Universitat Oberta de Catalunya

uoc.edu

BleesYou - La forma actual de conectar personas y profesionales de la salud

Paloma Terán Pérez
Grado de Ingeniería Informática
Desarrollo web

Gregorio Robles Martínez
Santi Caballe Llobet

08/01/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*BleesYou - La forma actual de conectar personas y profesionales de la salud*
Nombre del autor:	*Paloma Terán Pérez*
Nombre del consultor/a:	*Gregorio Robles Martínez*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	Grado de Ingeniería Informática
Área del Trabajo Final:	TFG - Desarrollo web
Idioma del trabajo:	*Español*
Palabras clave	*Aplicación web, consulta médica, Google Fit*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo del siguiente proyecto es el análisis y desarrollo de una aplicación que permite conectar médicos y pacientes de forma online y sincronizar los datos de Google Fit de los pacientes. De esta forma, los médicos podrán utilizar estos datos para hacer mejores diagnósticos y ofrecer tratamientos más adecuados para el paciente.

Para llegar a esta meta, se ha decidido seguir una metodología ágil usando un tablero Kanban en Trello [4]. Para desarrollarla, se ha utilizado Spring Framework [3] en Java para el desarrollo del backend, Angularjs [2] para el desarrollo frontend, junto con Bootstrap [1] para la realización del diseño, y MySQL [9] como sistema de bases de datos. También, hay que nombrar que la autentificación se hace mediante los servicios Oauth que provee Spring y la sincronización de los datos de Google Fit, mediante los servicios Rest que provee Google, Google Rest Services.

Para la validación de la aplicación, se han realizado pruebas manuales, donde se comprueban los requisitos y posibles escenarios de ellos, validando la aplicación.

Abstract (in English, 250 words or less):

The goal of the following project is the analysis and the development of an application that allows to connect online doctors and patients and synchronize Google Fit data of patients. In this way, doctors will be able to use this data to make better diagnoses and offer more appropriate treatments for the patient.

To reach this objective, it has been decided to use an agile methodology with a Kanban board in Trello [4]. Spring Framework [3] (Java) has been used for the development of the backend, Angularjs [2] together with Bootstrap [1] has been used for the frontend development and for the realization of the design, and MySQL [9] has been used as a database system. Also, it should be noted that the authentication is done through the Oauth services provided by Spring and the synchronization of the Google Fit data, through the Rest services provided by Google, Google Rest Services.

Manual tests have been performed to validate the application. Requirements and possible scenarios are checked in these tests, and, so, the application is validated.

Índice

Lista de figuras

1 Introducción

1.1 Contexto y justificación del Trabajo

Desde hace unos años, estamos viviendo una transformación digital de los negocios y servicios, en algunos sectores algo lenta a pesar de los grandes avances disponibles en tecnologías, velocidades de conexión, sistemas de almacenamiento, etc.

Ante el descubrimiento del nuevo virus Covid y su expansión por todo el planeta, un gran número de profesionales y empresas han tenido que buscar formas no presenciales para realizar su trabajo. En este grupo se incluyen a los profesionales de la salud, que, a pesar de estar forzados a realizar parte de su trabajo de forma presencial, otra parte de este pueden realizarlo de forma telemática y de esta forma evitar riesgos innecesarios.

Además, gracias a las pulseras y relojes inteligentes, y los móviles, disponemos de información muy útil almacenada en la nube, que puede ser de gran ayuda para estos profesionales de la salud, pero esta información a veces no está disponible para ellos.

Con este proyecto se intenta solucionar este problema, conectando a médicos y pacientes, y compartiendo información sobre la salud de los pacientes, extraída de la nube, con estos profesionales de la salud.

En resumen, **BleesYou mira por nuestra salud, facilitando la comunicación entre los profesionales de la salud y las personas, a través de seguimientos de los pacientes online, permitiendo a los profesionales consultar sus historiales y datos obtenidos de las pulseras y relojes inteligentes, *smartbands* y *smarthwachs* respectivamente, a través de los servicios que facilita Google.**

1.2 Objetivos del Trabajo

1.2.1 Objetivos principales

Este proyecto consta de varios objetivos:

- Analizar el desarrollo una de aplicación web para que personas puedan conectar con profesionales de la salud y estos puedan ver su perfil y datos de pulseras y relojes inteligentes almacenados en Google Fit.
- Realizar el desarrollo de la aplicación anteriormente mencionada.
- Realizar la documentación necesaria para su desarrollo: Análisis, diseño, etc.

1.2.2 Subobjetivos

Estos objetivos son más variados y no tienen que estar relacionados unos con otros:

- Mejorar los conocimientos en la realización de la documentación del desarrollo de una aplicación.
- Mejorar los conocimientos de Angularjs [2].
- Mejorar los conocimientos de Java.
- Ampliar el conocimiento de librerías, tecnologías y aplicaciones que se pueden usar en este tipo de proyectos, como puedes ser Eclipse [5], Visual Studio Code [6], frameworks de maquetación, etc.
- Conocer el funcionamiento de las APIS de Google Fit.
- La aplicación debe desarrollarse con código fácilmente entendible y adaptable a nuevos cambios.
- Se debe documentar el código mínimamente.

1.3 Enfoque y método seguido

Para la realización del proyecto, se ha decidido realizar una aplicación nueva usando Spring Framework [3], Angularjs [2] y MySQL [9]. Como metodología de trabajo se ha decidido usar metodología Ágil, dividiendo el trabajo en Sprints y usando un tablero Kanban en Trello [4].

1.3.1 Java

Java es un lenguaje de programación orientado a objetos y una plataforma informática. Es un lenguaje semicompilado que funciona con una máquina virtual, Java virtual machine, o JVM, que interpreta y transformar los programas en Java a lenguaje máquina para la PC.

Se ha elegido Java por las siguientes ventajas:

- Es un lenguaje simple y fácil de aprender.
- Es un lenguaje recomendado para el desarrollo de aplicaciones distribuidas.
- Es un lenguaje interpretado y compilado, su método de compilado permite ejecutarlo en un gran número de máquinas.
- Es seguro.
- Es de código abierto.
- Tiene un porcentaje de uso muy alto, por lo cual es fácil obtener ayuda e información.
- Existen un gran número de librerías y frameworks en este lenguaje que nos ayudan al desarrollo de aplicaciones.

Aunque existe un gran número de detractores de este lenguaje, y una eterna “pelea” entre Java y .Net y entre Java y Ruby, nunca se podrá negar, si se decide implementar una aplicación con Java, que ha sido una buena opción, debido a todas las ventajas anteriores.

1.3.2 Angularjs

AngularJS [2] es un framework MVC de JavaScript para el Desarrollo Web Frontend que permite crear aplicaciones SPA Single-Page Applications de lado de cliente y es mantenido por Google. AngularJS [2] es la primera versión de este framework, a partir de esta primera versión, se decidió rehacer el framework, y pasó a llamarse Angular.

Se ha decidido realizar el proyecto con Angularjs [2] debido a las siguientes ventajas:

- Angularjs [2] está mantenido por Google.
- Existe un gran número de librerías de código abierto y gratuitas para este lenguaje.

- Tiene una curva de aprendizaje complicada, se tarda mucho en entender su funcionamiento, meses, pero una vez hecho, el aprendizaje va mucho más rápido.
- Permite un manejo de eventos web sin uso de JQuery, muy eficiente y sencillo.
- Permite desarrollar el código front, dividiendo el código en componentes muy pequeños, lo cual facilita su comprensión y mantenimiento.
- Angularjs [2] divide la parte visible (HTML) de la parte lógica (JavaScript), lo cual facilita la programación.

1.3.3 MySQL

MySQL [9] es un sistema de gestión de bases de datos relacionales de código abierto con un modelo cliente-servidor, sirve para crear y administrar bases de datos basadas en un modelo relacional.

Las razones por las que se ha escogido MySQL [9] son:

- Es Open Source (Licencia GPL).
- Tiene una gran velocidad al realizar las operaciones, es uno de los motores de bbdd con mejor rendimiento.
- Permite su ejecución en máquinas con pocos recursos.
- Es fácil de configurar, instalar y manejar.
- Es bastante seguro, tiene una probabilidad baja de corromper datos.
- Es una buena base de datos para utilizarla en sistemas distribuidos.

1.3.4 Metodología ágil

El proyecto se va a desarrollar bajo metodología ágil, utilizando un tablero Kanban en Trello [4], y centrando el objetivo en diseñar una aplicación útil y servible frente a la documentación. Por esta razón la documentación se ha ido actualizando a la vez que se ha ido desarrollando el proyecto.

Agile describe un conjunto de principios rectores que utilizan un enfoque iterativo para el desarrollo de proyectos o software y reivindica 4 valores:

1. Las interacciones de las personas sobre los procesos y las herramientas.
2. Un software en funcionamiento frente a documentación exhaustiva.
3. La participación activa del cliente durante todo el proceso de desarrollo.
4. La capacidad de respuesta ante los cambios e imprevistos.

1.4 Planificación del Trabajo

Se realiza una planificación temporal teniendo en cuenta lo siguiente:

- La documentación se va a ir actualizando al final de cada tarea.
- En la fase de desarrollo:
    - Se va a ir actualizando el documento de pruebas a medida que se desarrollen las tareas, no se dará por finalizada una tarea si no está finalizada su parte correspondiente de pruebas.
    - Al igual que con las pruebas, se deberá ir actualizando la documentación de desarrollo a medida que se realiza cada tarea.

De esta forma, la documentación siempre estará actualizada, y será menos probable que existan “incongruencias” entre desarrollo y documentación.

Nombre de tarea	Comienza en	Duración en días	Fecha de comienzo	Fecha de fin
Inicio del proyecto	**0**	**13**	**16/09/2020**	**28/09/2020**
Investigación sobre la idea de proyecto y recopilación de información	0	3	16/09/2020	18/09/2020
Título, descripción y objetivos	3	2	19/09/2020	20/09/2020
Tecnologías	5	3	21/09/2020	23/09/2020
Riesgos	8	2	24/09/2020	25/09/2020
Planificación	10	3	26/09/2020	28/09/2020
Análisis y diseño de la aplicación	**13**	**34**	**29/09/2020**	**02/11/2020**
Alcance	13	3	29/09/2020	01/10/2020
Casos de uso	16	6	02/10/2020	07/10/2020
Requisitos	22	5	08/10/2020	12/10/2020
Documento de requisitos	27	1	13/10/2020	13/10/2020
Prototipo de pantallas	28	6	14/10/2020	19/10/2020
Configuración del espacio de trabajo	34	4	20/10/2020	23/10/2020
Arquitectura básica frontend	38	6	24/10/2020	29/10/2020
Arquitectura básica backend	44	3	30/10/2020	02/11/2020
Desarrollo de la aplicación	**47**	**49**	**03/11/2020**	**21/12/2020**
Modelo de bases de datos	47	3	03/11/2020	05/11/2020
Diagramas de clases	50	5	06/11/2020	10/11/2020
Diseño de la aplicación	55	5	11/11/2020	15/11/2020
Primera versión documento desarrollo	60	3	16/11/2020	18/11/2020

Nombre de tarea	Comienza en	Duración en días	Fecha de comienzo	Fecha de fin
Primera versión documento pruebas	63	3	19/11/2020	21/11/2020
Login	66	5	22/11/2020	26/1212020
Historial paciente	71	4	27/11/2020	30/11/2020
Contacto persona/profesional	75	5	01/12/2020	05/12/2020
Google Fit	80	10	06/12/2020	15/12/2020
Finalización del documento de desarrollo	90	3	16/05/2020	18/12/2020
Finalización del documento de pruebas	93	3	19/12/2020	21/12/2020
Finalización del proyecto	**96**	**18**	**22/12/2020**	**08/01/2021**
Compilación de la documentación y código fuente	96	8	22/12/2020	29/12/2020
Elaboración de la presentación final	104	10	30/12/2020	08/01/2021

1.4.1 Diagrama de Gantt

Este diagrama nos presenta una vista general de las tareas programadas, así pues, todas las partes implicadas sabrán qué tareas tienen que completarse y en qué fecha. En él mostramos:

- La duración del proyecto.
- Qué tareas hay dentro del proyecto.
- Una estimación de cuánto llevará cada tarea.
- Cómo se superponen las tareas y/o si hay una relación entre ellas.

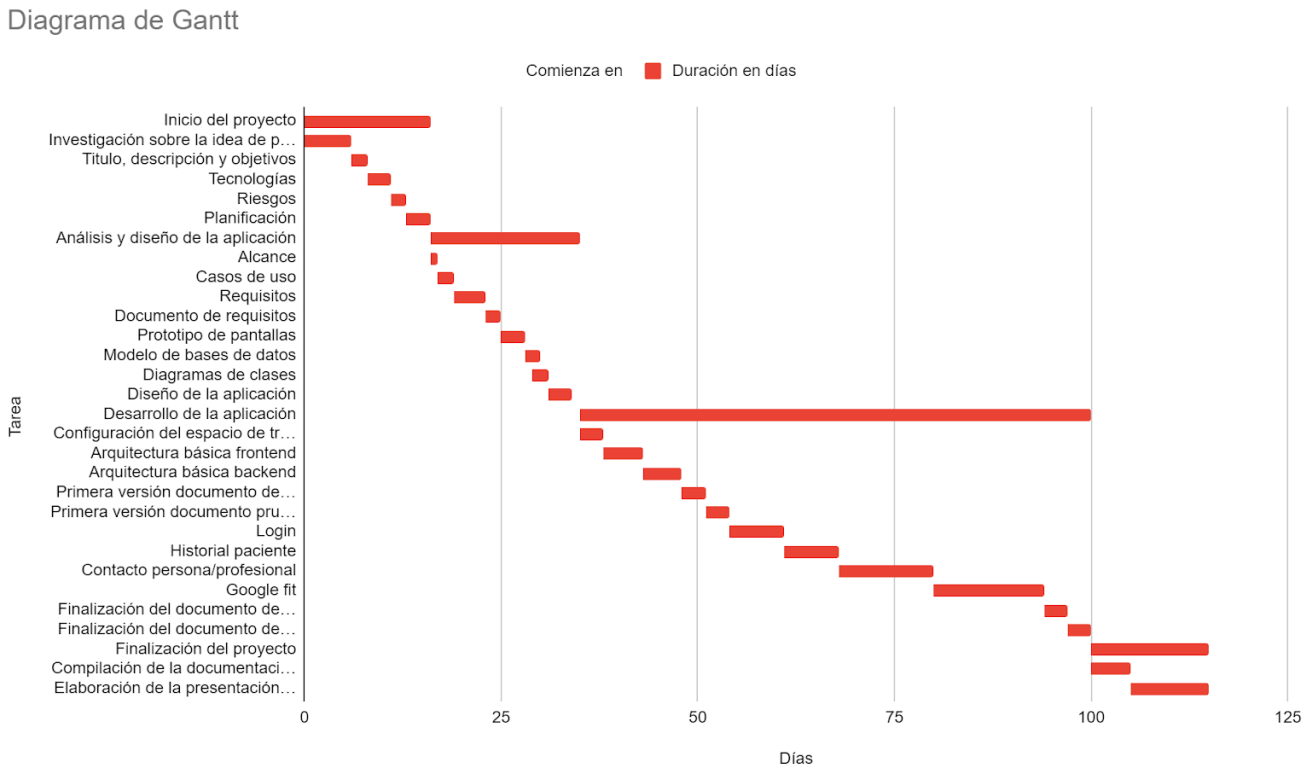
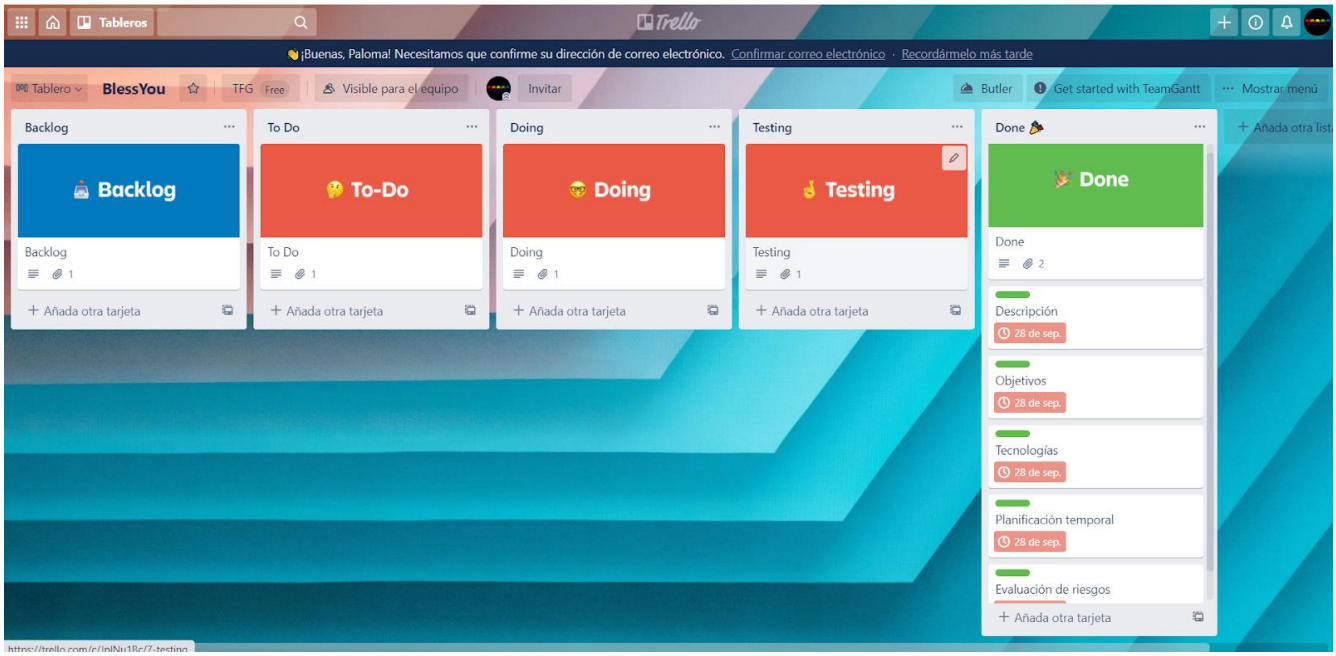


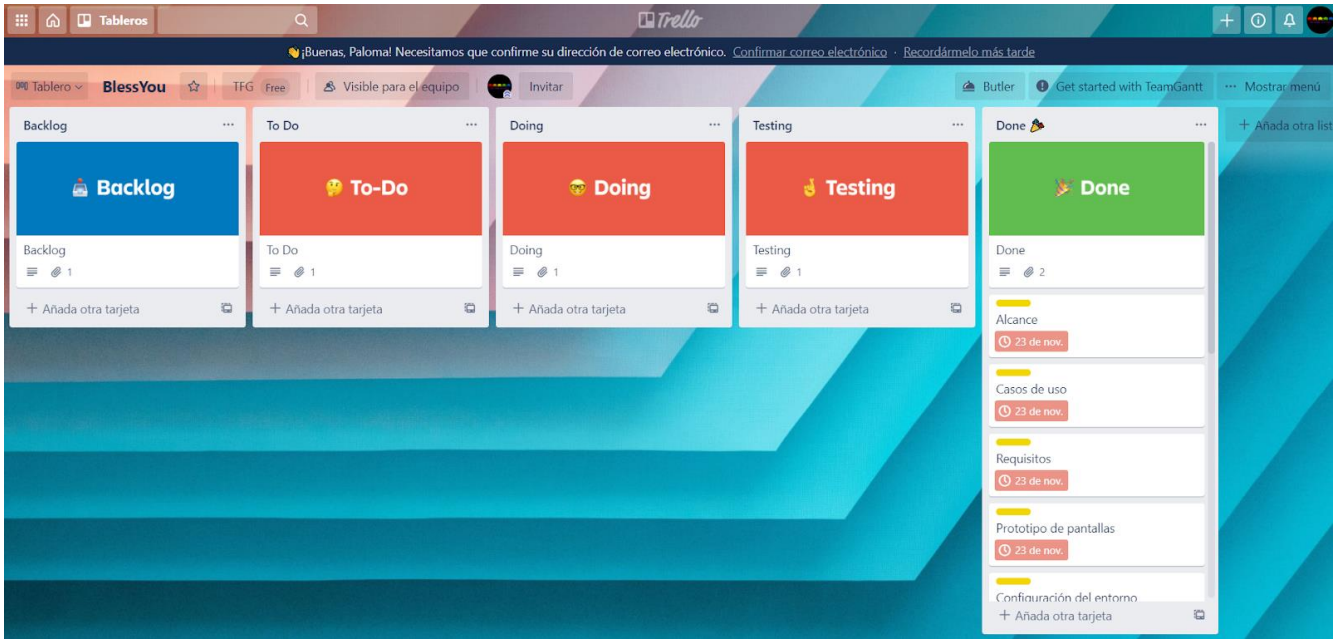
Diagrama de Gantt 1

1.4.2 Sprints

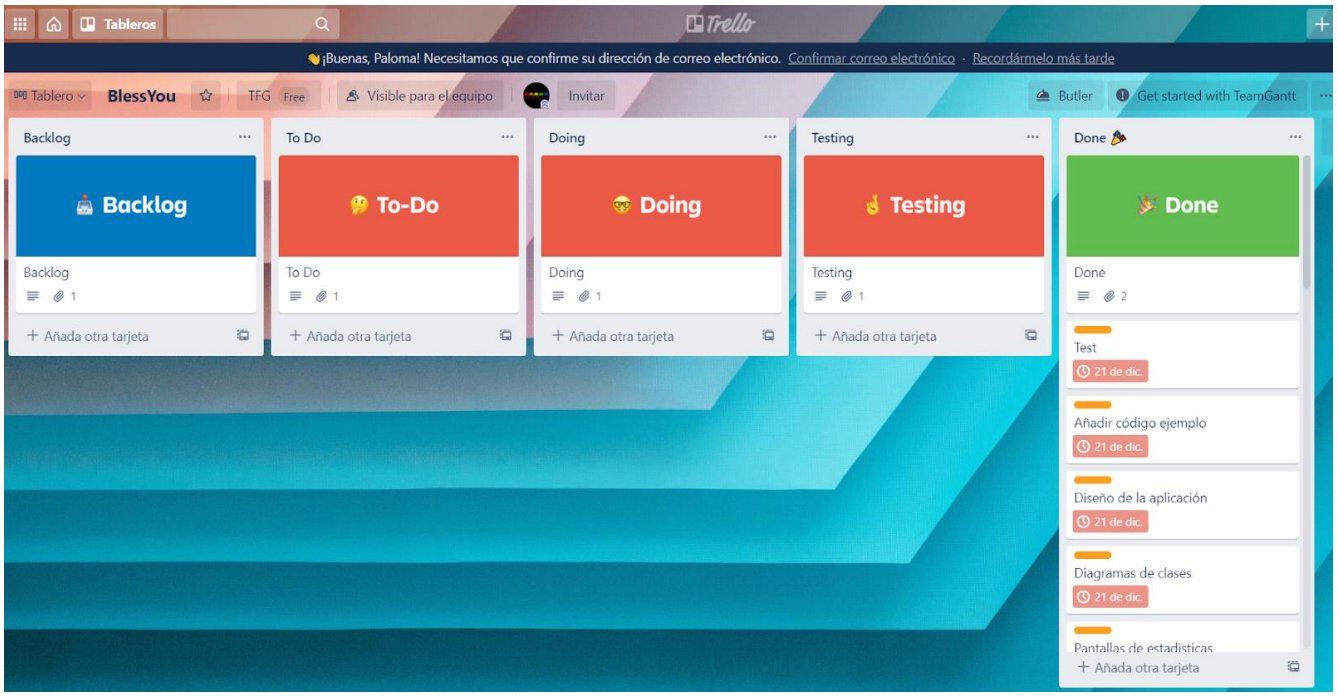
A continuación, se presentan las capturas de pantalla del tablero Kanban en Trello [4] de los 4 Sprints realizados.



