UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación Interacción Medicamentos

Presentado por Iñigo Sanz Delgado
en Universidad de Burgos — 11 de junio de 2024
Tutor: José Manuel Aroca Fernández

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Iñigo Sanz Delgado, con DNI 73109122B, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 11 de junio de 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

El desarrollo económico de nuestro tiempo lleva a todos los sectores de la economía a una necesidad de digitalizarse para no perder el ritmo en el que crecen estas tecnologías. Los procesos médicos o la gestión sanitaria es crucial en la sociedad, por lo que soluciones como una aplicación que acompañe a los especialistas sanitarios a realizar su trabajo de manera más efectiva y reduciendo los tiempos en la dedicación de estos profesionales en su trabajo, crean oportunidades excelentes para el desarrollo de aplicaciones como esta.

Profesionales y pacientes demandan que este proceso de digitalización sea lo más rápido y consistente posible. Se pueden crear soluciones como Activus que sean capaces de integrar una amplia variedad de funcionalidades que ayuden a los especialistas y a los pacientes en estos fines.

Descriptores

Aplicación web, Java, Spring Boot, JPA, MySQL, Thymeleaf, API, Python, Seguridad, Frontend, Backend, Medicina, Medicamentos, Principios activos, Sanidad, Farmacia

Abstract

The economic development of our time drives all sectors of the economy to the necessity of digitizing to keep pace with technological advancements. Medical processes and healthcare management are crucial in society, so solutions such as an application that assists healthcare specialists in performing their work more effectively and reducing the time these professionals spend on their tasks create excellent opportunities for the development of applications like this one.

Professionals and patients demand that this digitization process be as quick and consistent as possible. Solutions like Activus can be created to integrate a wide variety of functionalities that help specialists and patients achieve these goals.

Keywords

Web application, Java, Spring Boot, JPA, MySQL, Thymeleaf, API, Python, Security, Frontend, Backend, Medicine, Medications, Active principles, Healthcare, Pharmacy

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Con Activus se desarrolla una aplicación para gestionar y facilitar la administración de medicamentos y los respectivos tratamientos en el ámbito sanitario. Se intenta buscar la mejora de los procesos de gestión médica proporcionando para ello una herramienta digital que ayude a los especialistas sanitarios a realizar su trabajo de manera más rápida y eficiente.

1.1. Funcionalidades

1. **Gestión de Medicamentos:** Permitir a los usuarios, es decir a los médicos, gestionar medicamentos. Obtener la información necesaria sobre los medicamentos.
2. **Gestión de Pacientes:** Facilitar el seguimiento clínico de un paciente.
3. **Interacciones Medicamentos:** Proporciona una herramienta para poder obtener información sobre las diferentes interacciones de los medicamentos, basado en los principios activos de estos.
4. **Interfaz Visual:** Tener una interfaz de usuarios actualizada, sencilla e intentando que la usabilidad sea acorde a lo que los usuarios necesiten.
5. **Interfaz Multilingüe:** Soporte para múltiples idiomas.
6. **Seguridad:** Implementa seguridad dentro de la aplicación, como permisos para diferentes usuarios, roles y autorizaciones.

Para implementar las funcionalidades se han utilizado tecnologías como Java, Spring Boot, JPA, MySQL, Thymeleaf y API desarrollada en Python.

La aplicación se despliega en un entorno de producción local.

1.2. Estructura de la memoria

1. **Introducción:** Descripción del proyecto y planificación de la documentación.
2. **Objetivos del Proyecto:** Objetivos generales, personales y técnicos que se esperan cumplir realizando el proyecto.
3. **Conceptos Teóricos:** Explicación sobre los conceptos más importantes del proyecto.
4. **Técnicas y Herramientas:** Listado y descripción de la técnicas y herramientas utilizadas durante el proyecto. Incluye bibliotecas, servicios y tecnologías, proporcionando una explicación sobre su uso y la justificación.
5. **Aspectos Relevantes:** Puntos más importantes e interesantes que han surgido conforme se iba desarrollando el proyecto.
6. **Trabajos Relacionados:** Trabajos relacionados con Activus.
7. **Conclusiones y Líneas de Trabajo Futuras:** Conclusiones finales que se han reunido después de realizar el proyecto. Mejoras y necesidades que la aplicación tiene que cumplir en líneas futuras.

