplicación web necesita de tareas de mantenimiento, ya que el mercado y sus necesidades evolucionan de una manera vertiginosa y sino las aplicaciones se quedarían desfasadas.

Con el objetivo de dedicarle el tiempo suficiente y necesario a las tareas de mantenimiento, la aplicación se desplegó lo más pronto posible, a finales de febrero. Una vez desplegada la aplicación, el médico Alejandro Rabanal y su equipo de investigadores nos dieron el feedback necesario para empezar las tareas de mantenimiento.

A continuación se describe una lista de las tareas de mantenimiento solicitadas por el equipo de investigadores, ordenada por orden cronológico.

- **Registro de nuevos administradores.**

La primera tarea a realizar una vez desplegada la aplicación fue dar de alta a usuarios con perfil Administrador, teniendo estos permiso para gestionar la mayoría de elementos de la aplicación.

Se procedió a dar de alta como administradores, con sus respectivos documentos de identificación, a Pablo Rabanal, Alejandro Rabanal, Sergio Pacheco, Eduardo Gonzalo y a otros dos investigadores del estudio más.

De esta manera dichos usuarios podían acceder a la aplicación con su perfil y empezar a probar las funcionalidades de la misma, dando un feedback temprano y preciso.

- **Imposibilidad de registrar investigadores extranjeros.**

Cuando un administrador quiere registrar a un nuevo investigador, a la hora de llenar el formulario e introducir el campo DNI, este no admitía NIEs, de manera que es imposible registrar también a investigadores extranjeros.

Este bug se solucionó añadiendo una nueva validación al campo DNI/NIE para que admitiese ambos formatos, pudiendo registrar a investigadores extranjeros.

- **El administrador no puede exportar los datos de todos los pacientes.**

Se requiere de una nueva funcionalidad que consiste en que el administrador pueda generar un documento de tipo excel que contenga los datos de todos los pacientes de todos los investigadores del estudio.

Esta funcionalidad se implementó sin ningún problema destacable, añadiendo un botón el cual generaba el documento excel descargándolo a través del navegador en el dispositivo del usuario.

- **Falta de información en los mensajes de error de formato.**

A la hora de registrar un investigador, los administradores de nuestro proyecto nos solicitaron una mejora. Al llenar el campo DNI/NIE, si el valor introducido no es correcto y el formato es erróneo, se debe de añadir al mensaje de error mostrado información del formato válido.

La mejora solicitada se implementó correctamente, añadiendo al mensaje de error de formato del campo DNI/NIE información del formato de DNI y del NIE admitidos.

- **Falta de información en los campos de la cita.**

Se nos hizo saber que faltaba información relativa al formato correcto en la gran mayoría de los campos del formulario de llenar cita, tanto indicando el formato del campo como del rango de valores admitidos.

La mejora solicitada se implementó de manera exitosa a través de tooltips o pequeñas etiquetas en los campos, ayudando al usuario a saber como llenar los campos.

- **Cambio de formato en los campos de la cita.**

Tras un tiempo de pruebas, los investigadores nos solicitaron una nueva mejora. Tanto el campo de la cita *Tiempo de exposición solar*, como el campo *Ejercicio Físico*, en un principio, admitían números con decimales. La mejora consistía en que estos dos campos no admitiesen decimales, ya que carecía de sentido que alguien tomase el sol, por ejemplo, 50.3 minutos al día, además de que esto provocaba posibles errores a la hora de registrar una cita.

La mejora se implementó con éxito, cambiando el formato de ambos campos tanto en base de datos, como en BackEnd y FrontEnd, además de cambiar las validaciones para que ajustarse a dicha mejora.

- **Necesidad de nuevas validaciones en los campos de la cita.**

Dado que los campos *Puntuación SPF* y *Uso de crema SPF* están relacionados, se nos solicitó añadir nuevas validaciones a la hora de llenar dichos campos para completar la cita.

La mejora consiste en que si el usuario marca como falso el campo *Uso de crema SPF*, el valor del campo *Puntuación SPF* debe ser cero para que tenga sentido. De la misma manera, si el usuario marca como verdadero el campo *Uso de crema SPF*, el valor del campo *Puntuación SPF* debe ser mayor que cero.

- **Bug encontrado a la hora de generar documento excel.**

Los investigadores se dieron cuenta de que, al generar un documento excel con los datos de las citas de los pacientes, por cada cita se generaba una fila. Esto no era el comportamiento esperado, ya que se especificó que se generase una única fila por paciente conteniendo información de todas sus citas.

Este bug necesitó de bastante trabajo, ya que no resultó fácil el tratamiento de los datos y la generación del documento excel. Al final se realizaron los cambios oportunos de manera exitosa.

- **Falta de criterios de inclusión y exclusión al registrar un paciente.**

Los investigadores nos solicitaron mejorar una funcionalidad ya implementada. A la hora de registrar un nuevo paciente, tras introducir el número de identificación del paciente, debía de implementarse una ventana emergente en la cual el usuario rellenaba una serie de criterios de inclusión y exclusión.