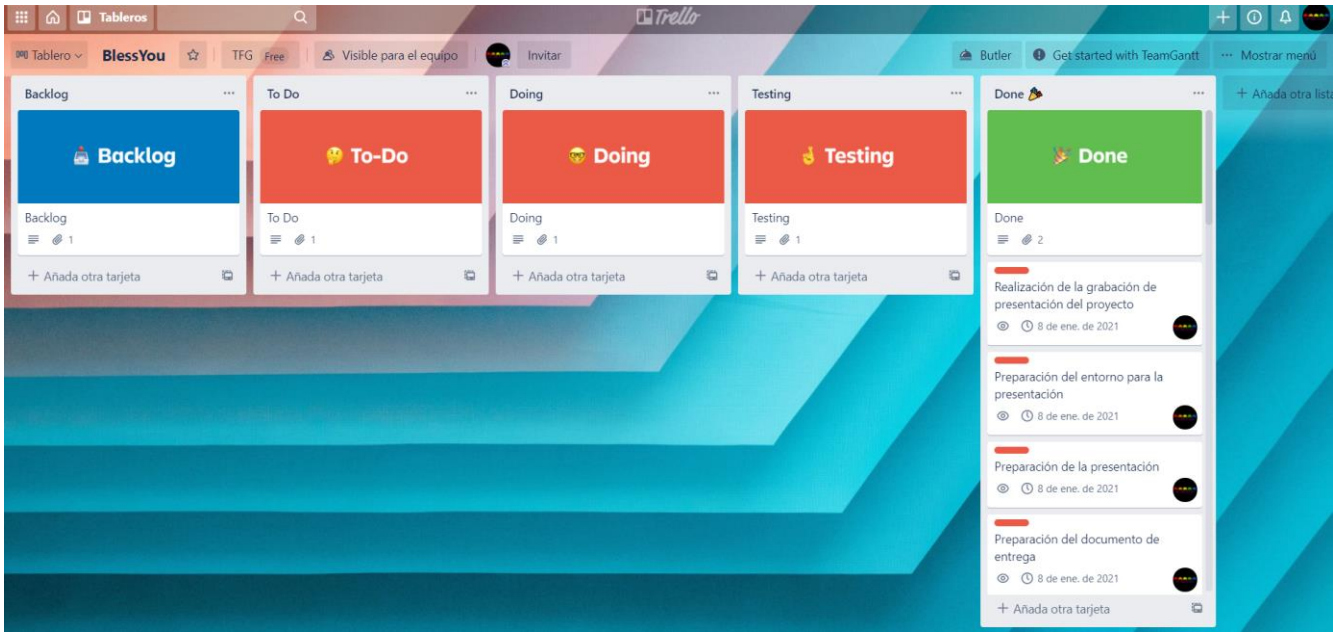
Kanban Trello - PEC 1



Kanban Trello - PEC 2



Kanban Trello - PEC 3



Kanban Trello - PEC 4

1.4.3 Evaluación de riesgos

En este apartado se muestra una lista de posibles riesgos que pueden surgir durante el desarrollo del proyecto:

Riesgo	Impacto	Probabilidad	Acciones
Dificultad para utilizar alguna de las tecnologías propuestas debido a problemas de licencias	Alto	Baja	Buscar una tecnología alternativa
Desfases y pérdida de información entre versiones de documentación o desarrollos.	Medio	Alta	- Actualizar la documentación a medida que se vaya desarrollando la aplicación. - No se cerrará una tarea sin comprobar que la documentación está actualizada.
Problemas al obtener información de Google Fit	Bajo	Alta	Se intentarán sincronizar los datos posibles en una primera versión.
Estimación inexacta: Demasiado alcance	Alto	Bajo	Esta aplicación se va a considerar como una primera versión, por lo cual tiene que ser sencilla y debe ser ampliada. Realizar un control de cambio periódicamente.
Bugs durante el desarrollo	Alto	Alta	Se intentarán realizar test durante el desarrollo, cada tarea deberá ser probada antes de cerrarse.
Arquitectura o código de baja calidad	Medio	Media	Al realizar una tarea se debe mantener el código limpio y actualizarlo correctamente.
Problemas con la interfaz	Alto	Media	- Diseñar una interfaz sencilla. - Investigar acerca de algún framework que facilite el desarrollo de la interfaz.

1.5 Breve sumario de productos obtenidos

Aplicación Blessyou que conecta médicos y pacientes y sincroniza datos de Google Fit de los pacientes. Dentro de esta se consideran tres subproductos que podrían ser independientes:

- Base de datos con la información que necesita la aplicación.
- Aplicación backend desarrollada en Java que ofrece una serie de servicios Rest de lectura y escritura.
- Aplicación frontend que provee al usuario del cliente para poder realizar las operaciones de forma accesible e intuitiva.

La aplicación se entrega como demo de la presentación, junto con el PowerPoint realizado para la entrega.

Documentación ofrecida en este documento necesaria para entender la arquitectura, funcionalidad y objetivos de la aplicación.

1.6 Breve descripción de los otros capítulos de la memoria

- **Capítulo 2 – Análisis funcional**: En esta parte especifica qué tiene que hacer la aplicación web y cómo tiene que hacerlo. Para ello se han elaborado requisitos, casos de uso y mocks de pantallas.

- **Capítulo 3 – Diseño de la aplicación**: En este capítulo se establece cómo debería ser o es la arquitectura de aplicación, desarrollando los apartados arquitectura de la aplicación, diagrama de bases de datos, diagrama de clases, un ejemplo de flujo de código y se especifica el control de versiones utilizado para la aplicación.

- **Capítulo 4 – Pruebas:** Se han realizado pruebas manuales, llamadas pruebas de caja negra, probando los posibles escenarios surgidos a partir de los requisitos redactados en su apartado correspondiente del capítulo 3.

2 Análisis funcional

2.1 Requisitos

2.1.1 Requisitos funcionales

Sistema de login	
RF.1.1	El sistema dispondrá de un sistema de login.
RF.1.2	Si el usuario de loguea correctamente se le redirigirá a su página principal
RF.1.3	Si hay un error durante el login, se mostrará el error en la pantalla.
RF.1.4	Los usuarios se podrán registrar como médico o paciente.
RF.1.5	El número de teléfono y el DNI serán únicos para cada usuario
RF.1.6	Durante el registro, si un dato no es válido, se informará al usuario del error.
RF.1.7	Si hay un error durante el registro, se mostrará el error al usuario.

Sistema contacto entre usuario y médicos	
RF2.1	Un paciente podrá buscar médicos por especialidad o nombre.
RF2.2	El listado de médicos mostrará su especialidad y nombre.
RF2.3	Un médico podrá buscar pacientes.
RF2.4	Un médico podrá ver el detalle de un paciente y acceder a su perfil y datos de Google Fit.

Perfil de paciente y médico	
RF.3.1	Un paciente podrá modificar su perfil
RF.3.2	Un médico podrá modificar su perfil.
RF.3.3	Un paciente podrá sincronizar sus datos de Google Fit con la aplicación

Perfil de paciente y médico	
RF.3.4	Un paciente puede actualizar sus datos de Google Fit.
RF.3.5	Los datos sincronizados de Google Fit se mostrarán en el perfil del paciente
RF.3.6	El paciente podrá sincronizar los datos del sueño.
RF.3.7	El paciente podrá sincronizar los datos de pasos.
RF.3.8	El paciente podrá sincronizar los datos de su frecuencia cardiaca.
RF.3.9	Un médico podrá consultar los datos de Google Fit de su paciente
RF.3.10	Un médico podrá modificar el perfil de sus pacientes.

2.1.2 Requisitos no funcionales

RNF.1	El software estará codificado de manera limpia y clara para garantizar su comprensibilidad.
RNF.2	El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 4 horas.
RNF.3	El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
RNF.4	La aplicación web debe poseer un diseño “Responsive” a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos tableta y teléfonos inteligentes.
RNF.5	El sistema debe poseer interfaces gráficas bien formadas.
RNF.6	El sistema será desarrollado para plataforma Chrome como mínimo
RNF.7	El sistema debe asegurar que los datos estén protegidos del acceso no autorizado.

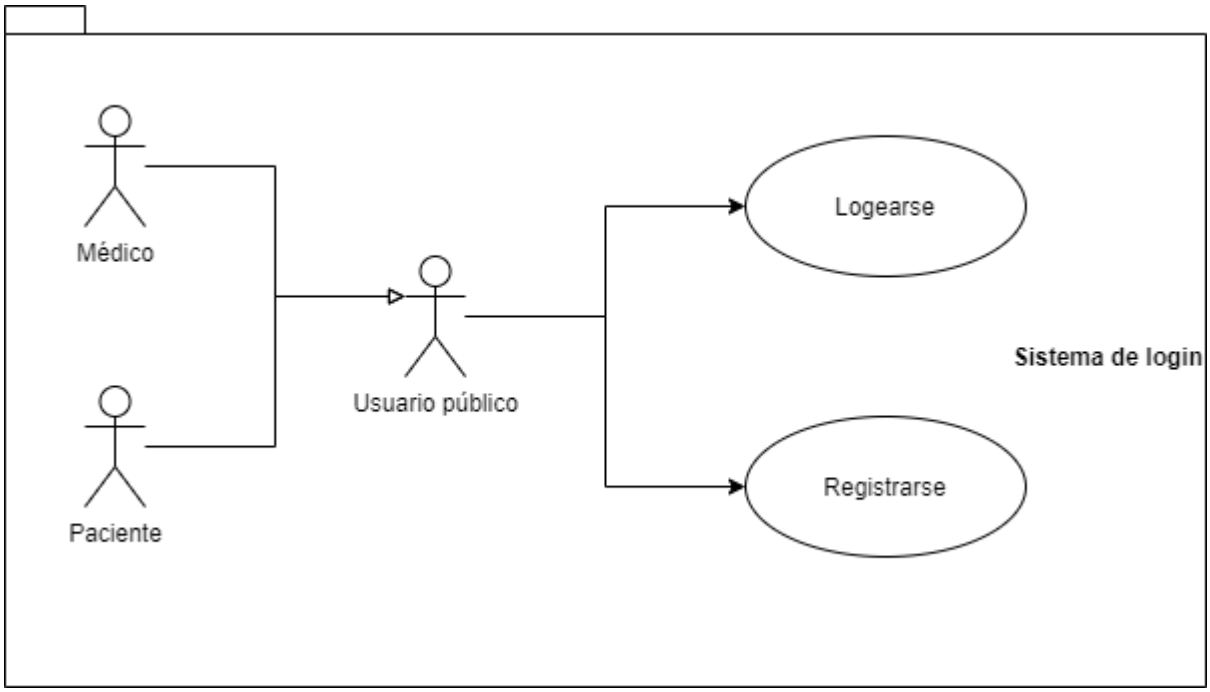
2.2 Casos de uso

El sistema se ha dividido en tres paquetes y se han obtenido los siguientes casos de uso a partir de los requisitos:

- Sistema de login
- Perfil de paciente
- Contacto entre médicos y pacientes

2.2.1 Subsistema de login

Un usuario público, independientemente de si es médico o paciente, podrá loguearse en la aplicación, o registrarse si no lo está ya.

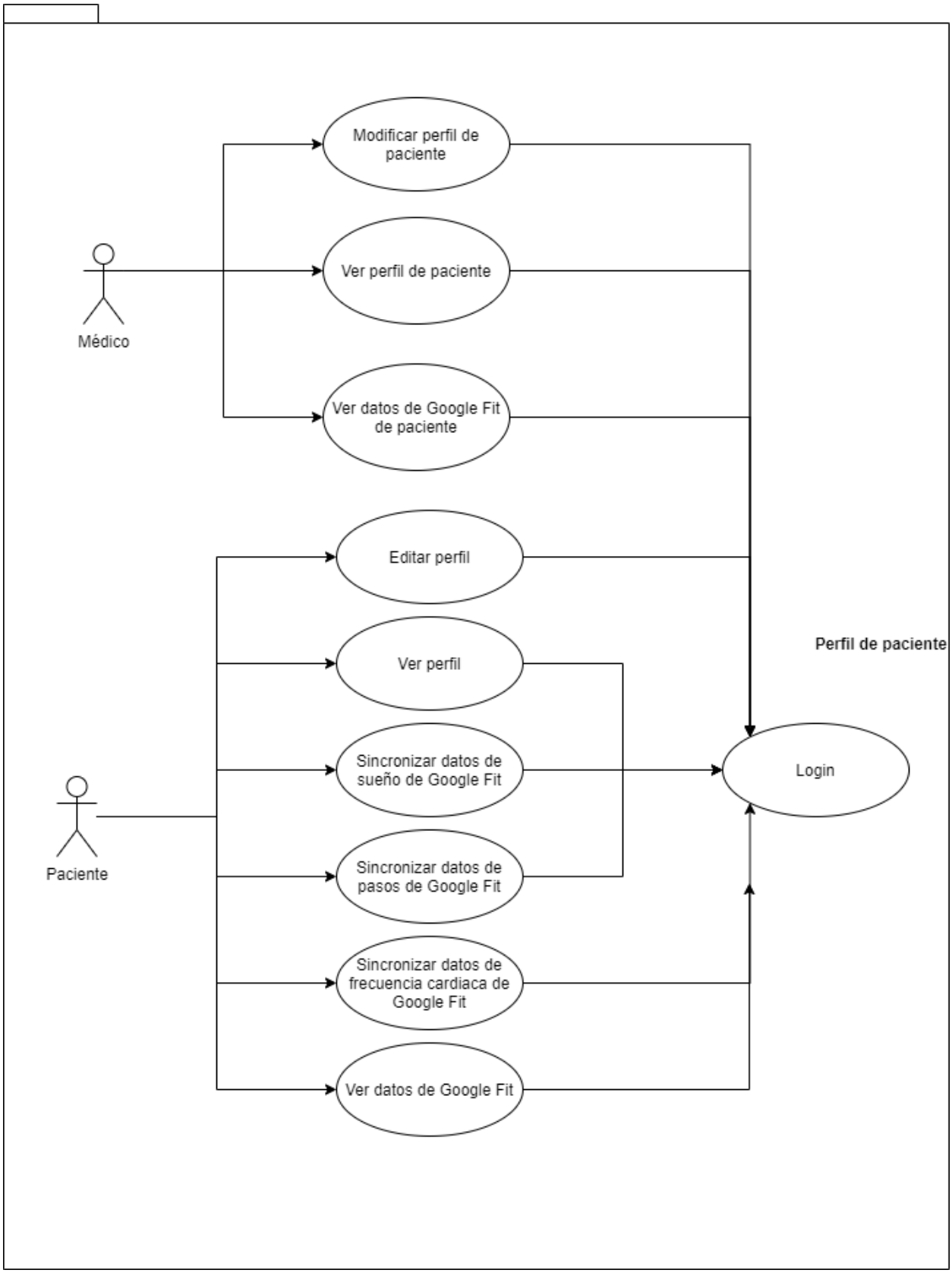


Caso de uso 1 – Sistema de login

2.2.2 Subsistema de perfil de paciente

Un paciente podrá rellenar su perfil, ver su perfil, sincronizar sus datos de Google Fit, tanto de pasos, de sueño como de pulsaciones y ver estos datos sincronizados.

Un médico podrá ver estos datos de Google Fit y su perfil, y modificar el perfil del paciente.

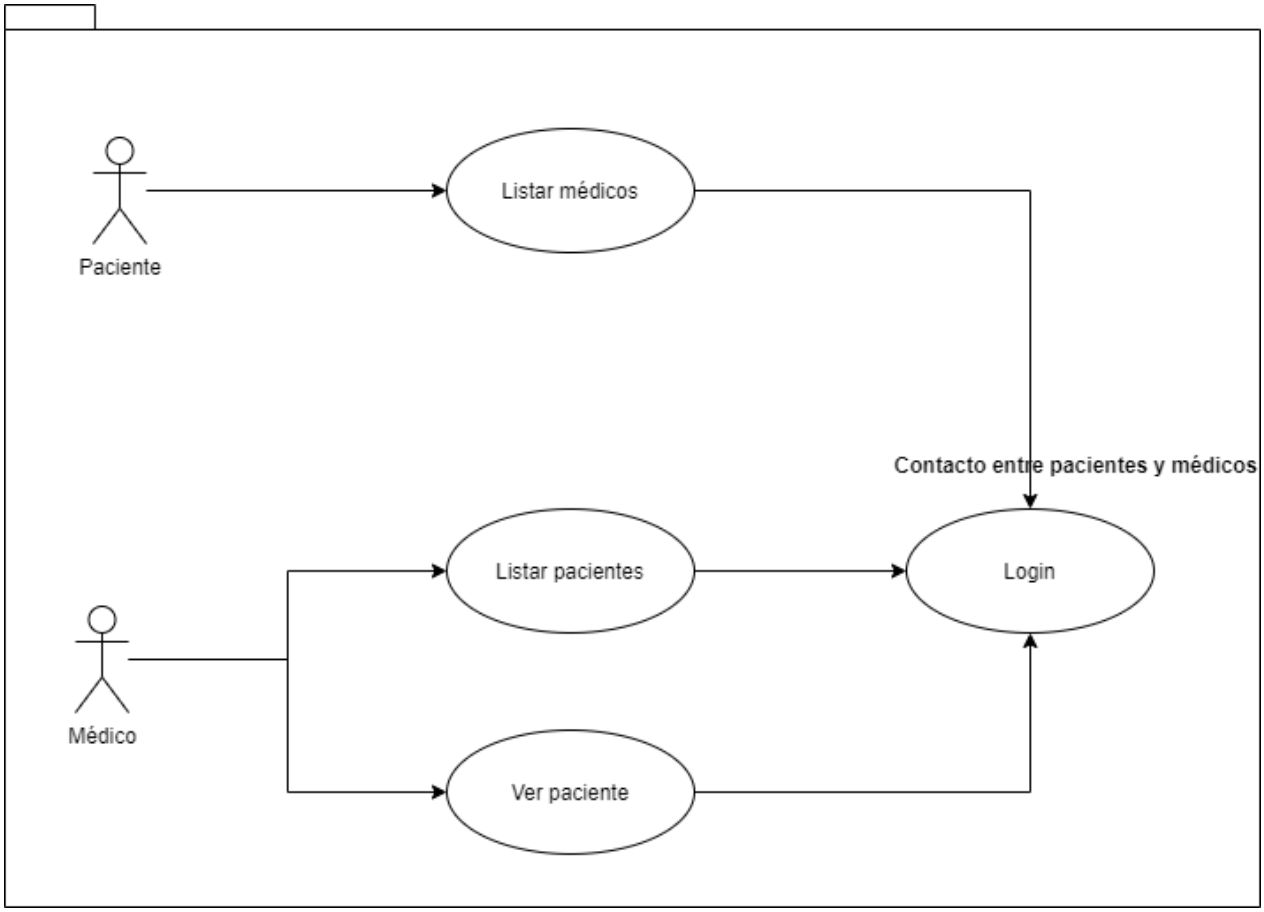


Caso de uso 2 – Perfil de paciente

2.2.3 Subsistema de contacto entre médicos y pacientes

Un paciente podrá listar los médicos.

Un médico podrá listar los pacientes existentes en la aplicación, para poder acceder desde ahí a su perfil.