1.3. Estrucutura de los Anexos

1.4. Materiales adicionales

- **Aplicación Web:** Implementación tanto del Frontend como del Backend.
- **API:** Herramientas y Backend incorporar la lógica y funcionalidades requeridas.
- **Base de Datos:** Contiene los datos utilizados para la gestión de usuarios y medicamentos.

2. Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por el desarrollo del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

2.1. Objetivos Generales

1. Diseñar e implementar un servicio web para la gestión de usuarios y aplicaciones. El servicio debe permitir registrar, autenticar y autorizar usuarios, así como gestionar roles y permisos.

2. Interconectar una arquitectura que soporte múltiples servicios y tecnologías. La arquitectura incluye una base de datos MySQL, un backend programado en Java utilizando JPA y una API en Python para las funcionalidades adicionales.

3. Facilitar la administración de medicamentos y pacientes a través de un sistema web. Proporcionar una interfaz que permita comparar medicamentos, gestionar pacientes y sus tratamientos, consultar medicamentos y gestionar datos de los pacientes.

2.2. Objetivo Técnicos

1. Implementar una interfaz web utilizando Thymeleaf y Bootstrap. Integrar Thymeleaf para el renderizado dinámico de páginas y Bootstrap para un diseño que sea responsive y moderno.

2. Implementar un sistema de autenticación y autorización robusto con Spring Security. Configurar los roles y permisos de acceso para los diferentes usuarios (administradores, usuarios, ...).
3. Utilizar técnicas avanzadas de diseño web con HTML5 y CSS3. Crear componentes web utilizando HTML y CSS para mejorar la experiencia de los usuarios. Ayudarse de JavaScript para implementar funcionalidades dinámicas.
4. Desarrollar y gestionar una base de datos MySQL utilizando JPA. Crear y mantener la estructura de la BBDD, implementar operaciones CRUD y realizar consultas utilizando JPA.
5. Integrar una API en Python con el backend en Java para funcionalidades adicionales. Desarrollar una API en Python para que realice ciertas tareas y conectarla al backend Java para que se comuniquen.
6. Emplear frameworks y bibliotecas que nos faciliten el desarrollo del código, como puede ser loombok.
7. Documentar el código de manera eficiente y mantener el código limpio, entendible y legible.

2.3. Objetivos Personales

1. Aprender sobre el diseño de una API y como se integra con aplicaciones web. Desarrollar y consumir APIs RESTful en el proyecto.
2. Conocer el ciclo de desarrollo que conlleva un proyecto web. Participar en todas las fases del desarrollo, desde la planificación y diseño hasta la implementación y corrección de errores.
3. Ampliar los conocimientos en Java y JPA para la gestión y comunicación con una base de datos.
4. Aprender sobre la seguridad web y la gestión y autenticación de usuarios. Implementar Spring Security para manejar la seguridad de la aplicación, incluyendo cifrado de contraseñas, validaciones, etc.
5. Desarrollar técnicas y métodos para la corrección de errores durante las pruebas de la aplicación.
6. Mejorar las habilidades de lógica en programación y entender las dependencias del código.

3. Conceptos teóricos

Algunos conceptos teóricos de LaTeX

Para la comprensión y el correcto desarrollo del proyecto, se deben tener claros diversos conceptos teóricos relacionados con las tecnologías utilizadas. A continuación se irán explicando estos conceptos desglosados en diferentes categorías:

3.1. Desarrollo Web

MVC (Modelo-Vista-Controlador)

MVC es una arquitectura software que divide una aplicación en tres partes diferenciadas, el Modelo, la Vista y el Controlador. Comúnmente se utiliza para implementar interfaces de usuario, datos y lógica de control. Es decir que separa la lógica de negocio de su visualización. Al hacer esta separación hace que la realización del proyecto y su mantenimiento sea más sencillo.

- **Modelo:** Aquí está la sección de la aplicación encargada del manejo de datos y la lógica de negocio. En este lugar se establecen las normas y comportamientos de los datos en la aplicación. Un ejemplo de esto es que, en nuestro proyecto, el modelo incorpora las entidades de usuario y roles, así como otras entidades vinculadas a la base de datos.
- **Vista:** Es a través de la vista que los datos son mostrados al usuario. El usuario interactúa y visualiza a través de la interfaz gráfica. En el proyecto, se hace uso de Thymeleaf para realizar la representación de las vistas en el servidor.

- **Controlador:** El controlador cumple la función de ser el intermediario entre el modelo y la vista. Encargarse de administrar la entrada del usuario, ejecutar las operaciones lógicas correspondientes y determinar qué vista es apropiada para mostrar los resultados. En el proyecto, los controladores son creados para manejar las solicitudes de registro, autenticación y navegación entre distintas pantallas principalmente.