Esta mejora se implementó, requiriendo de bastante tiempo y esfuerzo a la hora de implementarla de manera limpia y estética.

CAPÍTULO 7

Conclusiones

Una vez finalizado el desarrollo y avanzados varios meses en el mantenimiento de la aplicación podemos hacer una valoración de la experiencia y los resultados del proyecto.

En cuanto al resultado obtenido en la aplicación web, tanto el equipo médico como nosotros, los desarrolladores, no podríamos estar más contentos y satisfechos. La aplicación no incluye nada que ellos no necesiten, está simplificada a exactamente las funcionalidades buscadas, agilizando el proceso lo máximo posible, ya que este proyecto de investigación es algo extra a hacer a parte de su trabajo y no puede consumirles mucho tiempo. El acceso a la aplicación es privado al no permitir ningún tipo de registro desde el exterior y los datos de los pacientes están asegurados al ni siquiera poder identificar a cada paciente con una persona real (no almacenamos DNI) y tener los datos de estos restringidos a detalles médicos poco sensibles. Las funcionalidades están muy probadas y durante el mantenimiento nos hemos asegurado de dejarlas lo más intuitivas y con prevención de errores posible.

Por otro lado este proyecto nos ha aportado una buena experiencia en tecnologías que dominábamos poco o nada en muchos casos. Nos ha servido de experiencia profesional con un cliente real y nos ha permitido explorar el despliegue y mantenimiento de una aplicación, un terreno totalmente inexplorado durante la carrera.

Como última conclusión, y realmente la más importante de este proyecto, los médicos han comenzado a obtener los datos que necesitan. Los incidentes derivados del COVID-19 han retrasado sus planes pero para el año que viene deberían poder recabar información tanto de la época invernal como veraniega que precisan y quizás para estas fechas del año que viene estén ante una posible gran aportación a la comunidad de diabéticos, tanto nacional como mundial. No podemos estar más contentos y orgullosos de haber escogido este proyecto y esperamos que sirva en años futuros como base para muchos más.

CAPÍTULO 8

Trabajo Futuro

Finalmente en este apartado nos gustaría resaltar algunas funcionalidades extra que, aunque no son imprescindibles, nos habría gustado tener tiempo de implementar:

- **Gráficas comparativas en detalle.**

La funcionalidad que se quedó en el tintero de mayor importancia. Aunque el equipo médico aseguró en numerosas ocasiones que ellos funcionaban siempre con tablas de excel para el manejo de datos, no cabe duda de que unas gráficas ilustrativas de la información recopilada en el estudio habrían sido útiles. Quizá no aportasen nada nuevo a este proyecto de investigación en concreto pero a futuro podrían permitir descubrir otros patrones en los pacientes o servir de base a investigaciones distintas.

- **Formularios dinámicos.**

Una idea que surgió durante el desarrollo de los últimos detalles de la aplicación y habría ampliado muchísimo su potencial. Básicamente se planteó la idea de poder añadir, eliminar o modificar campos en los formularios de tal forma que la plataforma sería totalmente reutilizable para cualquier proyecto. Esto por supuesto supone un desafío a nivel de modelo de datos, la información pasaría a ser más cómoda almacenada en JSON que en estructuras estáticas, la comunicación entre componentes debería saber adaptarse, etc. En definitiva un trabajo complejo que nos habría encantado realizar.

- **Sistema de notificaciones.**

Por último actualmente la comunicación entre investigadores y administradores se realiza de forma externa a la aplicación. Esto actualmente no supone un problema pues todos son compañeros de trabajo y tienen sus propios móviles para hablar en caso de no poder en persona, pero si a futuro se buscara ampliar el proyecto tendría que haber algún tipo de sistema de mensajería o notificaciones.

Dicho sistema serviría principalmente para que los investigadores hicieran peticiones de cambios al administrador, ya que ellos mismos no pueden gestionar las citas ya completadas, y para que el administrador les comunicase errores o cambios detectados. En principio un sistema de mensajes cortos sería más que suficiente pero al ponerlo en uso quizás se descubriesen otras herramientas de comunicación más útiles o eficientes.

CAPÍTULO 9

Bibliografía

- [1] Uriel Hernandez. Qué es typescript.
- [2] Wikipedia. Html5, 2019.
- [3] Wikipedia. Css, 2019.
- [4] Wikipedia. Java, 2019.
- [5] Wikipedia. Java8, 2019.
- [6] Carlos Eduardo Plasencia Prado. Sql.
- [7] Wikipedia. Visual studio code.
- [8] Hostinger.
- [9] Wikipedia. Eclipse.
- [10] David Pacios Izquierdo José Luis Vázquez Poletti. Oficina sotware libre ucm - curso básico latex.
- [11] Publicaciones oficina software libre ucm.
- [12] Jonathan Zea. Spring framework ¿qué es y para qué sirve? – java, 2017.
- [13] Grégor González. ¿que es jpa? diferencia con hibernate?, 2019.
- [14] Álvaro Fontela. ¿que es bootstrap?, 2015.
- [15] Bootstrap team. ¿que es bootstrap?