Caso de uso 3 – Contacto entre médicos y pacientes

2.3 Mocks de pantallas

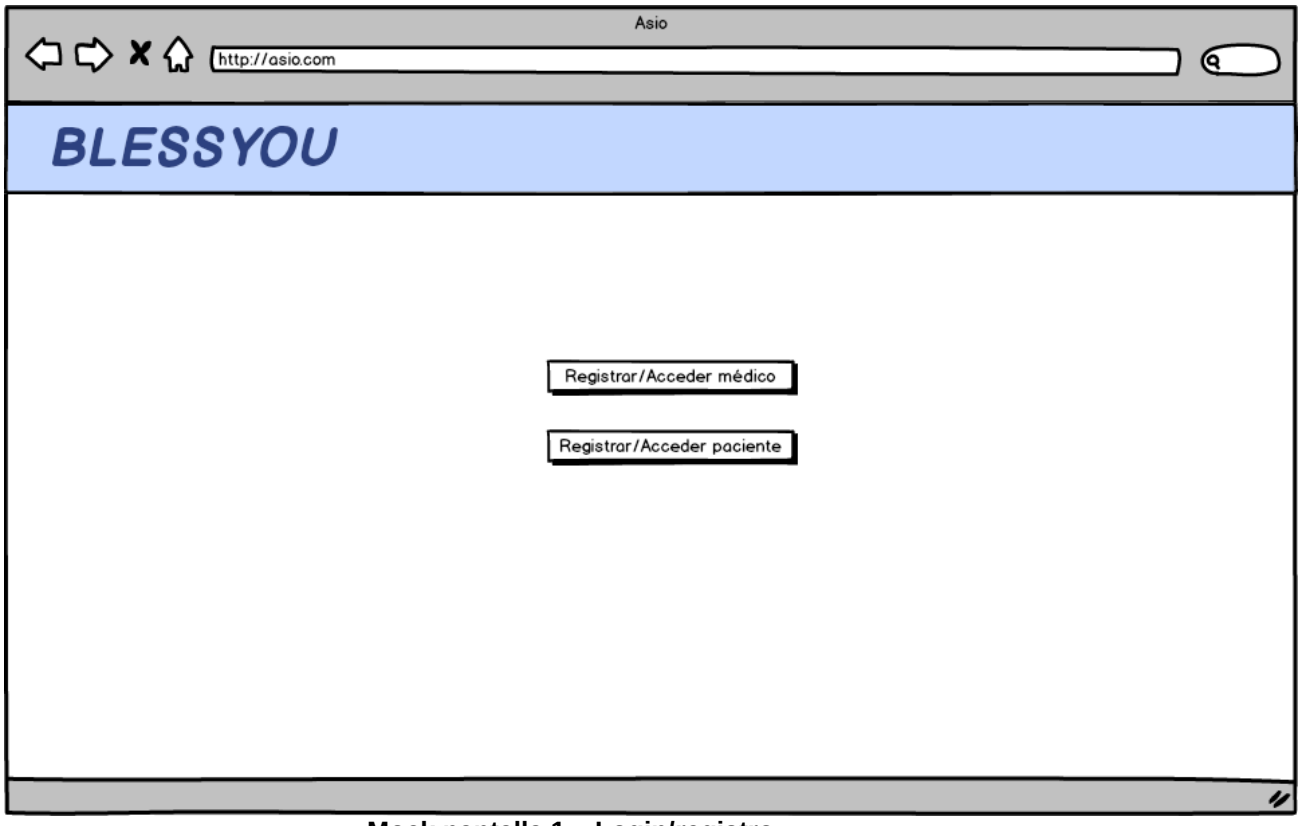
Se han realizado mocks de pantallas de varias de ellas, ya que algunas son bastante sencillas y no se consideran necesarios los mocks por ahora.

En cada pantalla, se puede ver en su título el resumen de su funcionalidad.

Se proponen pantallas sencillas y con fácil usabilidad.

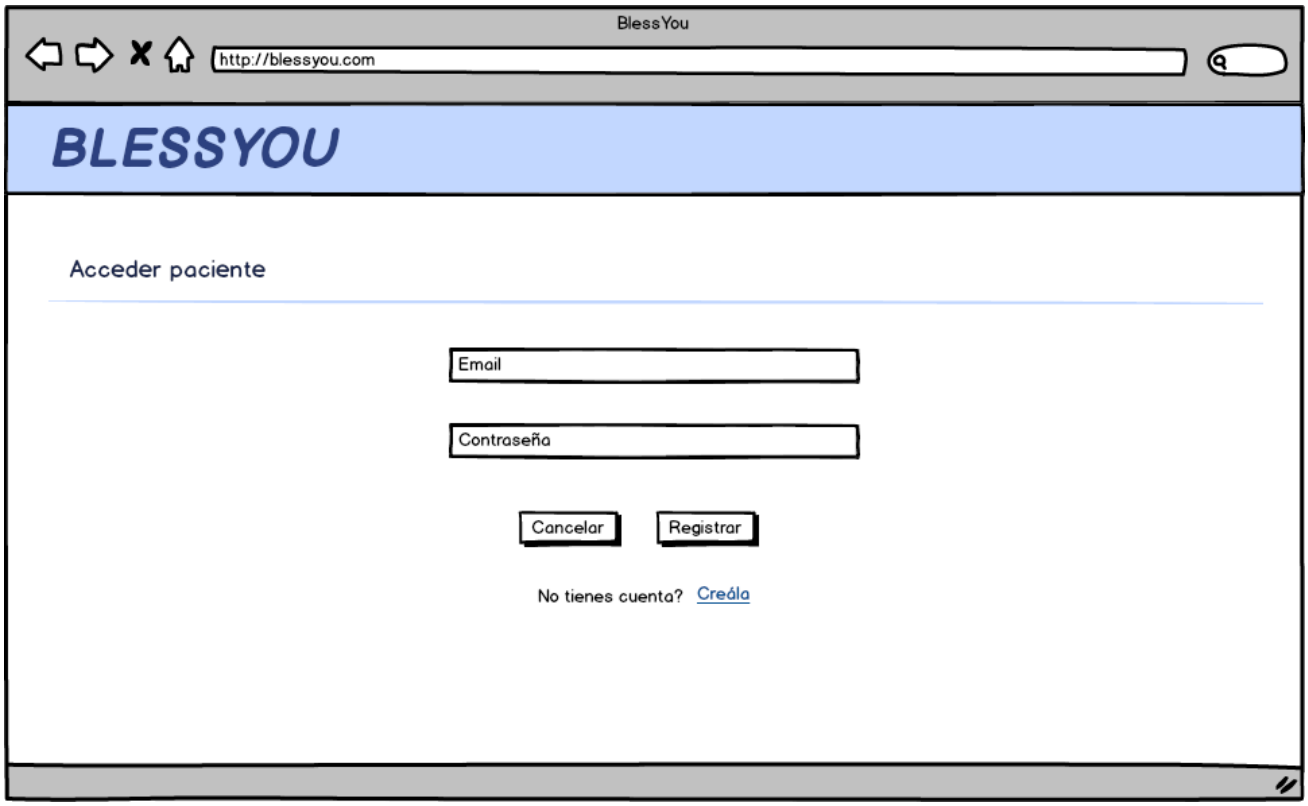
2.3.1 Pantallas de login/registro

Al acceder a la aplicación, se mostrará por defecto una pantalla donde el usuario podrá elegir acceder o loguearse como médico o cómo paciente, como se puede ver en la imagen *Mock pantalla 1*.



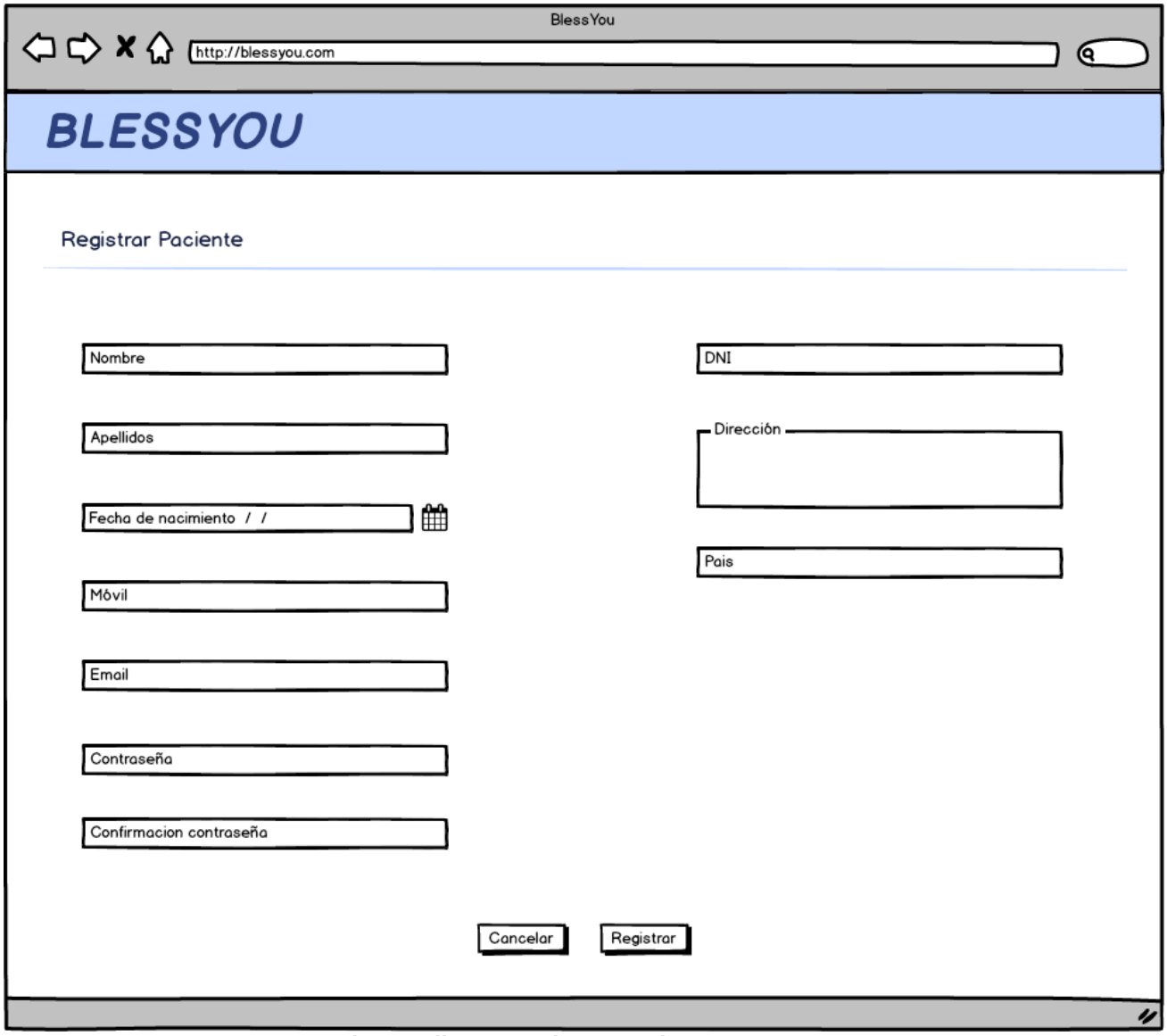
Mock pantalla 1 – Login/registro

En las imágenes *Mock pantalla 2 y Mock pantalla 4*, que serán similares tanto como para medico como para paciente, podemos ver, que el usuario podrá loguearse, y si no se dispone de cuenta, se le mostrará la opción de registrarse.

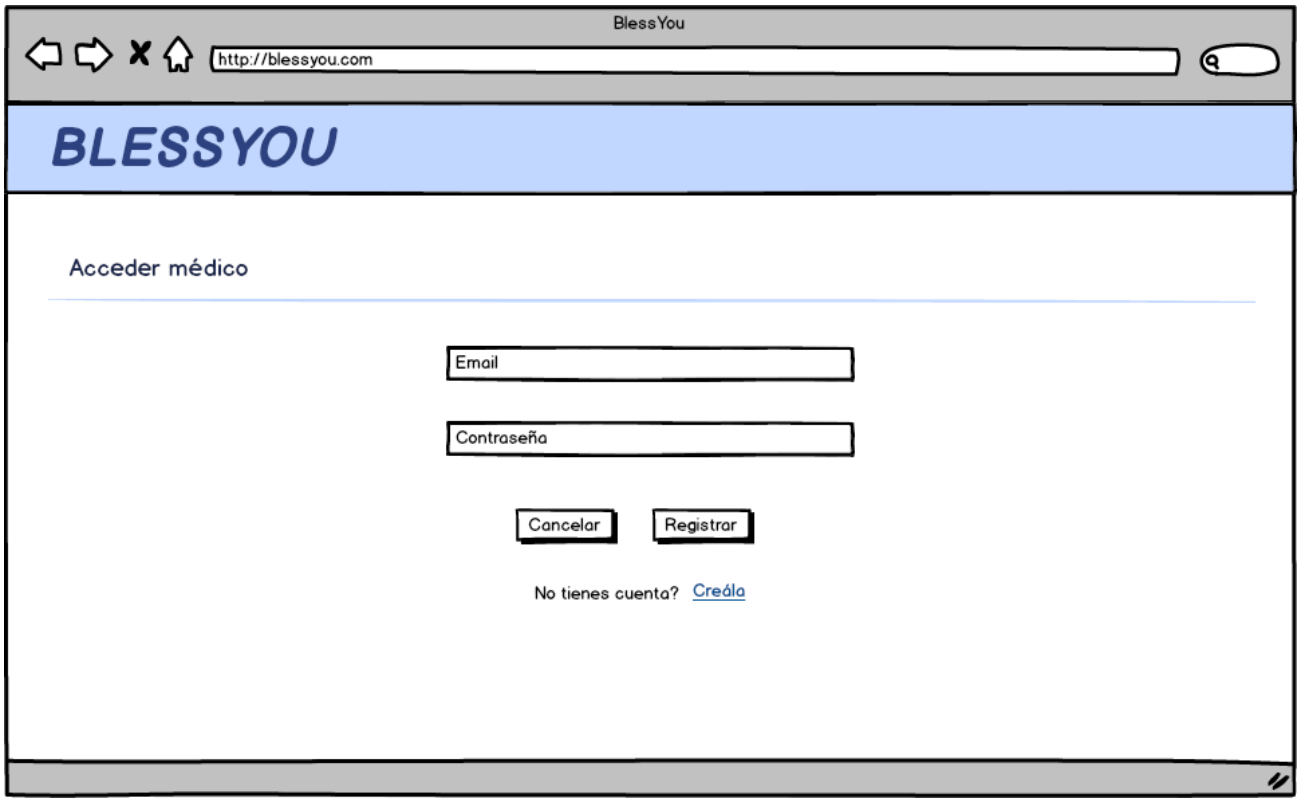


Mock pantalla 2 – Login paciente

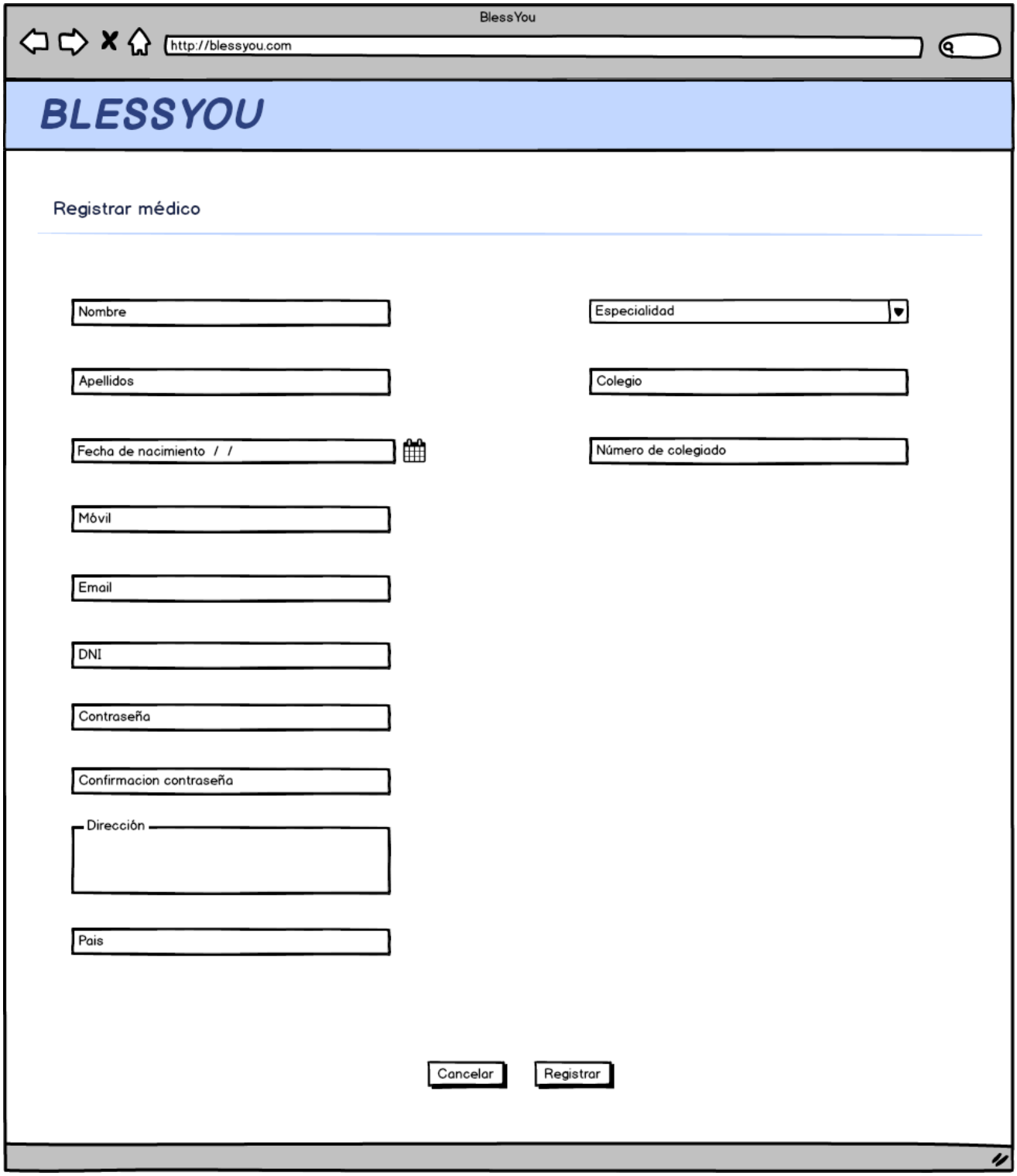
En la siguiente imagen, *Mock pantalla 3*, se pueden ver un esbozo de los campos que se mostrarán para registrar un paciente. Si estos campos no son correctos se mostrará un mensaje avisando del error y no se permitirá el registro. Se puede ver la acción de registro de médico, que funcionará de forma similar, en la imagen Mock pantalla 5.



Mock pantalla 3 – Registrar paciente



Mock pantalla 4 – Acceso médicos



Mock pantalla 5 – Registro de médicos

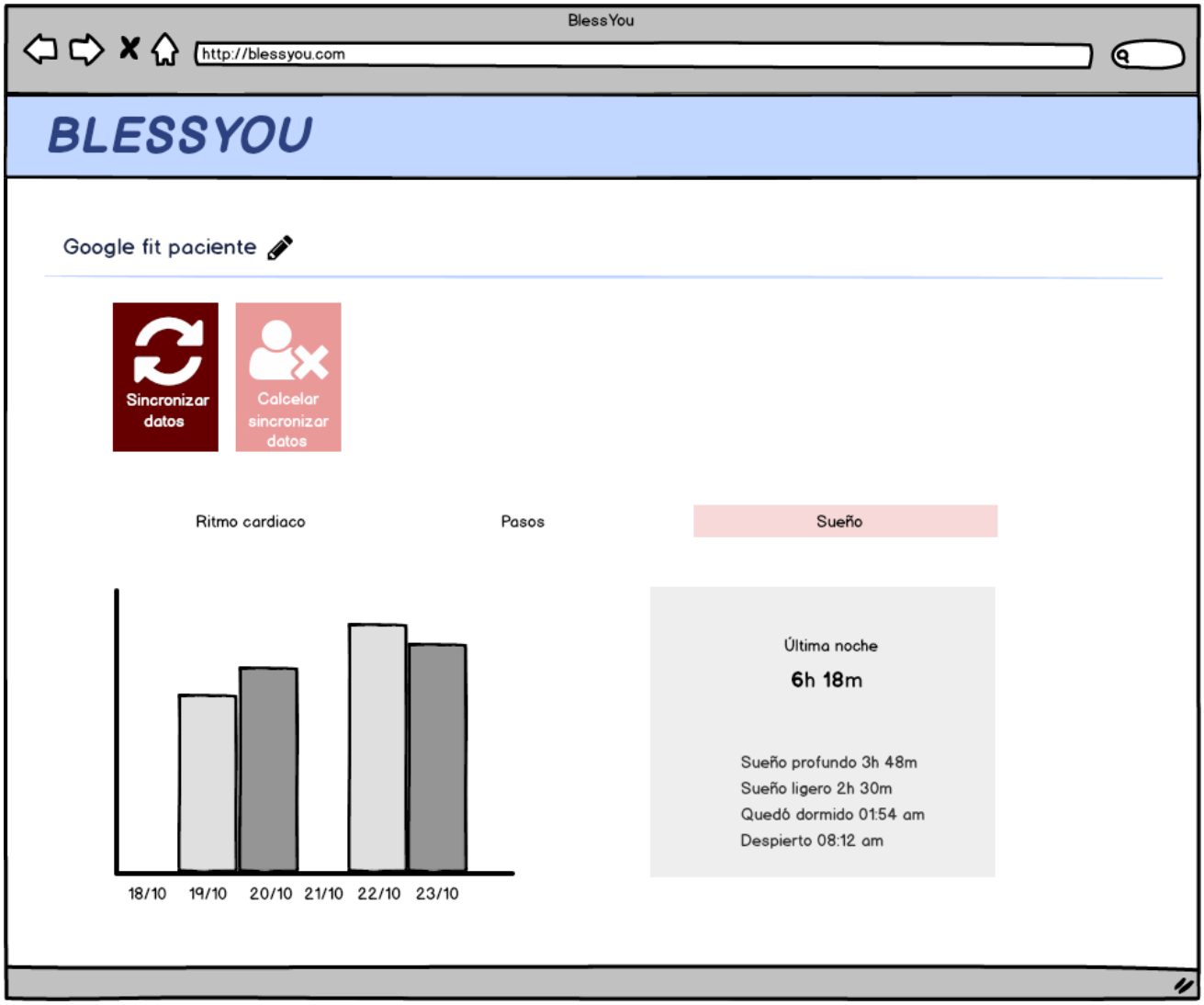
2.3.2 Pantallas de paciente

Cuando un usuario de tipo paciente accede a la aplicación, se mostrará un resumen de su perfil con su foto, y las opciones más importantes como son buscar médico y actualizar su perfil. Esto se puede ver en *Mock pantalla 6*.

Mock pantalla 6 – Perfil de paciente

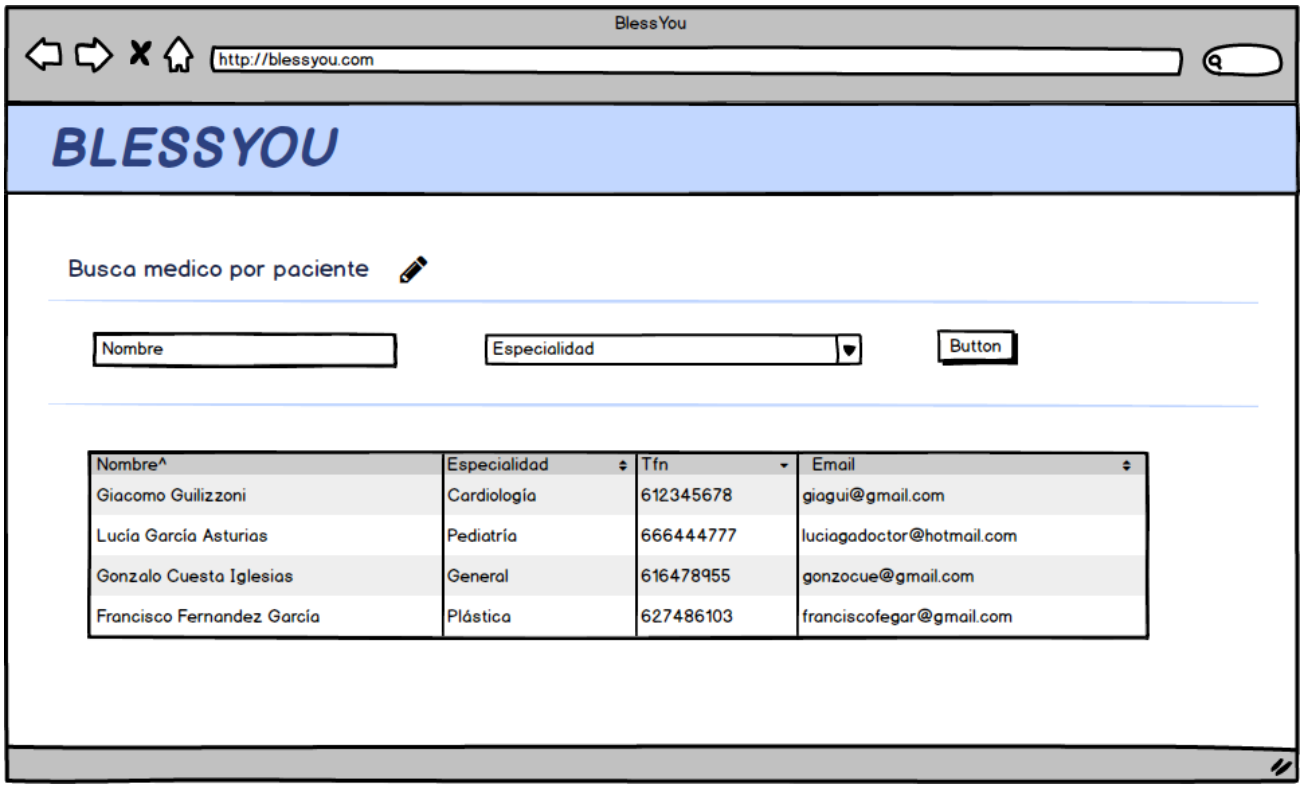
A través de un menú situado de la aplicación, el usuario podrá acceder a las funciones de sincronización con Google Fit, si no tiene datos tendrá que dar al botón sincronizar y dar los permisos a Google para que acceda a sus datos. Si ya está sincronizado de otras veces, podrá actualizar estos datos a través de un botón, o eliminar los permisos y los datos de Google Fit de la aplicación. Podemos ver un esbozo de estas pantallas en *Mock de pantalla 7 y Mock de pantalla 8.*

Mock pantalla 7 – Pasos paciente



Mock pantalla 8 – Sueño paciente

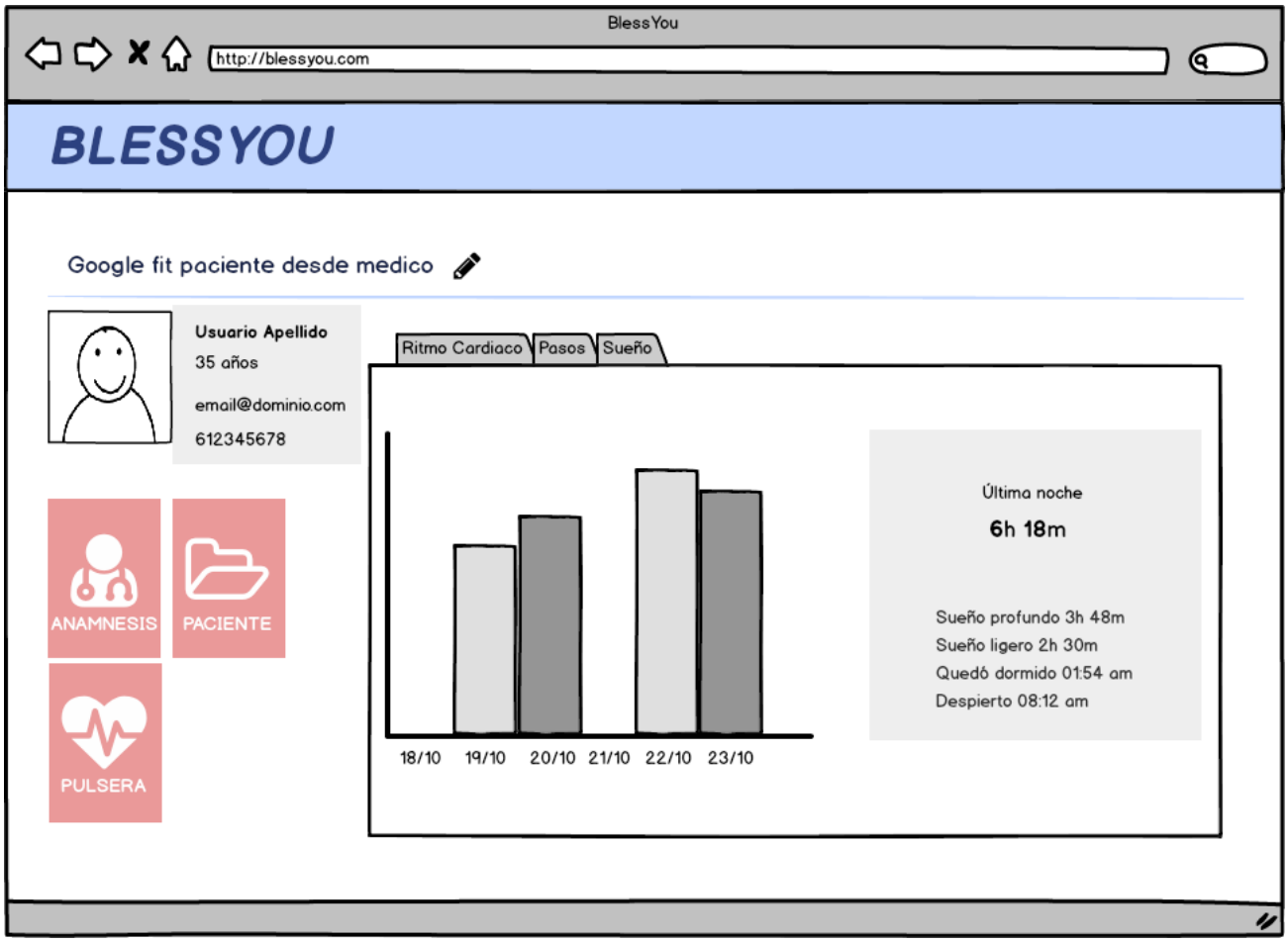
Además de todo esto, un paciente podrá buscar a un médico por nombre o especialidad a través de un buscador, donde se mostrará su email y teléfono para contactar con él, *Mock pantalla 9*.



Mock pantalla 9 – Buscador de médicos

2.3.3 Pantallas de médico

Un médico podrá buscar a un paciente y podrá ver los datos de su perfil y los datos que tiene sincronizados con Google Fit. La visualización de estos datos de Google Fit se mostrará de forma similar a como vemos en la imagen *Mock pantalla 10.*



Mock pantalla 10 – Google Fit de paciente desde medico

Por último

3 Diseño de la aplicación

3.1 Arquitectura

Se trata de un proyecto formado por una capa web en Angularjs [2] frontend y otra capa en Java backend, ambas son independientes.

3.1.1 Frontend

Esta capa es la encargada de mostrar la información al usuario a través de un navegador. Realiza las peticiones de login, registro, actualización de información, etc. al *backend* mediante peticiones http. Por esta razón, es una capa “tonta”, no realiza ninguna lógica de negocio, ya que estas acciones se harán en el *backend*.

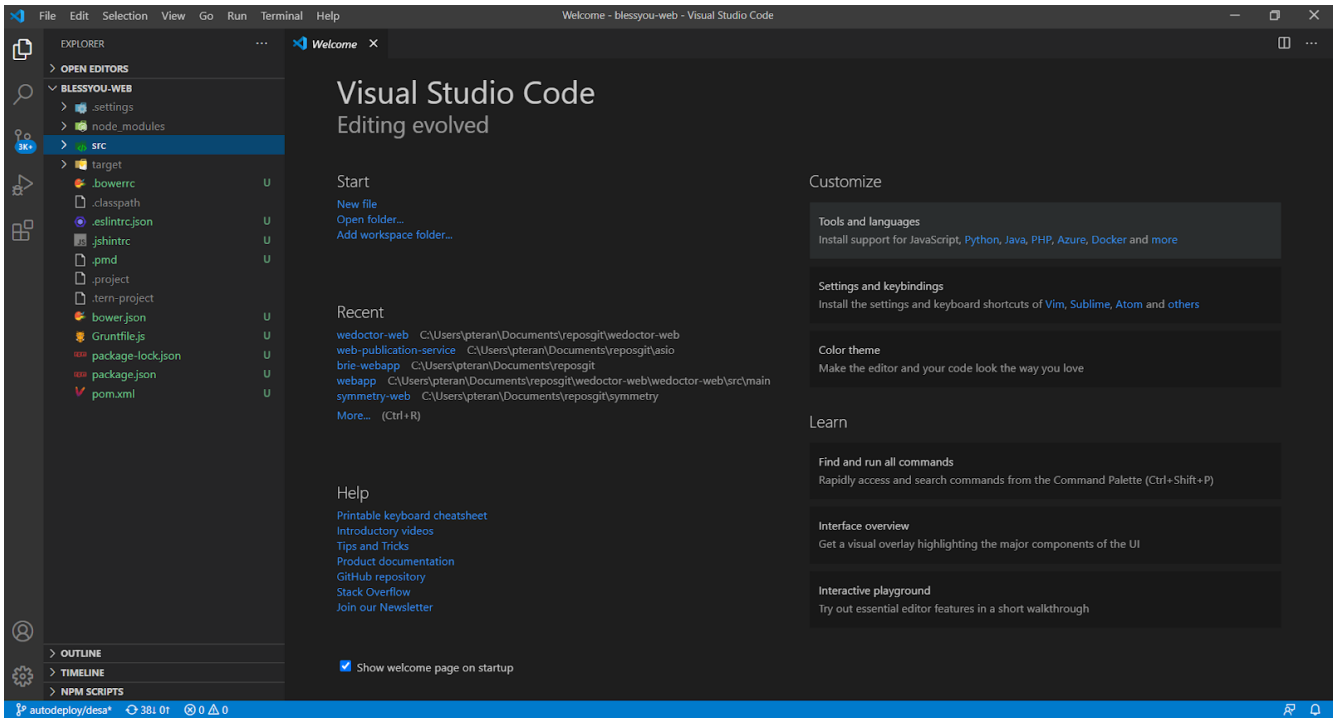
Angularjs [2] funciona con la inyección de elementos como:

- Directivas: Ayudan a extender los elementos y atributos básicos de HTML y a crear código reusable y comprobable.
- Componentes: compuestos por una template HTML, un archivo de lógica en JavaScript y un archivo de estilos.

Estos componentes son reusables en diferentes partes de la aplicación, facilitando la organización y el mantenimiento del código y evitando la duplicidad.

Para facilitar el diseño de la aplicación, tienes incluidas varias librerías:

- **Bootstrap [1]**: kit de herramientas de código abierto para desarrollos web responsive con HTML, CSS y JavaScript.
- **Font awesome [10]:** Framework de iconos vectoriales y estilos CSS, utilizado para sustituir imágenes de iconos comunes por gráficos vectoriales convertidos en fuentes.
- **Chartjs [11]**: Biblioteca JavaScript gratuita de código abierto para la visualización de datos, que admite 8 tipos de gráficos: barra, línea, área, circular, burbuja, radar, polar y dispersión.

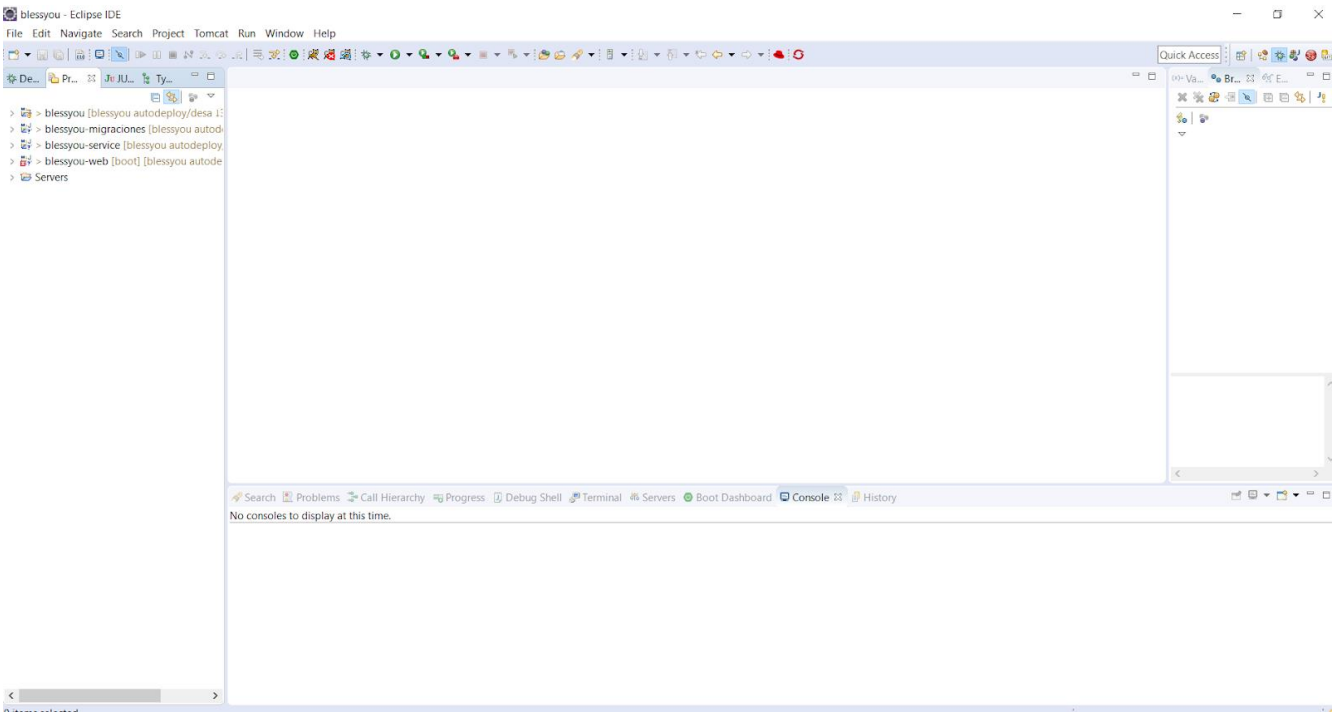


Diseño 1 – Visual Studio Code

3.1.2 Backend

Backend de la aplicación cuya finalidad es ejecutar la lógica de negocio de la aplicación. Para realizar el *backend* hay que destacar que se han utilizado dos tecnologías:

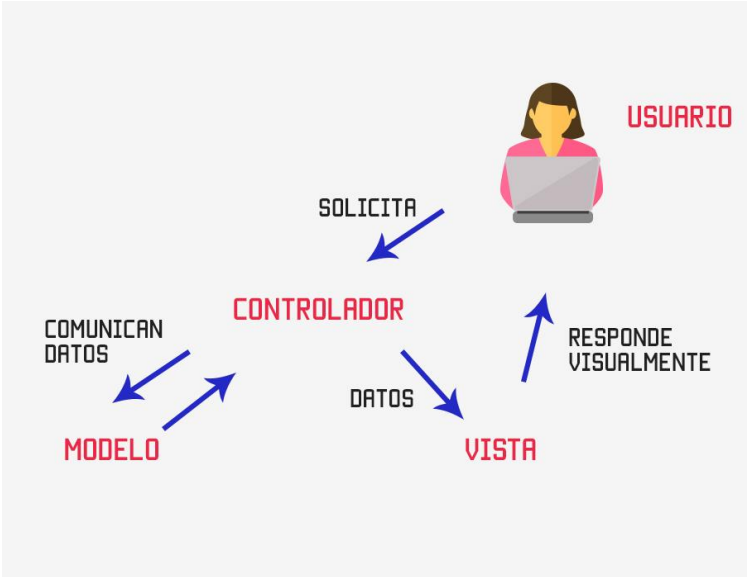
- **Maven:** Herramienta de software para la gestión y construcción de proyectos Java, similar a Apache Ant, pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.
- **Spring framework** [3]**:** Es un framework que nos facilita el trabajo y que fundamentalmente se centra en simplificar y flexibilizar la creación de objetos en nuestro programa. Trae incorporadas mejoras para una serie de librerías, de las que podemos destacar JPA y OAuth, y está implementado con el patrón MVC.



Diseño 2 - Eclipse

Se utilizan los siguientes patrones para la implementación:

- **Modelo Vista Controlador (MVC)**: patrón de arquitectura de software, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:
    - Modelo: contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
    - Vista, o interfaz de usuario: compone la información que se envía al cliente y los mecanismos interacción con éste.
    - Controlador: intermediario entre el Modelo y la Vista.



Diseño 3 – Modelo vista controlador

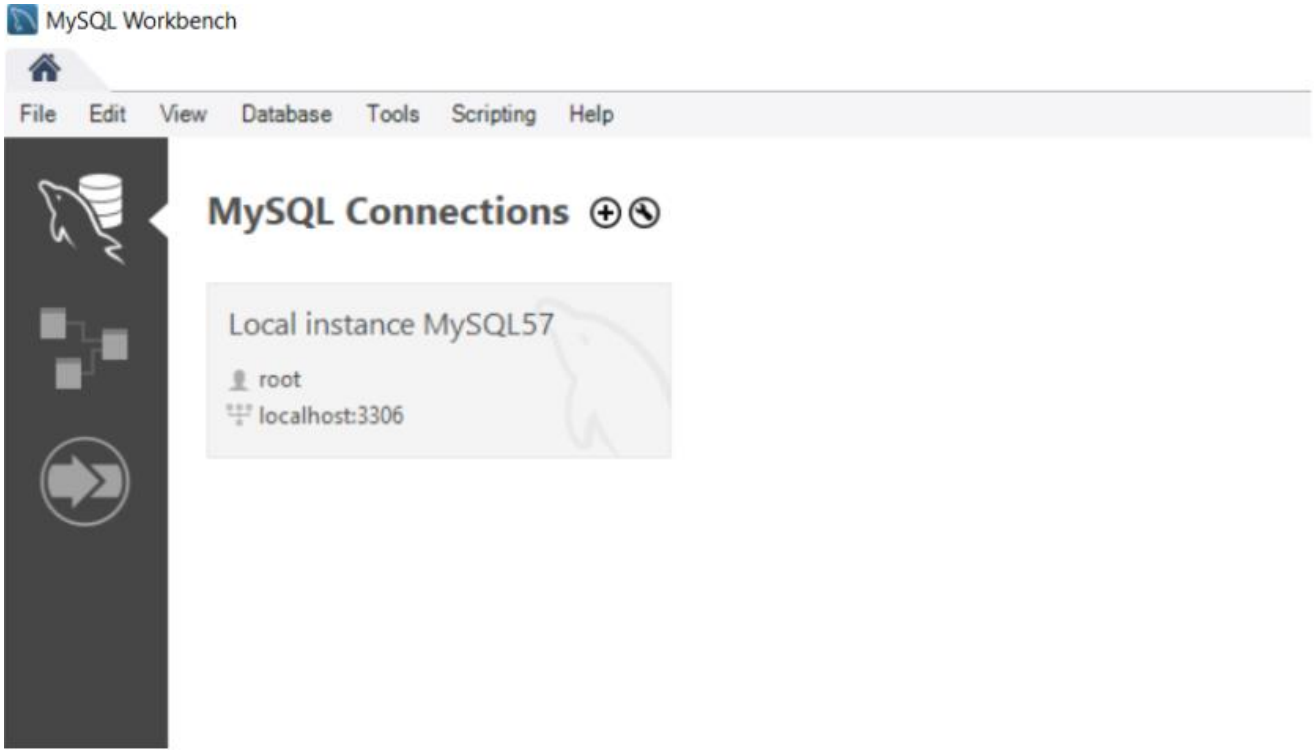
- **Patrón Proxy**: patrón estructural que tiene como propósito proporcionar un subrogado o intermediario de un objeto para controlar su acceso.
- **Patrón Data Access Object (DAO):** Es un patrón arquitectónico que permite separar la lógica de acceso a datos de los objetos de negocio, de tal forma que el DAO encapsula toda la lógica de acceso de datos al resto de la aplicación.
- **Patrón Data Transfer Object (DTO):** Es un patrón arquitectónico que se utiliza para transferir varios atributos entre el cliente y el servidor o viceversa. Va acompañado del patrón DAO.

El *backend* de la aplicación está formado por 3 proyectos:

- **BleesYou-web**: Proyecto que contiene los controladores para los servicios necesarios del front, en este caso Rest Services.
- **Blessyou-service**: Este es el proyecto más extenso, contiene:
    - Entidades: Clase java que representa a los objetos que se van a persistir en la base de datos mediante JPA.
    - DTO: Clase java que representa a los objetos que devuelven los Rest Services.
    - Mappers: Ayudan a transformar las entidades en DTOs y, al contrario.
    - Repositorios: Clases que ayudan a acceder a la base de datos para leer o actualizarlos.
    - Servicios: Clases java que realizan la lógica de negocio.
    - Proxy: Capa de clases que hacen de intermediarias entre los servicios y los controladores.
- **BleesYou-migraciones:** Proyecto que contiene ficheros SQL necesarios para la actualización de la aplicación.

3.1.3 Bases de datos

Se utiliza **MySQL** [9] que es un sistema de gestión de bases de datos relacional de código abierto, y se le considera como uno de los sistemas de bases de datos más populares del mundo para entornos de desarrollo web.



Diseño 4 – MySQL Workbench

3.1.4 Google Fit

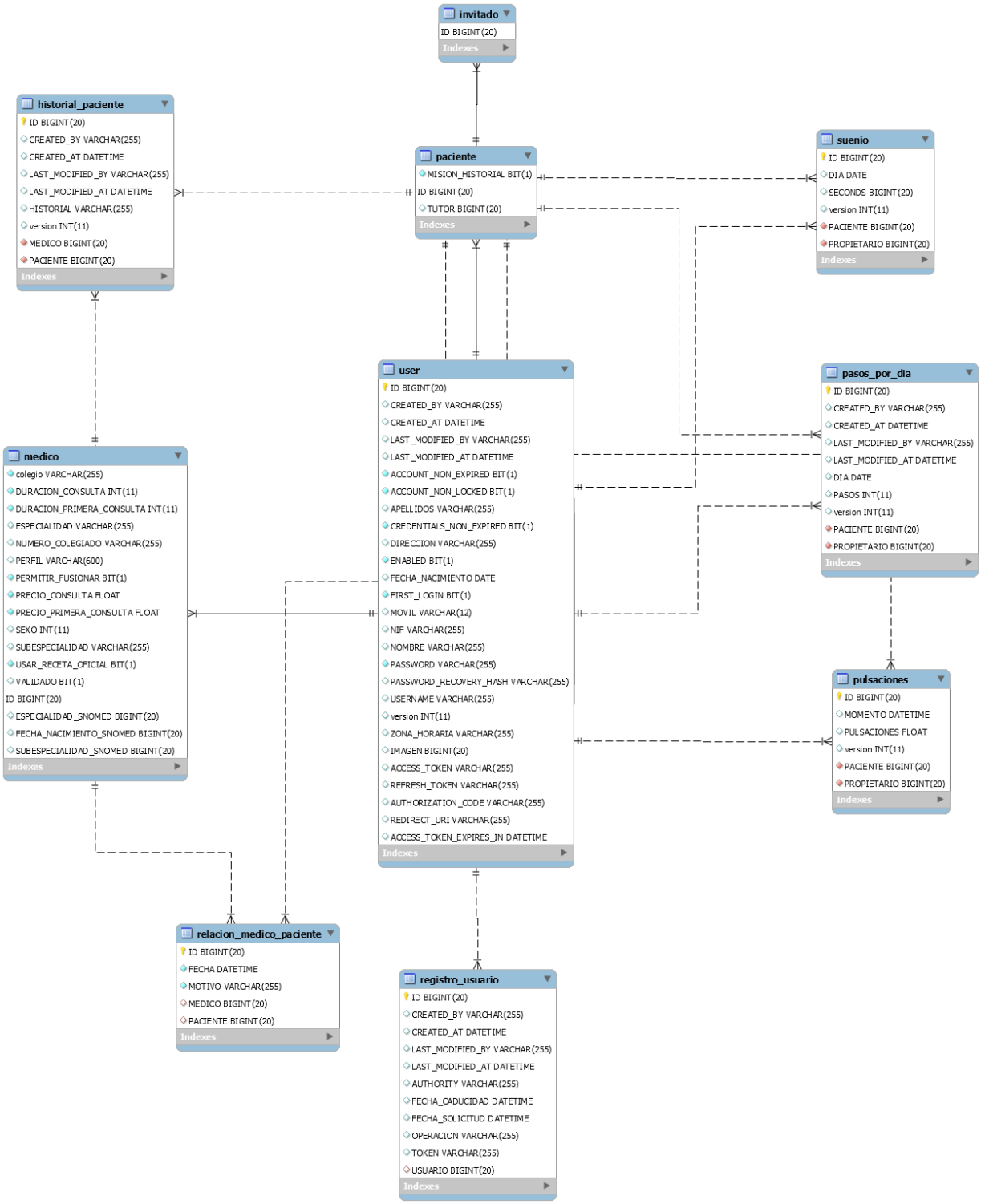
Se han utilizado los servicios REST de Google Fit para obtener la información sobre salud y bienestar de los usuarios almacenada dentro de los servidores de Google. Se accede a estos servicios a través de unos servicios creados en el proyecto BleesYou-service, pero son instanciados desde el front usando un controlador específico para ellos.

Los servicios de Google funcionan en varios pasos que se pueden entender de forma más detallada en este link a la consola OAuth2 de Google y son los siguientes:

1. Se solicitan permisos a Google para acceder a la información que necesitamos, en nuestro caso serían los necesarios para obtener los pasos, sueño y frecuencia cardiaca. Con esta llamada, Google nos devuelve un Authorization Code, a nosotros nos interesa guardar estos datos en la tabla de la base de datos del usuario.
2. Se solicita el Refresh Token y Access Token, usando el Authorization Code devuelto en la anterior llamada. El primero de ellos es necesario para refrescar el Access Token ya que tiene una validez de unos minutos. Los guardamos también en la base de datos.
3. Se solicita la información que nos interesa a Google, usando los servicios Rest que provee. Estos servicios necesitan el Access Token para comprobar los permisos y devolver la información. Tratamos la información como nos interesa, y la guardamos en la base de datos.

3.2 Diagrama de bases de datos

Aquí podemos ver un diagrama entidad-relación de la base de datos, que es un tipo de diagrama de flujo que ilustra cómo las "entidades", como personas, objetos o conceptos, se relacionan entre sí dentro de la base de datos, qué campos tiene cada una de esas entidades y de qué tipo son.



Diseño 5 – Diagrama de bases de datos

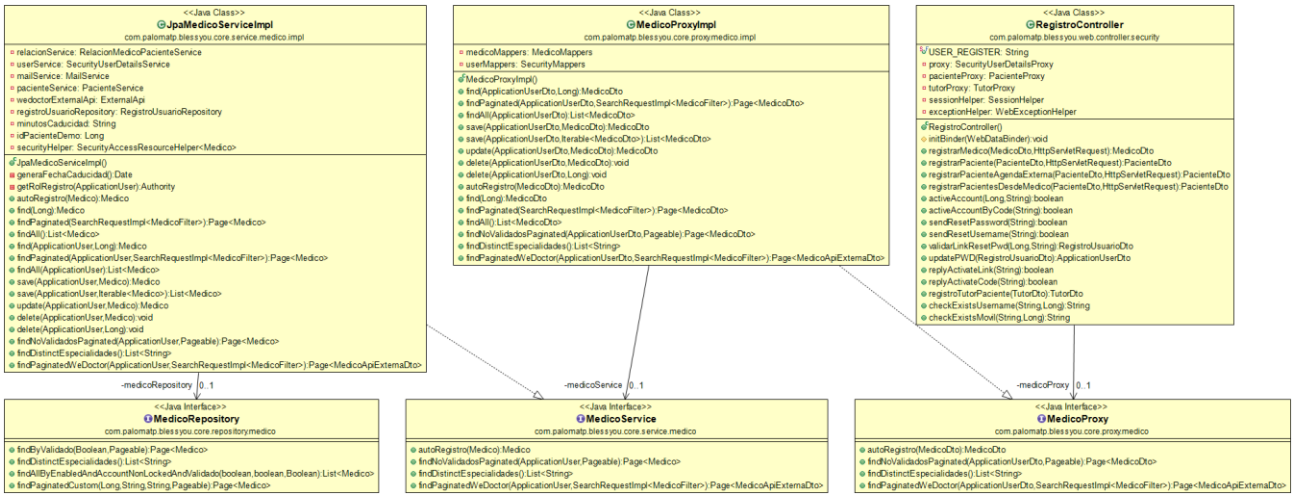
3.3 Diagramas de clases

En ingeniería de software, un diagrama de clases en Lenguaje Unificado de Modelado (UML), es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos. Podemos decir que es similar al diagrama mostrado en el apartado anterior, orientado a la base de datos, pero este tiene el dominio de las clases Java.

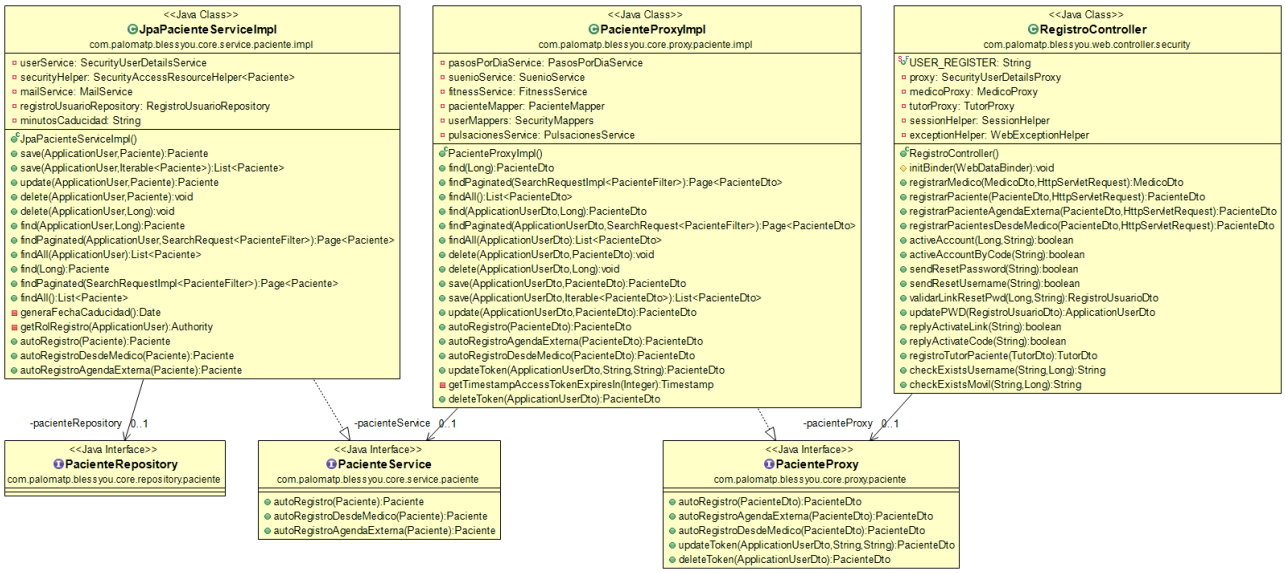
Se ha realizado ingeniería inversa, y los diagramas se han generado a partir de las clases Java usando ObjectAid [7] [8] en elipse [5].

3.3.1 Registro y Login

En este apartado se pueden ver las clases relacionadas con el servicio Rest correspondiente al sistema de Login y Registro, *RegistroController*.

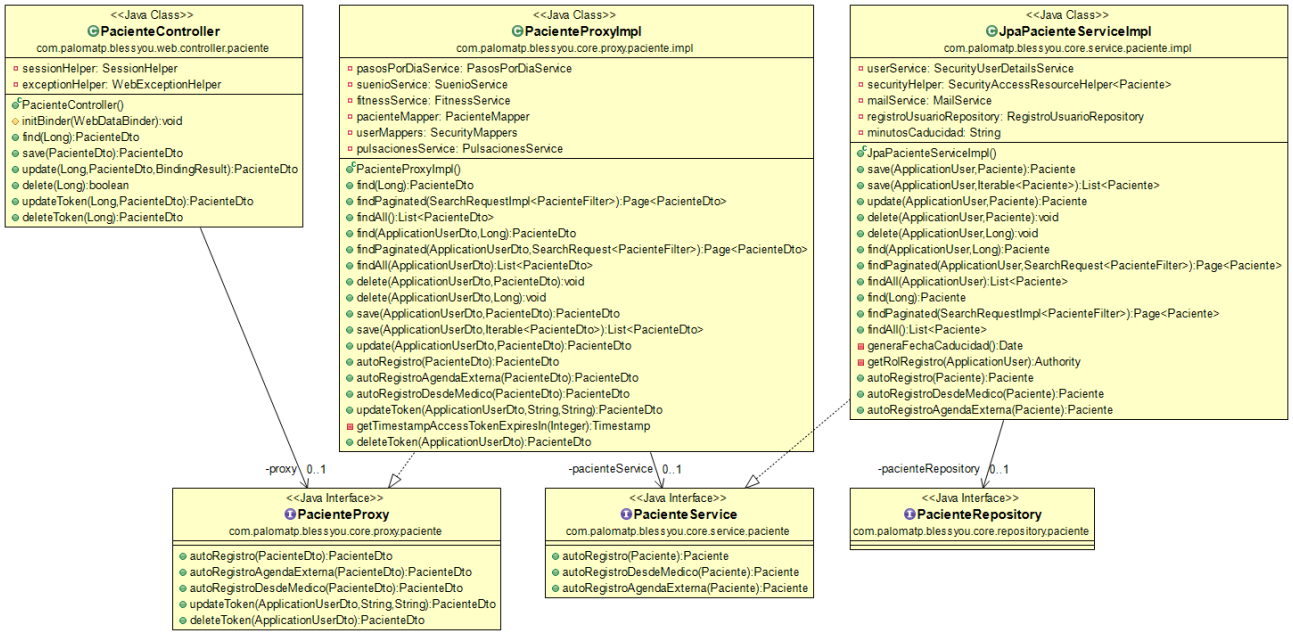


Diseño 6 – diagrama de clases de Registro y Login médico



Diseño 7 - diagrama de clases de Registro y Login paciente

3.3.2 Paciente

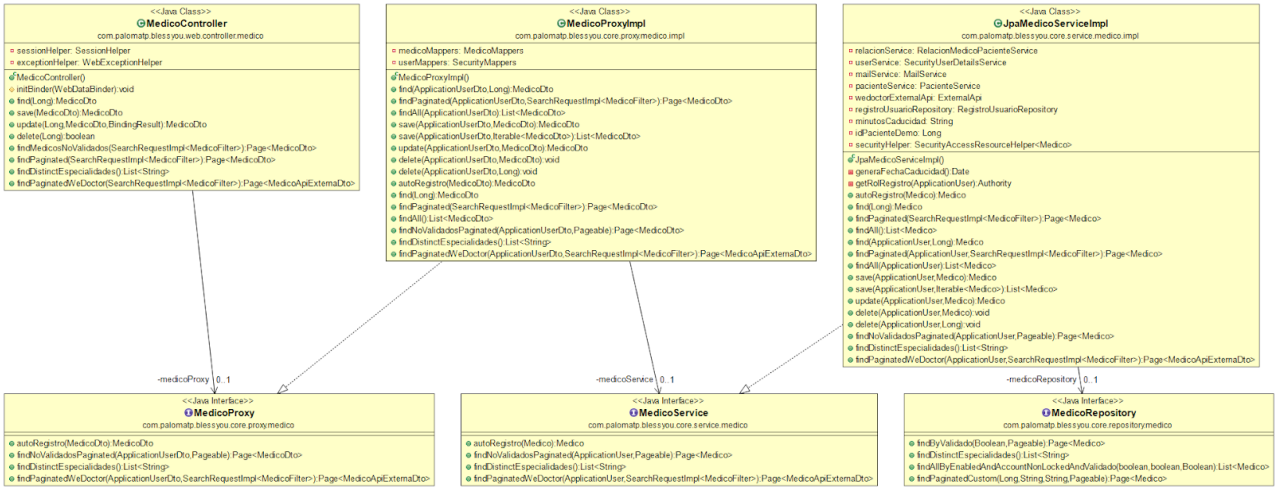


Diseño 8 – Diagrama de clases de paciente

En el diagrama anterior se pueden ver las clases relacionadas con *PacienteController*, que es el servicio Rest que se encarga de la gestión de pacientes, como la búsqueda, la devolución de los datos de un paciente, etc.

3.3.3 Medico

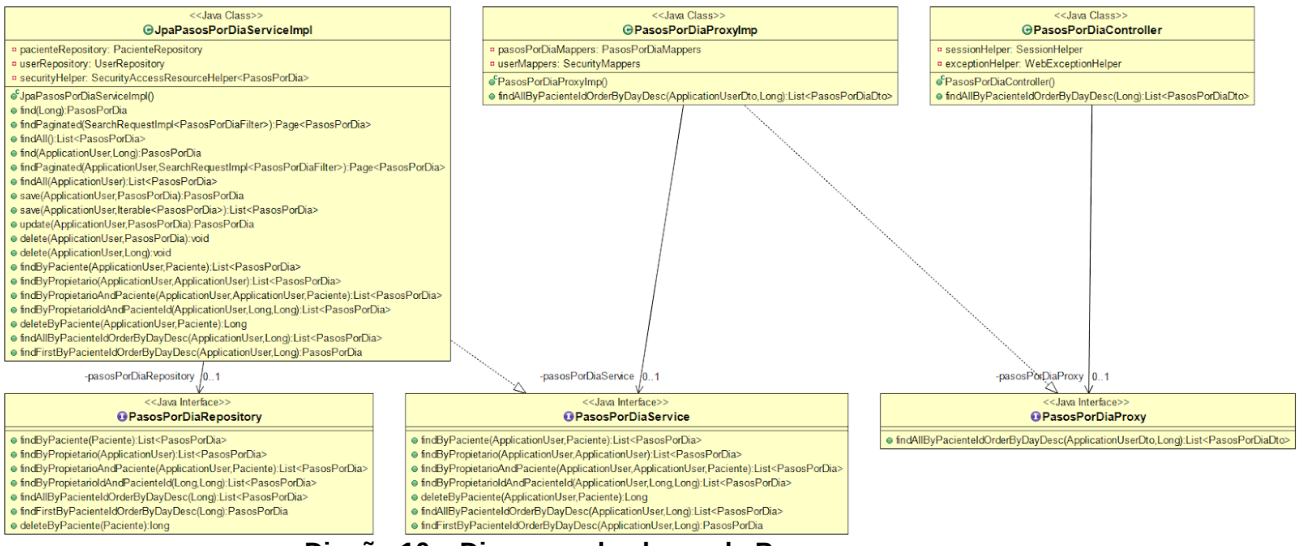
En este diagrama se pueden ver las clases relacionadas con *MedicoController*, que es el servicio Rest que se encarga de la gestión de médicos, como la búsqueda, la devolución de los datos de un paciente, etc.



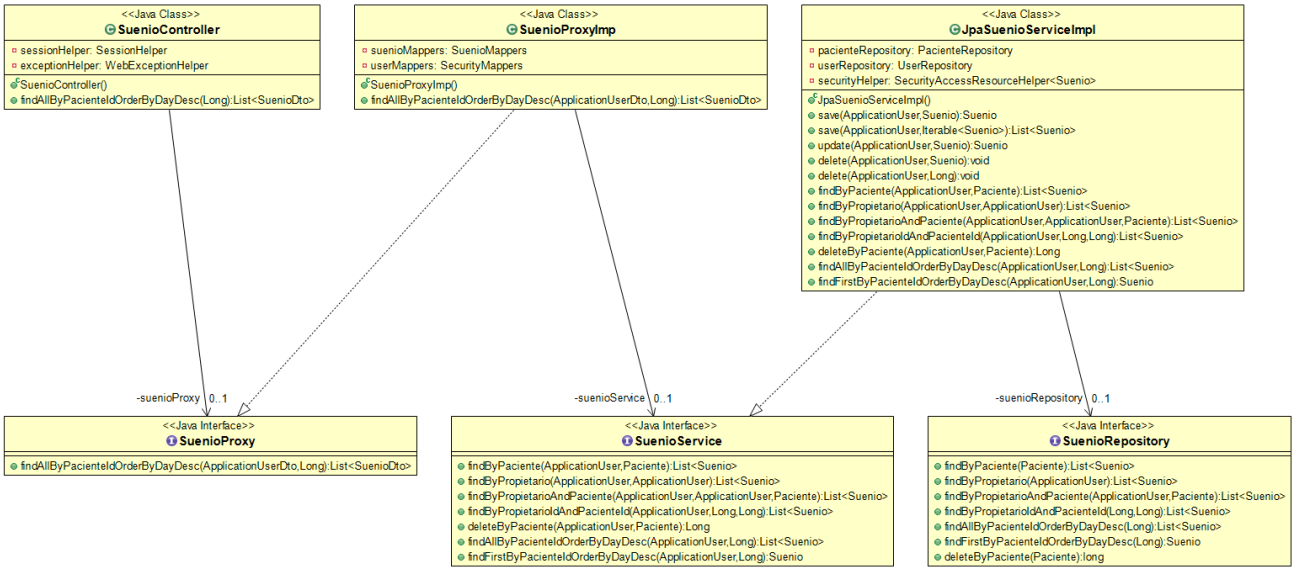
Diseño 9 - Diagrama de clases de médico

3.3.4 Pasos, Sueño y Pulsaciones

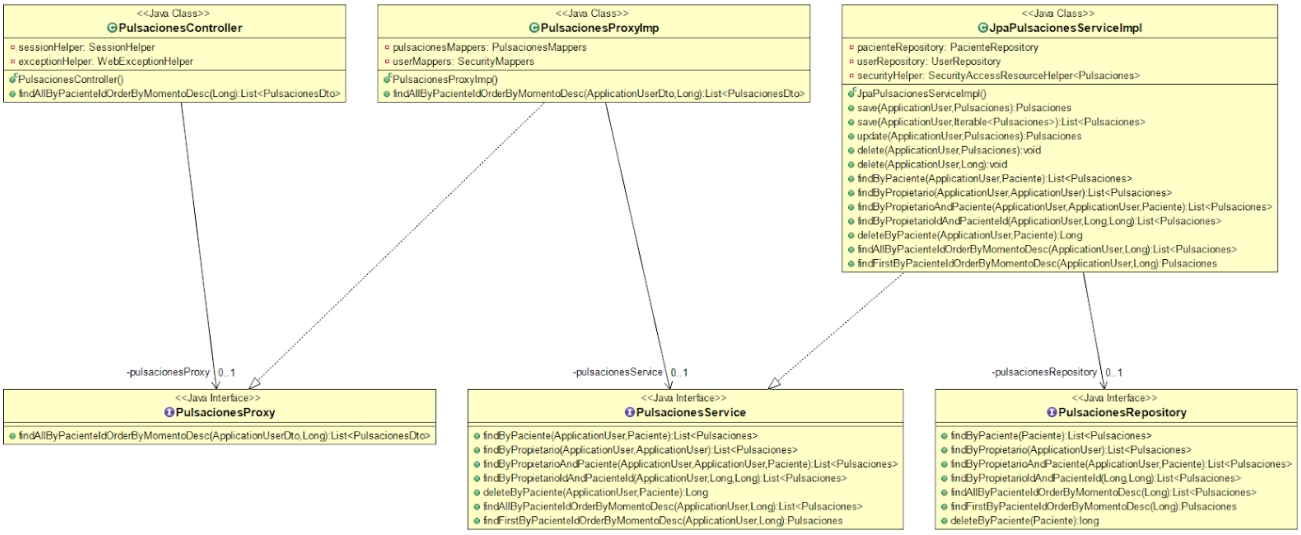
En los siguientes diagramas de clases se muestran la relación de clases Java que muestran los datos de Google Fit.



Diseño 10 – Diagrama de clases de Pasos



Diseño 11 – Diagrama de clases de Sueño



Diseño 12 – Diagrama de clases de pulsaciones

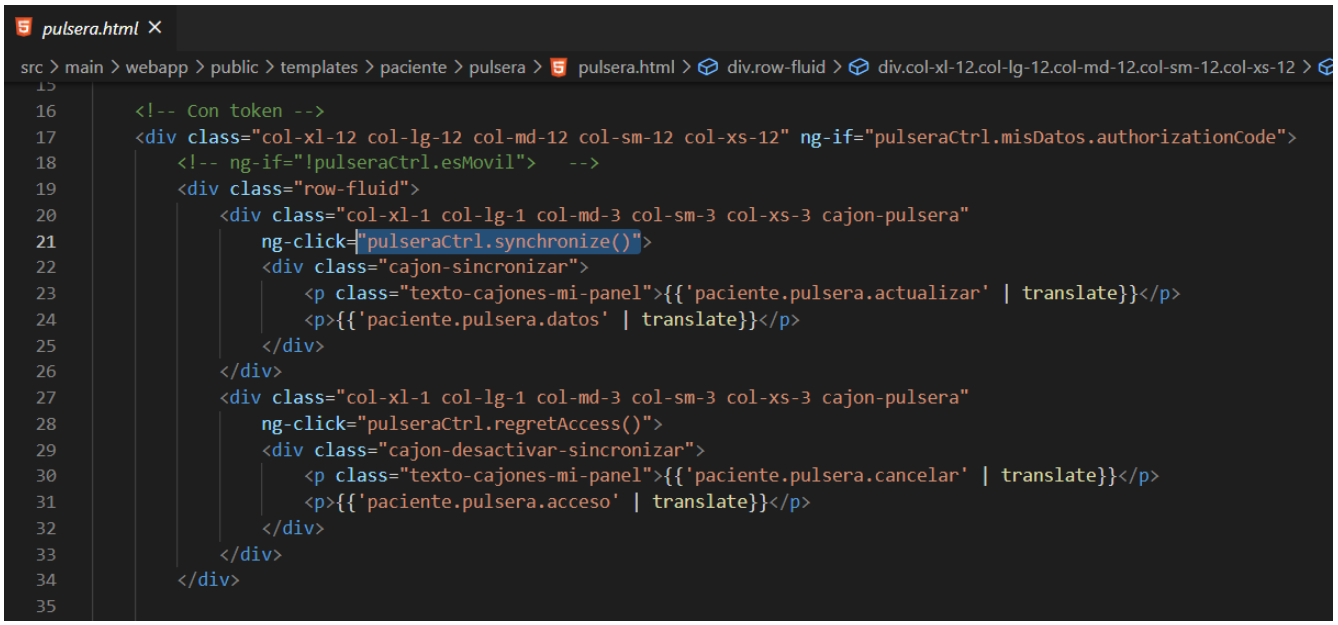
3.4 Ejemplo de flujo de código

Llegados a este punto ya podemos ver un ejemplo de una llamada a los servicios de Google Fit, la llamada al botón sincronizar. Este botón actualiza los datos de pulsaciones, sueño y pasos del paciente.

Diseño 13 – Captura de pantalla de actualizar datos de Google Fit

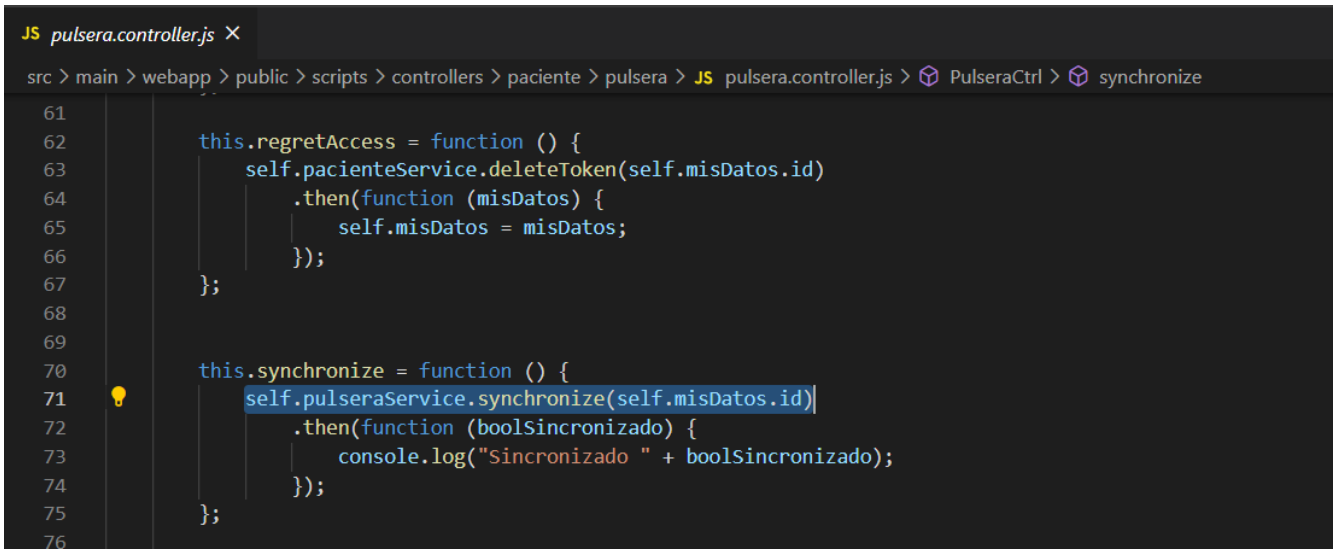
3.4.1 Aplicación Angular

En la aplicación Angular, tenemos declarado un HTML correspondiente con el componente pulsera. En este HTML, pulsera.html, llamamos a la función *synchronize* del componente:



Diseño 14 – pulsera.html

Dentro de la función *synchronize* del componente, se puede ver que se llama a otra función llamada de la misma forma del servicio PulseraService, y a este servicio le pasamos el id del paciente:



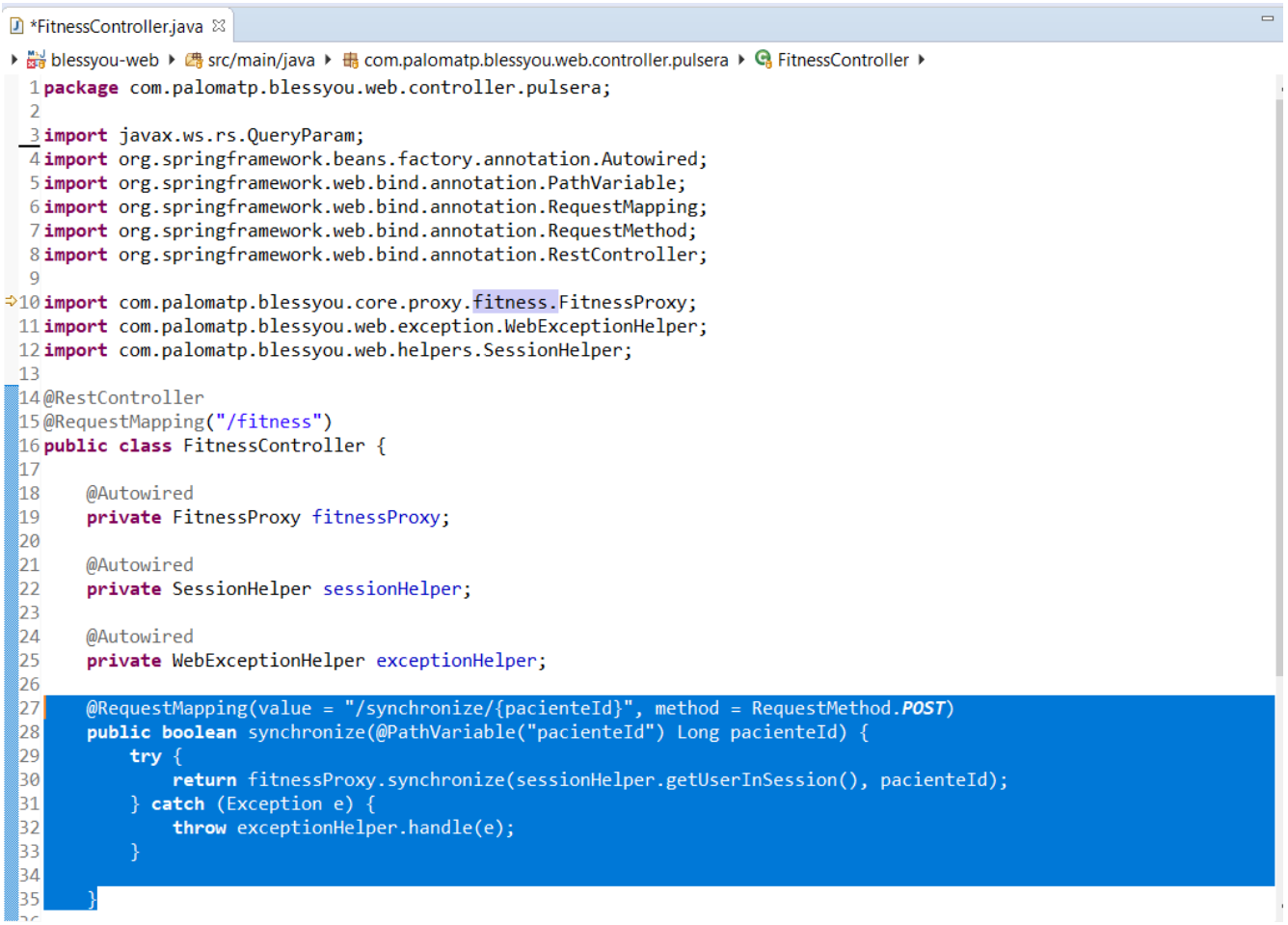
Diseño 15 - PulseraController

Dentro del servicio *PulseraService*, se llama al controlador Java, que es un servicio REST, mediante una petición http de *Angularjs* [2]:

Diseño 16 - PulseraService

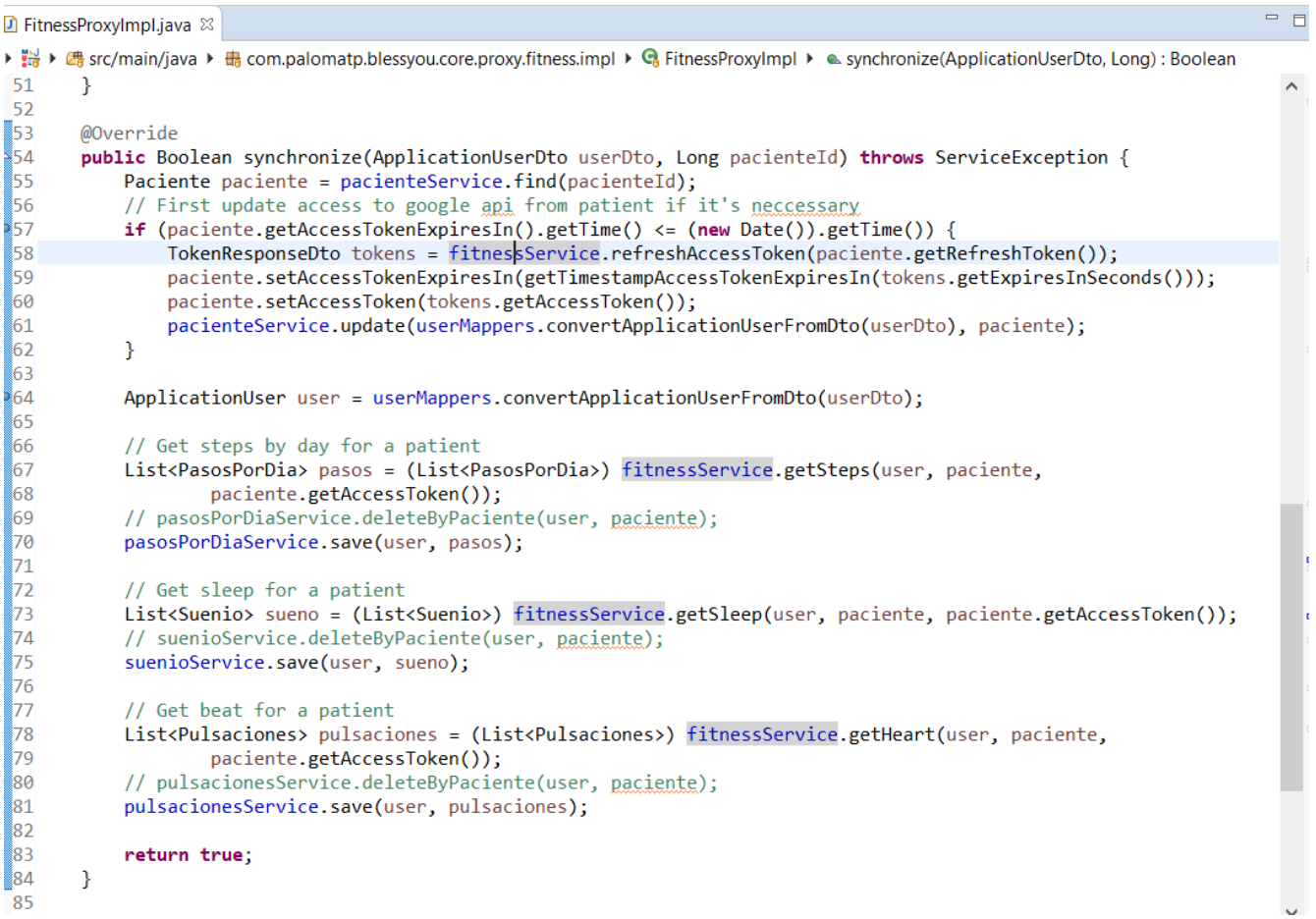
3.4.2 Aplicación Java

En primer lugar, veremos el servicio Rest al que se llama desde *Angularjs* [2], que se encuentra en el proyecto BleesYou-web:



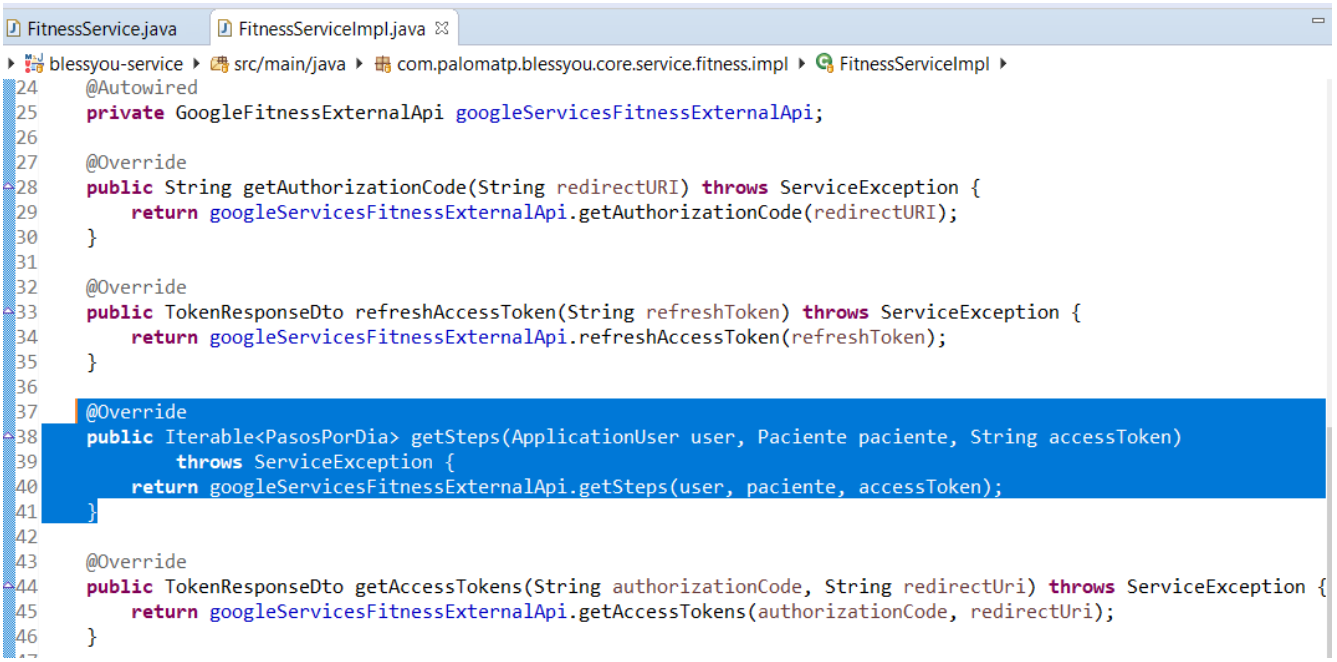
Diseño 17 – FitnessController.java

Este servicio únicamente llama a la capa proxy, que hace de intermediaria entre los Rest Services y los Servicios del proyecto BleesYou-service. Se puede observar que se utiliza la seguridad de Spring [3] para enviar el usuario que solicita la información al proxy y comprobar permisos:



Diseño 18 – FitnessProxyImpl.java

En la capa proxy se puede ver que en primer lugar se comprueba si *Access Token* ha caducado, y si es así los actualiza. No nos vamos a centrar en esta parte, nos vamos a centrar en los pasos. Una vez obtenido el nuevo token, si es necesario, llama al servicio *FitnessService* para obtener la información, en este caso nos interesa la función *getSteps*. Una vez obtenidos los pasos llama a la función save del servicio *PasosPorDíaService*.



Diseño 19 – FitnessServiceImpl.java

La función *getSteps*, únicamente llama a otro servicio, GoogleServiceFitnessExternalApi, que contiene las llamadas a los servicios REST de Google.

GoogleServiceFitnessExternalApi es el último servicio al que se llama para obtener la información que nos interesa de Google. En su función getSteps se construye la llamada al servicio REST de Google que nos devuelve los pasos mediante funciones privadas. La llamada a estas funciones privadas nos sirve para hacer el código más legible y mantenible, y nos facilita su reutilización.

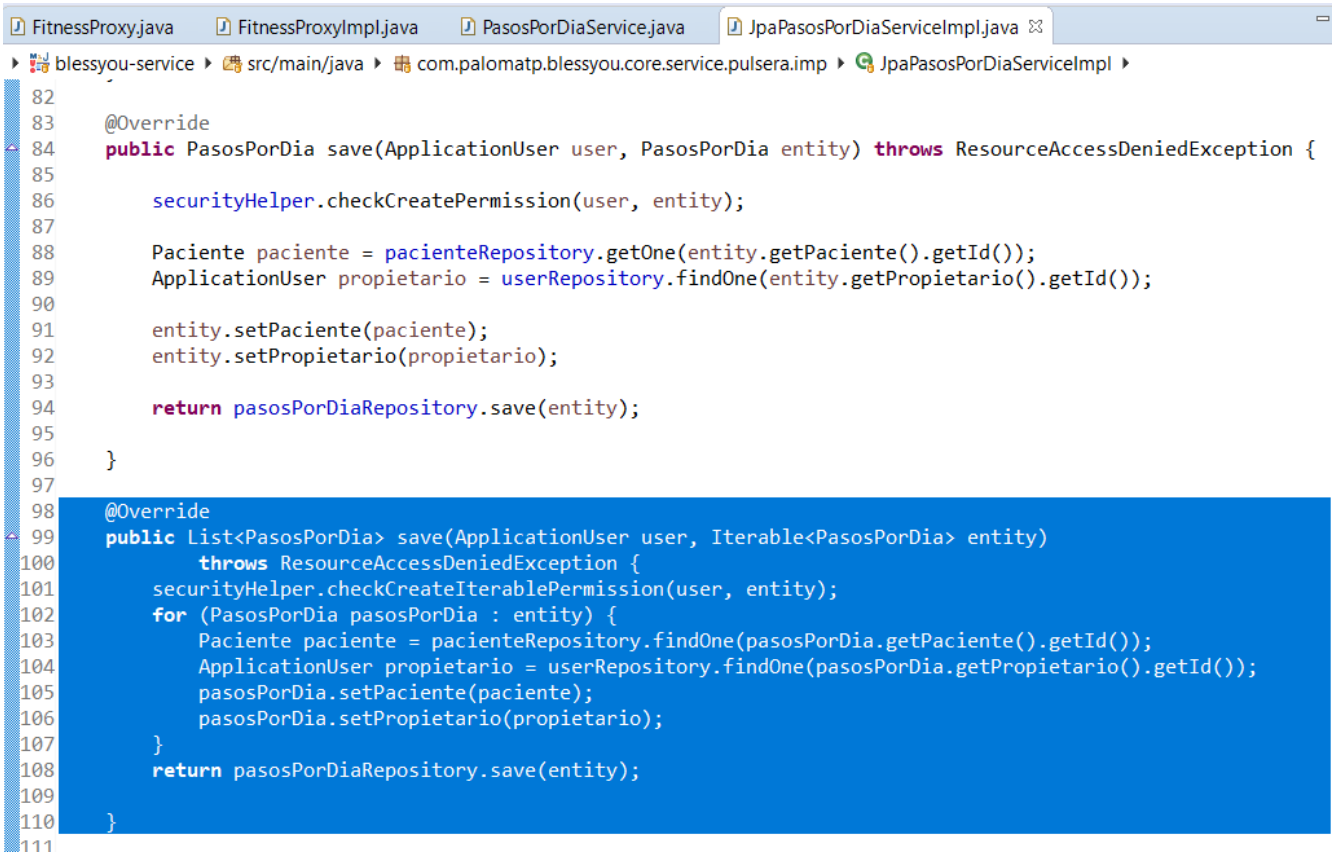
GoogleFitnessExternalApi.java | GoogleFitnessExternalApiImp.java

blessyou-service ▸ src/main/java ▸ com.palomatp.blessyou.core.external.apis ▸ GoogleFitnessExternalApiImp ▸

```java
301     /**
302      * Gets the steps.
303      *
304      * @param refreshToken
305      *            the refresh token
306      * @return the steps
307      * @throws ServiceException
308      *             the service exception
309      */
310     @Override
311     public Iterable<PasosPorDia> getSteps(ApplicationUser user, Paciente paciente, String accessToken)
312             throws ServiceException {
313         ObjectMapper mapper = new JacksonObjectMapper();
314         mapper.configure(Feature.ALLOW_UNQUOTED_FIELD_NAMES, true);
315 
316         JacksonJaxbJsonProvider jacksonProvider = new JacksonJaxbJsonProvider();
317         jacksonProvider.setMapper(mapper);
318 
319         Client client = ClientBuilder.newClient().register(jacksonProvider);
320 
321         AggregateRequest body = createBodySteps(user, paciente);
322 
323         WebTarget target = client.target(fitnessBaseUrl).path(stepsPath);
324 
325         AggregateResponseDto response = new AggregateResponseDto();
326         try {
327             response = target.request(MediaType.APPLICATION_JSON).header(AUTHORIZATION, BEARER + accessToken)
328                     .header(CONTENT_TYPE, MediaType.APPLICATION_JSON_TYPE)
329                     .post(Entity.json(body), new GenericType<AggregateResponseDto>() {});
330         } catch (Exception e) {
331             log.error(e.getMessage(), e);
332         }
333 
334         return getPasosPorDia(user, paciente, response);
335     }
```

Diseño 20 – GoogleFitnessExternalApiImpl.java

Anteriormente, hemos visto que en *FitnessProxyImpl*, una vez obtenidos los pasos, se llamaba al servicio *PasosPorDiaService* para guardar la información obtenida. Ahora vamos a ver este servicio:



Diseño 21 -JpaPasosPorDiaServiceImpl.java

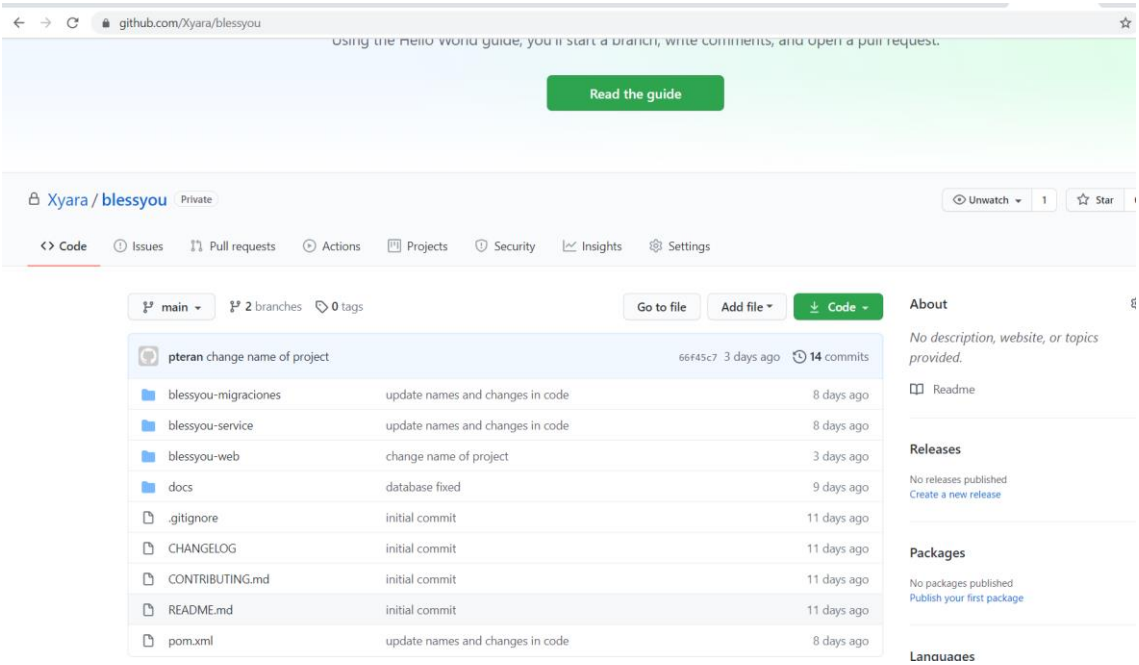
Primeramente, se puede ver que se comprueban los permisos de usuario por seguridad, a continuación, se obtiene el usuario correspondiente a los pasos por día, y se actualizan llamando a la función genérica del repositorio.

Se harían los mismos pasos con el sueño y las pulsaciones, y una vez finalizado todo el proceso, el Controlador o servicio Rest devolvería true, finalizando la llamada.

3.5 Control de versiones

Se ha creado un repositorio privado en **GitHub** sobre el que va subiendo el proyecto y su avance. GitHub es un portal creado para alojar el código de las aplicaciones que utiliza el sistema de control de versiones Git.

Git es un sistema de control de versión distribuida, que quiere decir que la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones.



Diseño 22 – Repositorio

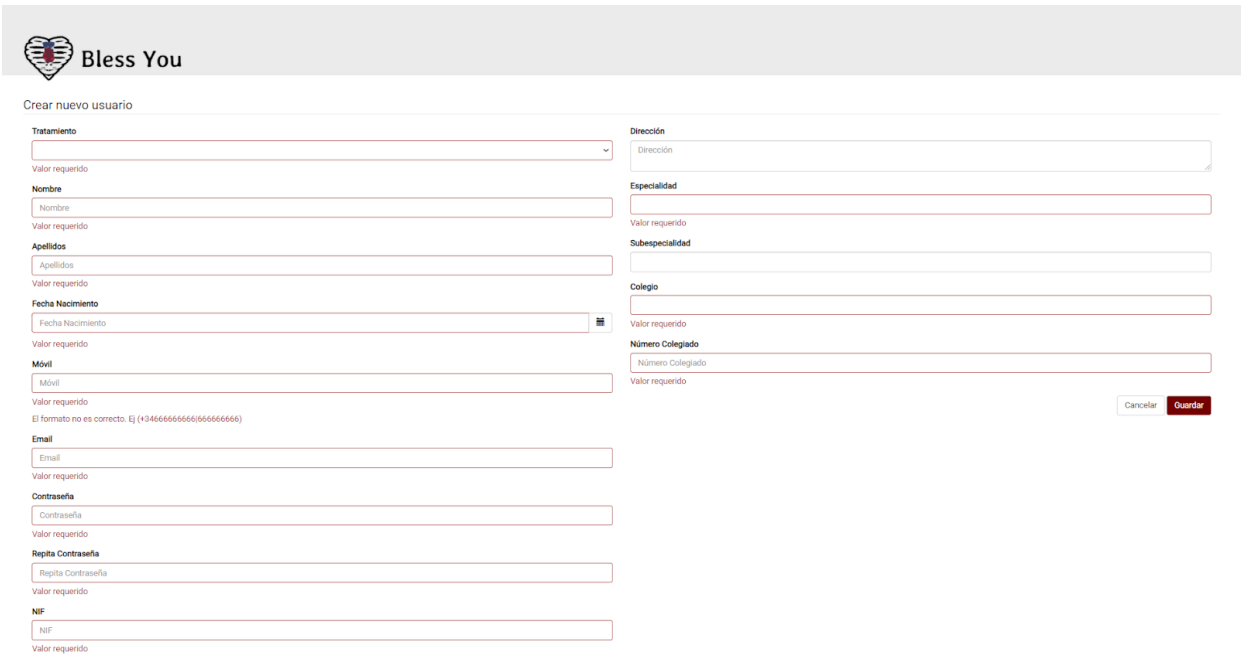
4 Pruebas caja negra

En este tipo de pruebas, se realizan test funcionales sin tener en cuenta la estructura interna del programa de software.

4.1 Registro

Se realizan varias pruebas en el subsistema de registro.

Test	T1
Requisitos	RF.1.1 - Los usuarios se podrán registrar como médico o paciente. RF.1.6 - Durante el registro, si un dato no es válido, se informará al usuario del error.
Entrada	Datos de registro vacíos
Prueba	Se accede a la aplicación, y nos intentamos registrar como médico dejando campos sin rellenar.
Resultado	No permite el registro y se muestran los errores en la pantalla

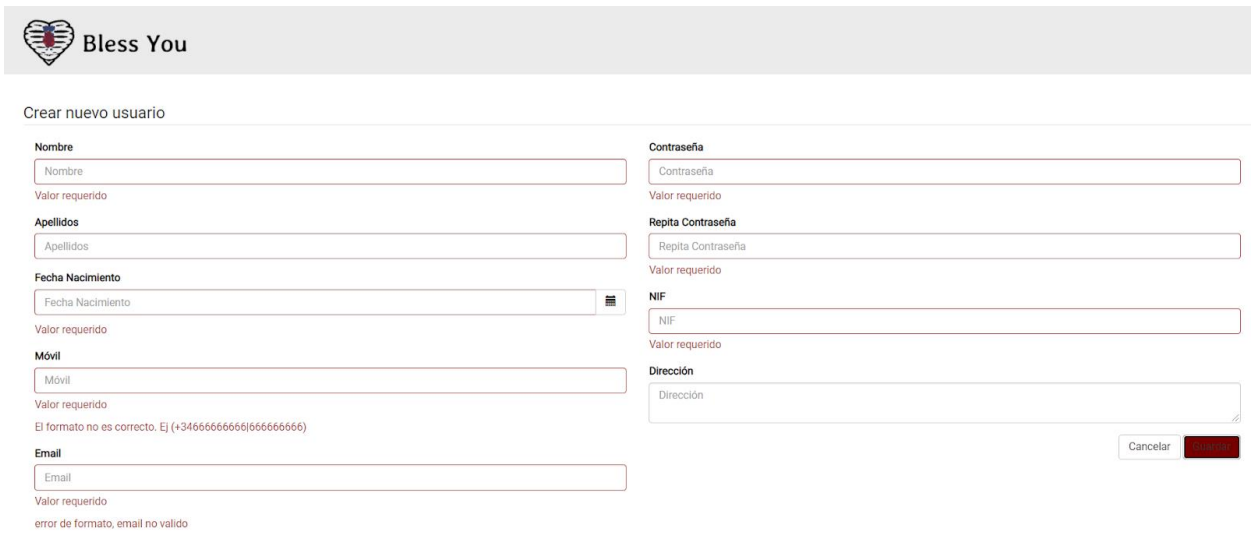


Prueba 1

Test	T2
Requisito	RF.1.1 - Los usuarios se podrán registrar como médico o paciente.
Entrada	Datos correctos de registro
Prueba	Se accede a la aplicación, y nos intentamos registramos como médico.
Resultado	Se muestra confirmación de registro.

Prueba 2

Test	T3
Requisito	RF.1.1 - Los usuarios se podrán registrar como médico o paciente. RF.1.6 - Durante el registro, si un dato no es válido, se informará al usuario del error.
Entrada	Datos de registro vacíos
Prueba	Se accede a la aplicación, y nos intentamos registrar como paciente dejando campos sin rellenar.
Resultado	No permite el registro y se muestran los errores en la pantalla



Prueba 3

Test	T4
Requisito	RF.1.1 - Los usuarios se podrán registrar como médico o paciente.
Entrada	Datos correctos de registro
Prueba	Se accede a la aplicación, y nos intentamos registramos como paciente.
Resultado	Se muestra confirmación de registro.

Prueba 4

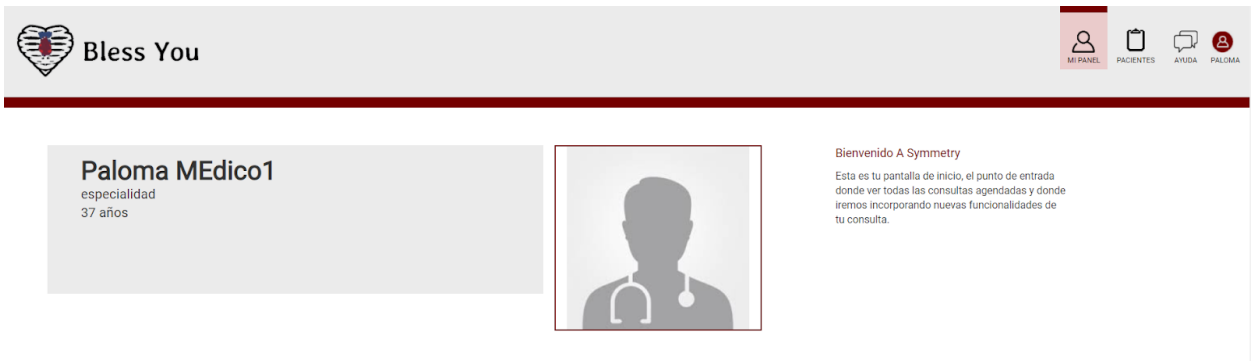
4.2 Login

El subsistema de login es sencillo de probar.

Test	T5
Requisitos	RF.1.1 - El sistema dispondrá de un sistema de login. RF.1.3 - Si hay un error durante el login, se mostrará el error en la pantalla.
Entrada	Datos de incorrectos de login
Prueba	Se accede a la aplicación, y nos intentamos loguear como médico.
Resultado	No permite el login y se muestra el error en pantalla

Prueba 5

Test	T6
Requisitos	RF.1.1 - El sistema dispondrá de un sistema de login.
Entrada	Datos de correctos de login
Prueba	Se accede a la aplicación, y nos logueamos como médico.
Resultado	Nos logueamos y la aplicación nos muestra la página de inicio del usuario



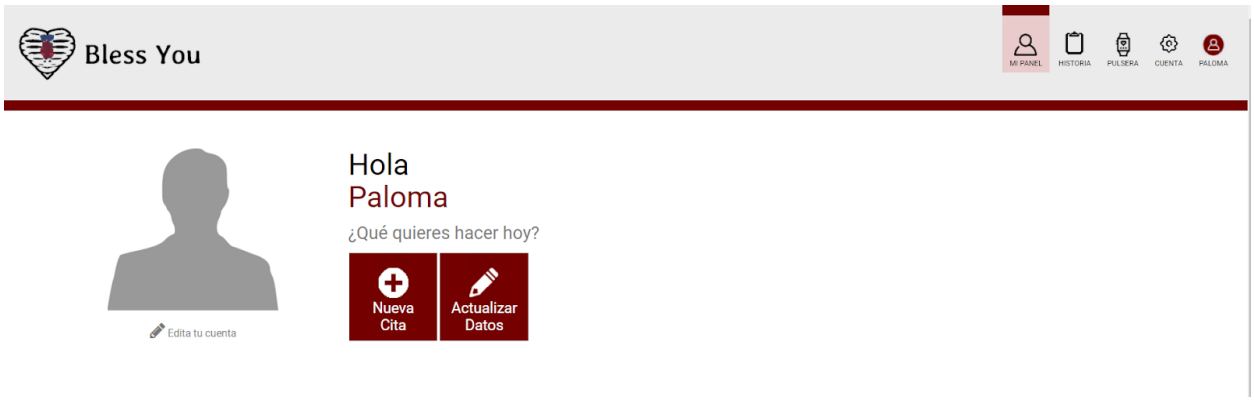
Prueba 6

Test	T7
Requisitos	RF.1.1 - El sistema dispondrá de un sistema de login. RF.1.3 - Si hay un error durante el login, se mostrará el error en la pantalla.
Entrada	Datos de incorrectos de login
Prueba	Se accede a la aplicación, y nos intentamos loguear como paciente.
Resultado	No permite el login y se muestra el error en pantalla

Prueba 7

Test	T8
Requisitos	RF.1.1 - El sistema dispondrá de un sistema de login.

Entrada	Datos de correctos de login
Prueba	Se accede a la aplicación, y nos logueamos como paciente.
Resultado	Nos logueamos y la aplicación nos muestra la página de inicio del usuario

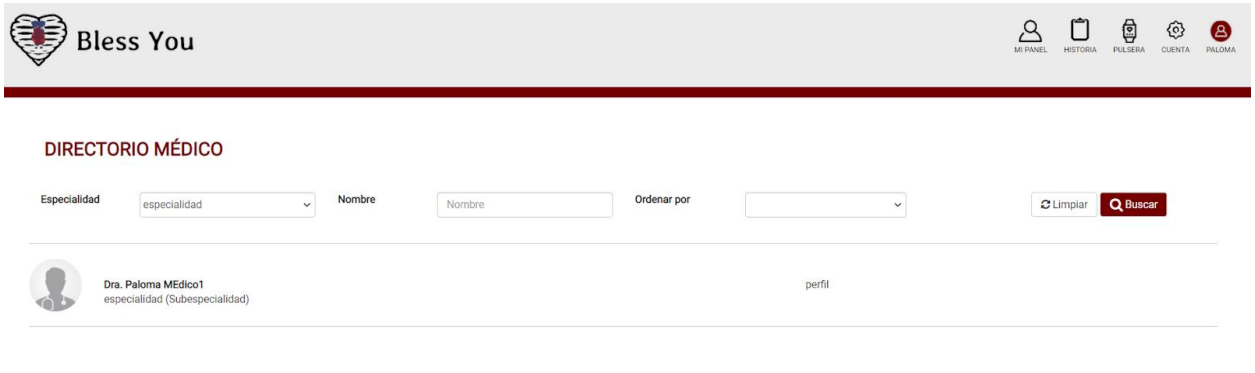


Prueba 8

4.3 Contacto entre médicos y usuarios

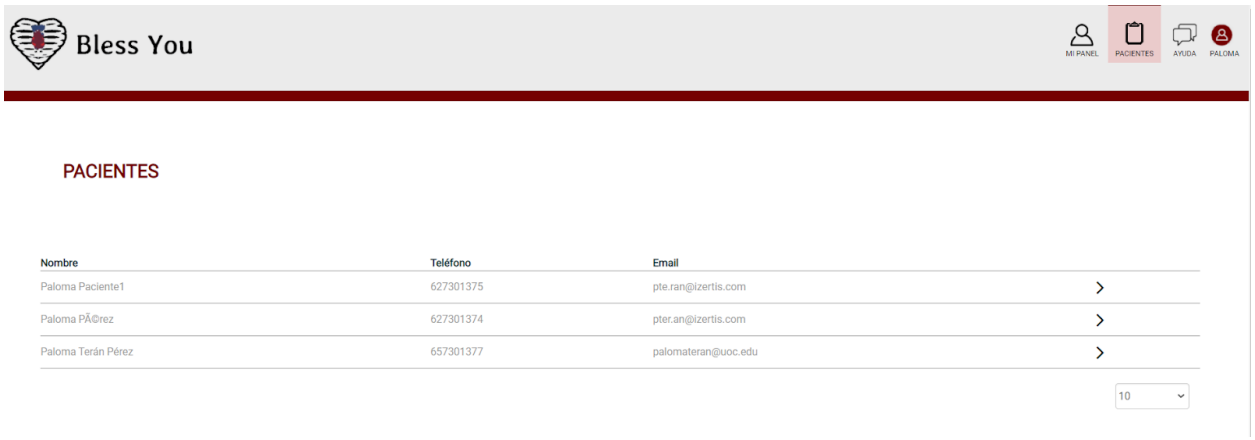
En este apartado se prueban listados desde médico y paciente y consulta de datos.

Test	T9
Requisitos	RF2.1 - Un paciente podrá buscar médicos por especialidad o nombre. RF2.2 - El listado de médicos mostrará su especialidad y nombre.
Entrada	Datos de filtro
Prueba	Se accede a la aplicación como paciente, se va a la sección directorio médico y se hacer una búsqueda
Resultado	La aplicación muestra los médicos que cumplen los filtros.



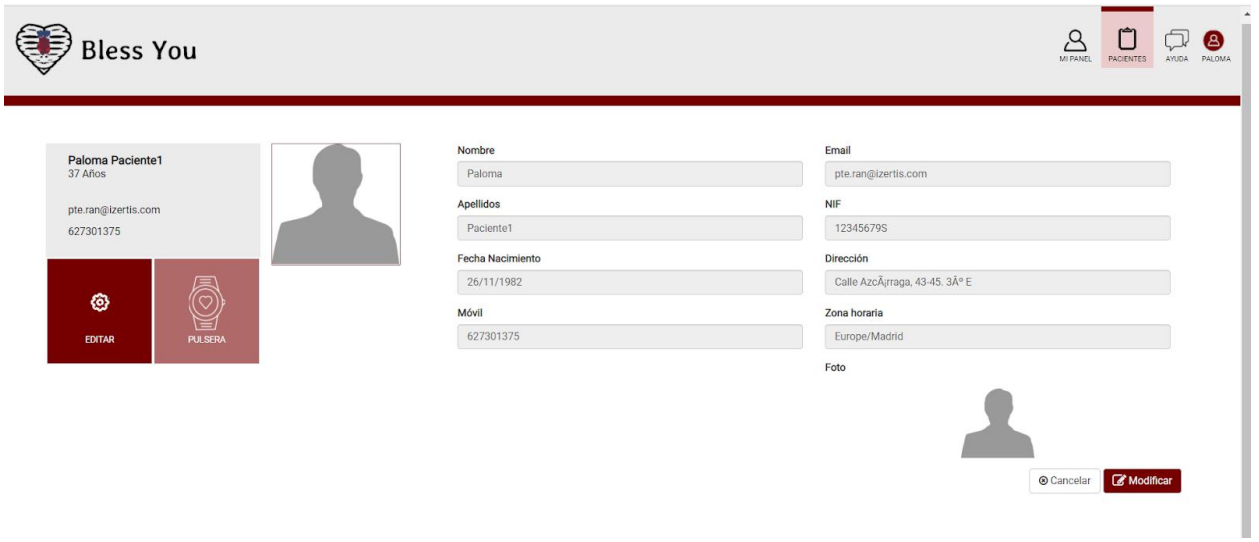
Prueba 9

Test	T10
Requisitos	RF2.3 - Un médico podrá buscar pacientes.
Entrada	Datos de filtro
Prueba	Se accede a la aplicación como médico, se va a la sección pacientes y se hace una búsqueda.
Resultado	La aplicación muestra los pacientes que cumplen con los requisitos del filtro



Prueba 10

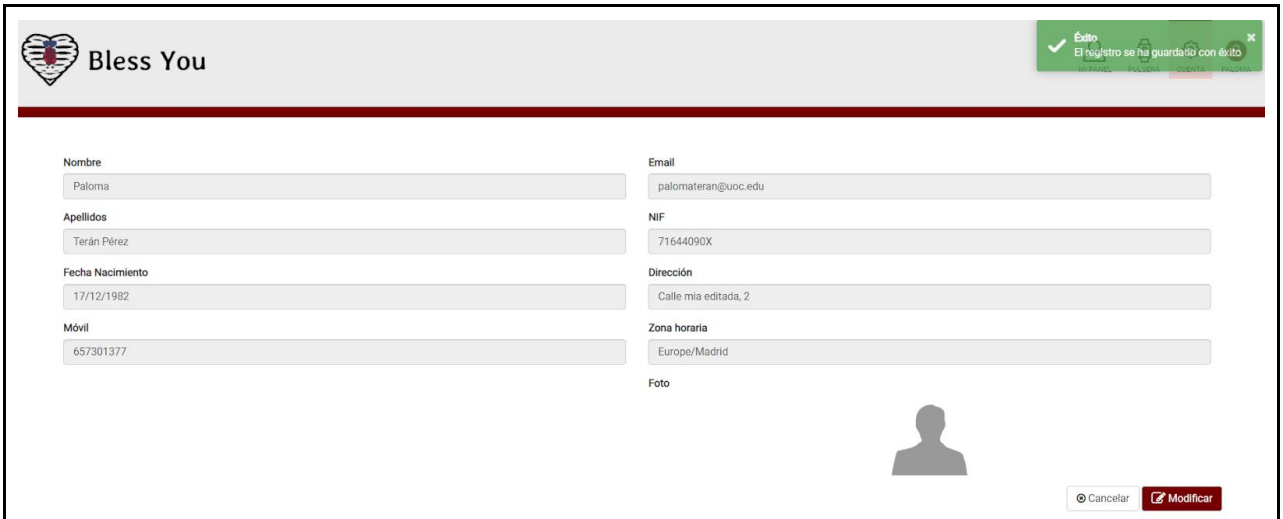
Test	T11
Requisitos	RF2.4 - Un médico podrá ver el detalle de un paciente y acceder a su perfil y datos de Google fin.
Entrada	Buscador de pacientes
Prueba	Se accede a la aplicación como médico, se va a la sección pacientes y se

	hace una búsqueda, a través de los resultados se accede a un paciente.
Resultado	La aplicación muestra los detalles de un paciente.
 **Prueba 11**	

4.4 Perfiles de usuario

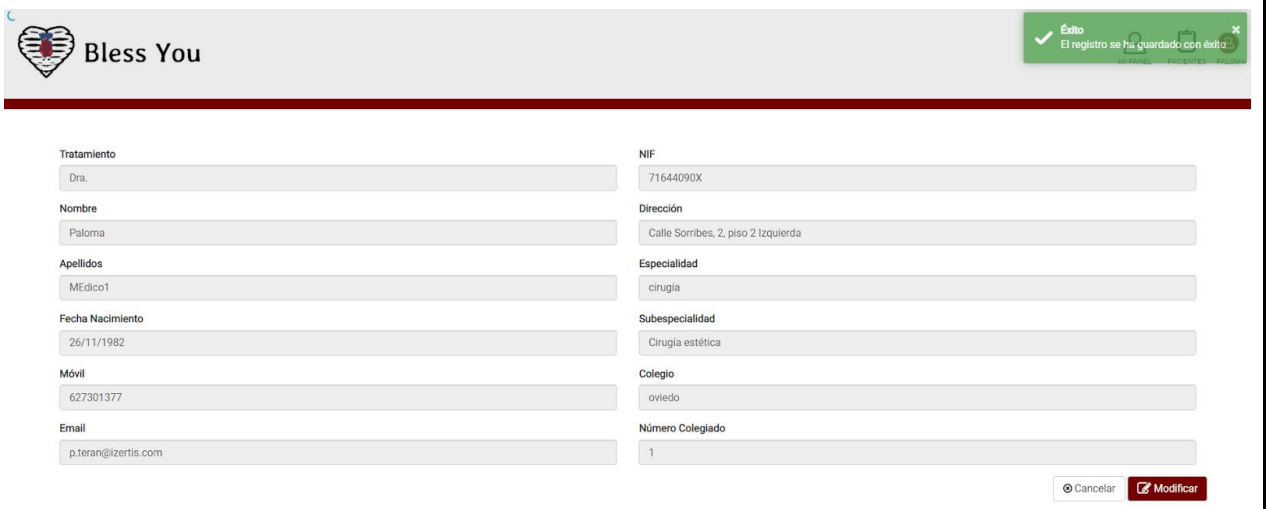
En este apartado se prueba la edición y consulta de datos de un usuario.

Test	T12
Requisitos	RF.3.1 - Un paciente podrá modificar su perfil
Entrada	Nos encontramos en el perfil de paciente
Prueba	Se accede a la aplicación como paciente, se va a la sección cuenta y se edita un dato.
Resultado	La aplicación muestra los datos actualizados



Prueba 12

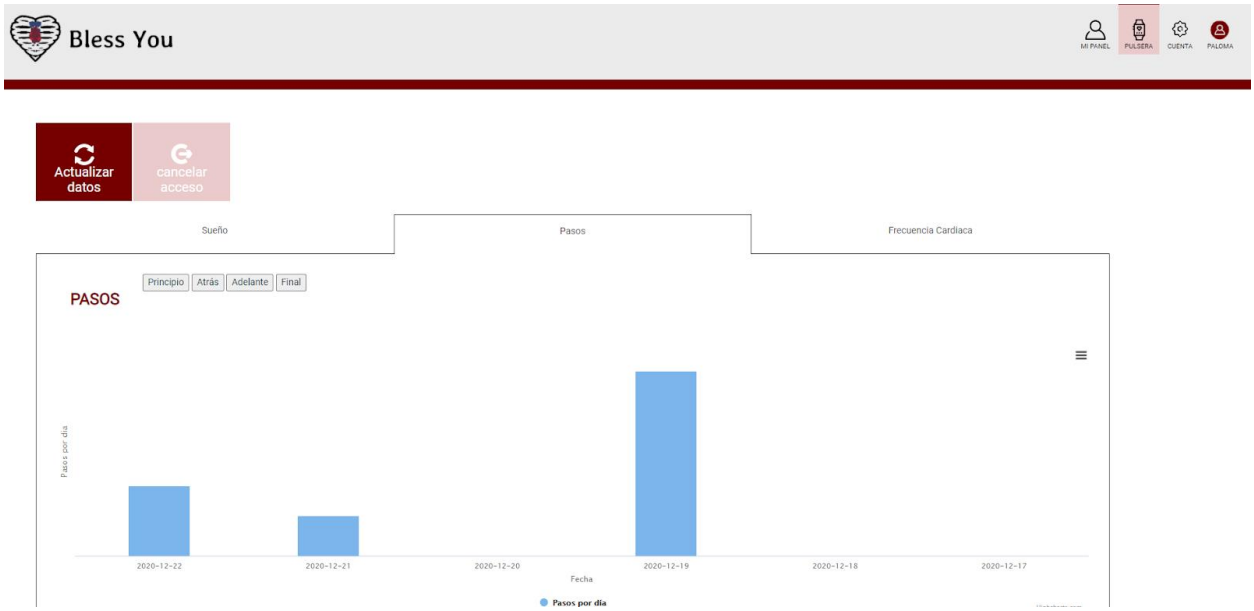
Test	T13
Requisitos	RF.3.2 - Un médico podrá modificar su perfil.
Entrada	Nos encontramos en el perfil de médico
Prueba	Se accede a la aplicación como médico, se va a la sección con el nombre del médico y se edita un dato.
Resultado	La aplicación muestra los datos actualizados



Prueba 13

Test	T14
Requisitos	RF.3.4 - Un paciente podrá sincronizar sus datos de Google Fit con la aplicación

	RF.3.5 - Los datos sincronizados de Google Fit se mostrarán en el perfil del paciente RF.3.6 - El paciente podrá sincronizar los datos del sueño. RF.3.7 - El paciente podrá sincronizar los datos de pasos. RF.3.8 - El paciente podrá sincronizar los datos de su frecuencia cardiaca.
Entrada	Nos encontramos en el perfil de paciente
Prueba	Se accede a la aplicación como paciente, se va a la sección pulsera y se sincroniza.
Resultado	Se aceptan los permisos y los datos aparecen en la aplicación.

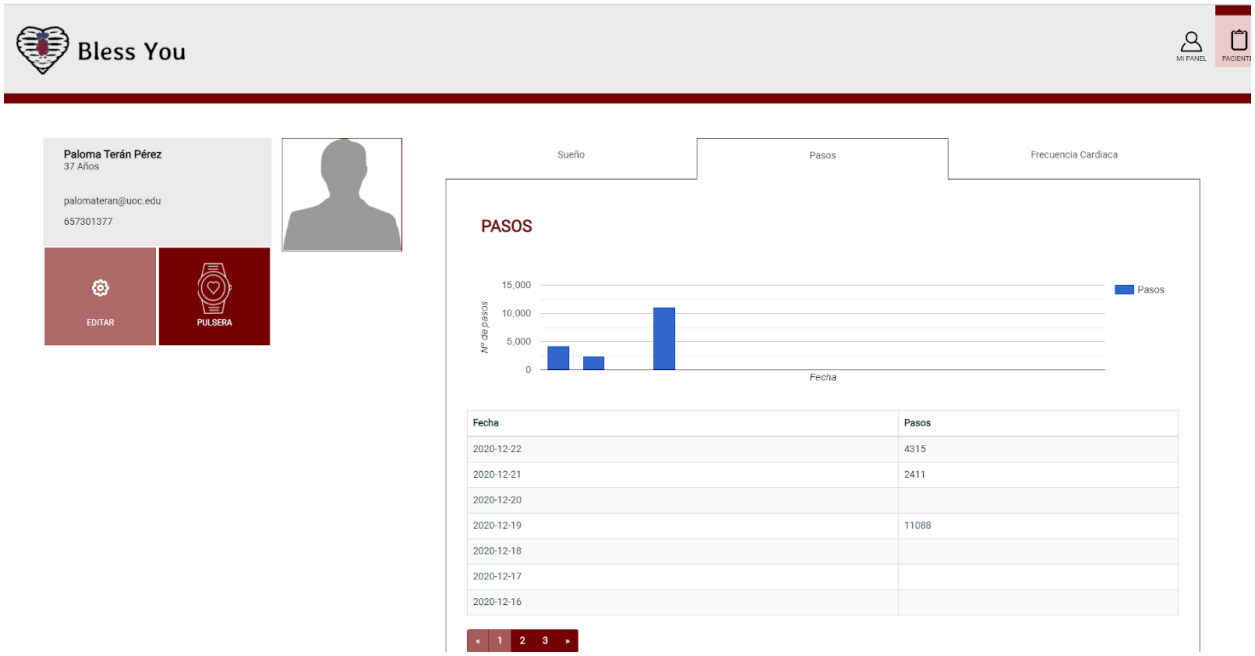


Prueba 14

Test	T15
Requisitos	RF.3.4 - Un paciente puede actualizar sus datos de Google Fit. RF.3.5 - Los datos sincronizados de Google Fit se mostrarán en el perfil del paciente RF.3.6 - El paciente podrá sincronizar los datos del sueño. RF.3.7 - El paciente podrá sincronizar los datos de pasos. RF.3.8 - El paciente podrá sincronizar los datos de su frecuencia cardiaca.
Entrada	Nos encontramos en el perfil de paciente
Prueba	Se accede a la aplicación como paciente y accede a la sección pulsera para actualizar los datos.
Resultado	Los datos de Google Fit se han actualizado

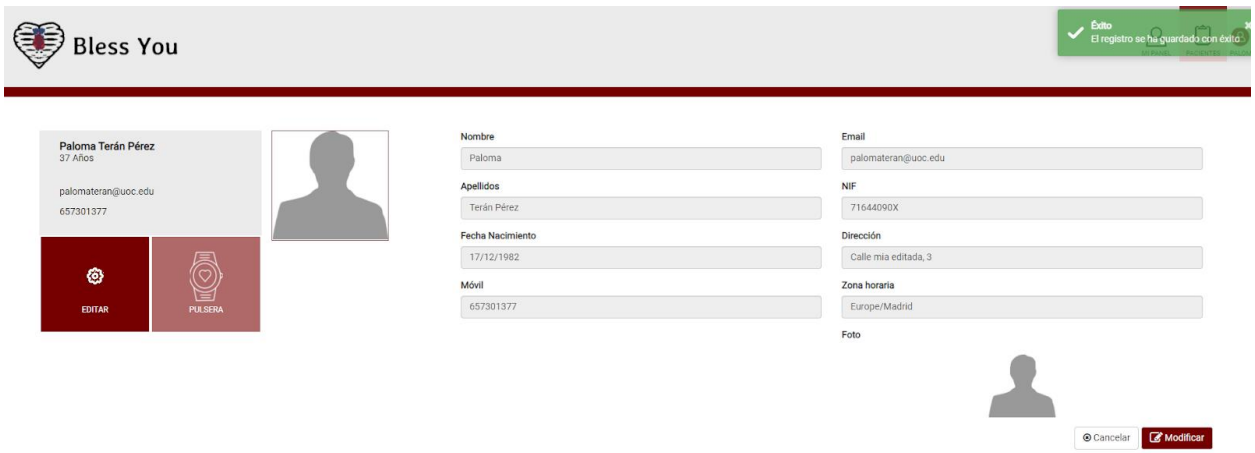
Prueba 15

Test	T16
Requisitos	RF.3.9 - Un médico podrá consultar los datos de Google Fit de su paciente
Entrada	Nos encontramos en el perfil de médico
Prueba	Se accede a la aplicación como médico, busca a un paciente y accede a su perfil, visualiza los datos de Google Fit
Resultado	El médico ve los datos de Google Fit del paciente



Prueba 16

Test	T17
Requisitos	RF.3.10 - Un médico podrá modificar el perfil de sus pacientes.
Entrada	Nos encontramos en el perfil de médico
Prueba	Se accede a la aplicación como médico, busca a un paciente y accede a su perfil, y edita sus datos
Resultado	El médico ve los datos del paciente actualizados.



Prueba 17

5 Conclusiones

5.1 Conclusiones personales

Nos encontramos en un año “particular” debido a la pandemia que estamos sufriendo por culpa del Covid, esto me ha motivado bastante en la realización del proyecto, a parte de la motivación de finalizar, POR FIN, los últimos créditos para poder obtener el título, pero también ha hecho que me resultara más difícil, pero no imposible, encontrar el tiempo que quería invertir en la realización de este proyecto.

Personalmente, ya llevo en el mundo de la informática más de 20 años: Empecé a estudiar la ingeniería técnica en el año 2000, en la universidad presencial, y la abandoné por empezar a trabajar y por falta de motivación. Precisamente he encontrado esta motivación años más tarde, no en el mejor momento, pero así funcionamos algunos.

A pesar de esta experiencia, la realización de este proyecto me ha resultado muy satisfactoria, ya que me ha servido para darme cuenta del potencial que tengo, que tal vez no esté aprovechando en mi actual empleo.

5.2 Conclusiones sobre el proyecto

La aplicación web, ha quedado sencilla, y usable, lo cual es verdaderamente importante hoy en día en una aplicación, para que el aprendizaje de su uso sea fácil y rápido.

Además, la estructura del código, has quedado bastante organizada y entendible, facilitando el desarrollo de futuras mejoras y nuevas implementaciones. A parte de esto, las librerías usadas han sido escogidas de forma correcta, ya que son librerías de las que se puede obtener muchísima información por internet, por tener documentación muy bien redactada, y por tener muchos usuarios, que hace que se encuentren respuestas a las dudas que tengamos fácilmente por *Stackoverflow*.

A grandes rasgos se han alcanzado los objetivos, se ha realizado todo lo planificado, pero me hubiera gustado invertir más tiempo en el desarrollo de la aplicación, como puede ser:

- Implementar diseño responsive.
- Implementación de test unitarios, aunque la aplicación está preparada para la realización de estos test.

5.3 Conclusiones sobre la planificación y metodología

La planificación del proyecto ha sido bastante correcta, aunque he tenido que modificar la planificación de la primera PEC debido a

problemas personales relacionados con el Covid y la familia.

El seleccionar una metodología ágil genérica, ha sido correcto, ya que me ha servido para organizar los bloques de trabajo de una forma muy organizada. Además, esta metodología junto con Kanban, y el tablero Trello [4], me han motivado durante la realización del proyecto, ya que iba viendo los avances y cómo se iba reduciendo la lista de tareas pendientes.

5.4 Posibles mejoras

La aplicación BleesYou generada en estos cuatro primeros Sprints es muy sencilla y fácil de usar. Se le pueden incorporar varias mejoras siguiendo distintos rumbos, según la importancia que se le quiera dar a cada uno de ellos.

Estas mejoras podrían ser:

- Mejoras en el diseño de la aplicación:
    - **Diseño responsive**: O lo que es lo mismo, preparación de la aplicación para su visualización correcta en todo tipo de dispositivos. Bootstrap [1] ya viene preparado para realizar diseño responsive, sólo habría que revisar las páginas implementadas y reparar las que no se vieran correctamente en distintos tamaños de pantalla.
    - **Mejora de su accesibilidad**: Ya que es una aplicación con fines médicos, podría ser muy útil mejorar su accesibilidad con mejoras para personas ciegas o con discapacidades.
    - **Mejora en las funcionalidades con conectividad entre médicos y pacientes**: Por ejemplo, que un médico tenga asignados sólo unos pacientes, y sólo pueda ver estos pacientes, y no todos los de la aplicación.
    - **Realización de test**: Realización de test unitarios y de integración, tanto en Angularjs [2] como en Java, a partir de los requisitos. Estos test, además de probar las funcionalidades, servirían para comprobar que, con futuras ampliaciones o modificaciones, las funcionalidades ya implementadas siguen haciendo su labor correctamente.

- Implementación de nuevas funcionalidades:
    - **Implementación de un calendario para la gestión de citas**: Habría que buscar una librería sencilla, apta para el diseño responsive y accesible, en la que implementar una gestión de citas médico-paciente, para que un paciente pudiera solicitar una cita o un médico reservar una cita con un paciente.
    - **Implementación de una pasarela de pago**: Una vez realizada la gestión de citas, se podría buscar e implementar una pasarela de pago para gestionar los cobros de esas citas, teniendo en cuenta

que el pago se podría hacer en diferentes monedas, según el país.

- **Implementación de un perfil más avanzado del paciente**: Que tanto el médico como el paciente puedan editar y en el que se puedan añadir datos al historial del paciente como alergias, antecedente familiares, un historial de consultas, etc.
- **Recetas médicas**: Algún tipo de gestión de recetas médicas, y por tanto de medicamentos, a través del cual un médico pueda expedir una receta, y que queden almacenadas en el perfil del paciente para poder volver a ser reutilizadas o consultadas.

Como se puede ver, esta aplicación tiene unas capacidades de crecimiento infinitas, pudiendo llegar a ser una aplicación de uso sencillo y con posibles funcionalidades verdaderamente útiles.

6 Glosario

- **Accesibilidad**: diseño Web que va a permitir que personas con algún tipo de discapacidad puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos.

- **Ágil o metodología ágil**: Metodología de trabajo que permite adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

- **AngularJS** [2]: framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

- **Backend**: es la parte que procesa la entrada desde el *frontend*, podría ser la capa de acceso a datos.

- **Base de datos:** conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

- **Bootstrap [1]:** Biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web, sólo de *frontend*, que contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales

- **Controlador** (**Controller**): Este componente es el que responde a la interacción (eventos) que hace el usuario en la interfaz y realiza las peticiones al modelo para pasar estos a la vista.

- **Diseño responsive**: es una técnica de desarrollo web cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas.

- **DTO**: Clase java que representa a los objetos que devuelven los Rest Services.

- **Entidad**: Clase java que representa a los objetos que se van a persistir en la base de datos mediante JPA.

- **Framework**: Esquema, patrón o estructura que nos ofrece un entorno genérico o funcionalidades para escribir código en un lenguaje concreto.

- **Frontend**: En diseño de software el frontend es la parte del software que interactúa con los usuarios.

- **Google Fit**: aplicación que mide el ejercicio diario de una persona, y que utiliza diferentes parámetros para que puedas saber si estás haciendo el ejercicio que debiera.

- **Interface o interfaz en Java:** colección de métodos abstractos y propiedades constantes. En las interfaces se especifica qué se debe hacer, pero no su implementación. Serán las clases que implementen estas interfaces las que describen la lógica del comportamiento de los métodos.

- **JPA**: Estándar de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos.

- **Kanban:** Un sistema de gestión de proceso visual que le indica qué producir, cuándo producirlo, y cuánto producir. Se encuentra dentro de las metodologías ágiles y fue creado por Toyota.

- **Mapper**: Clase java que ayuda a transformar las entidades en DTOs y, al contrario.

- **MySQL** [9]: sistema de gestión de bases de datos relacional de código abierto, considerado como uno de los sistemas de bases de datos más populares del mundo para entornos de desarrollo web.

- **Modelo Vista Controlador (MVC)**: patrón de arquitectura de software, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- **Open Authorization (OAuth)**: estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas.

- **Proxy**: Clase Java que hace de intermediarias entre los servicios y los controladores.

- **Repository o repositorio:** clase que se encarga de gestionar todas las operaciones de persistencia contra una tabla de la base de datos.

- **Servicios (MVC)**: Clases java que realizan la lógica de negocio.

- **Servicio REST, REST service**: Es un tipo de **Servicios** Web (En este caso sería nuestro controlador), que se adhieren a una serie de restricciones arquitectónicas englobadas bajo las siglas de **REST**, y que utilizan estándares Web tales como URIs, HTTP, XML, y JSON.

7 Bibliografía

[1] Otto, M. J. T. (s. f.). *Bootstrap*. Bootstrap. Recuperado diciembre de 2020, de https://getbootstrap.com/

[2] AngularJS — Superheroic JavaScript MVW Framework. (s. f.). AngularJS. Recuperado diciembre de 2020, de https://angularjs.org/

[3] Spring | Projects. (s. f.). Spring. Recuperado diciembre de 2020, de https://spring.io/projects

[4] Trello. (s. f.). Trello. Recuperado enero de 2021, de https://trello.com/

[5] Eclipse Foundation. (s. f.). Eclipse Downloads | The Eclipse Foundation. Eclipse. Recuperado octubre de 2020, de https://www.eclipse.org/downloads/

[6] Microsoft. (2016, 14 abril). Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/

[7] ObjectAid UML Explorer. (s. f.). ObjectAid. Recuperado diciembre de 2020, de https://www.objectaid.com/home

[8] Generate Class Diagram Using Eclipse. (2016, 28 abril). Javatips.net. https://www.javatips.net/blog/generate-class-diagram-using-eclipse#:%7E:text=Generate%20Class%20Diagram%20Using%20Eclipse%20Plugin,Now%20you%20can%20see%20test.

[9] MySQL. (s. f.). MySQL. Recuperado octubre de 2020, de https://www.mysql.com/

[10] Gandy, D. (s. f.). Font Awesome, the iconic font and CSS toolkit. Font Awesome. Recuperado diciembre de 2020, de https://fontawesome.com/v4.7.0/

[11] Chart.js | Open source HTML5 Charts for your website. (s. f.). Chartjs. Recuperado diciembre de 2020, de https://www.chartjs.org/