Diseño Responsive

El diseño responsive permite que las páginas se ajusten al tamaño de la pantalla del dispositivo donde se están visualizando. Esto principalmente es posible mediante el uso de queries @media de CSS o grids. Se implementa en el proyecto gracias a Bootstrap.

3.2. Desarrollo Backend

El término "backend"se refiere al desarrollo de la lógica del lado del servidor, la cual controla la aplicación web.

Java

"Java es un lenguaje de programación utilizado para crear software compatible con una gran diversidad de sistemas operativos. Este lenguaje tiene la particularidad de ser compilado e interpretado al mismo tiempo; esto significa que es un lenguaje simplificado que convierte automáticamente el código en instrucciones de máquina."

Página oficial: https://www.java.com/

JPA (Java Persistence API)

JPA es una especificación de Java que proporciona una interfaz para la persistencia de objetos en bases de datos relacionales. Permite mapear las entidades que hay en Java a tablas de la BBDD, habilitando el uso y manejo de operaciones CRUD. Que JPA permita manipular los datos de la BBDD como objetos hace que el código sea más fácil de mantener.

Spring Framework

Es un framework que facilita el trabajo centrado en simplificar y flexibilizar la creación de objetos en un programa. Nos permite hacer inyecciones

de dependencias, gestionar transacciones, realizar programación orientada a aspectos (POA), entre muchas funcionalidades más.

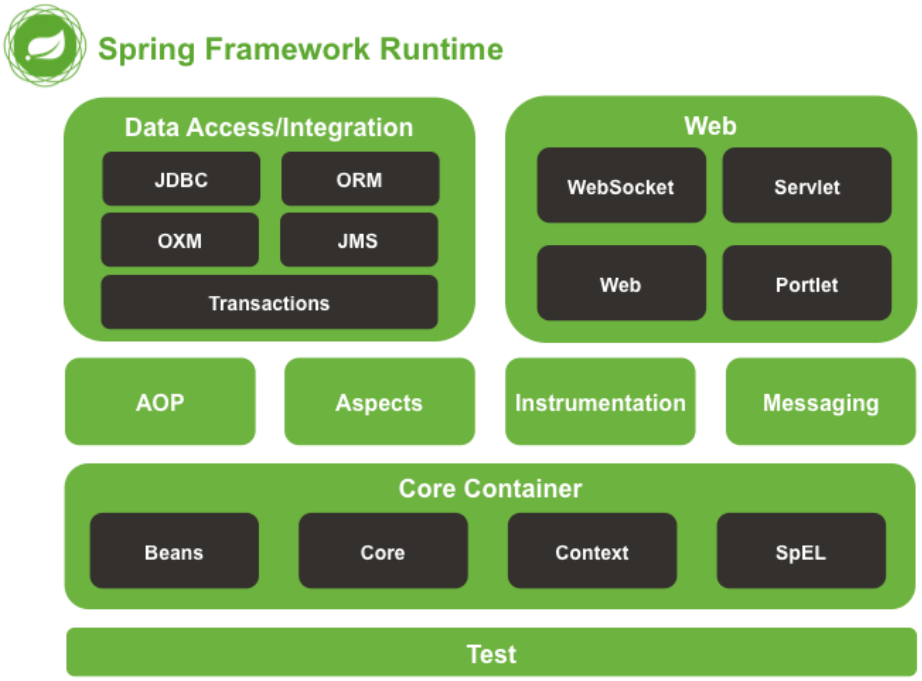

Figura 3.1: Spring Framework Runtime - https://docs.spring.io/

En este proyecto los módulos utilizados son:

- **Spring Boot:** Es un framework desarrollado para el lenguaje de programación Java. Se crea con el propósito de simplificar los pasos de buscar los JARs necesarios y el paso de desplegar la app en el servidor, pudiendo dedicar más tiempo al desarrollo de la aplicación como tal.
- **Spring Data JPA:** Es un módulo que facilita la comunicación e interacción con la base de datos. Evita que tengamos que tratar directamente con la BBDD, es decir, abrir y cerrar las conexiones, la construcción de sentencias entre muchas otras.

Spring Framework tiene un asistente de Boot muy eficaz, llamado Spring Initializer. Nos proporciona los pasos necesarios, elegir el paquete al que pertenecerán las clases, elegir el nombre del proyecto y seleccionar las dependencias que vamos a necesitar.

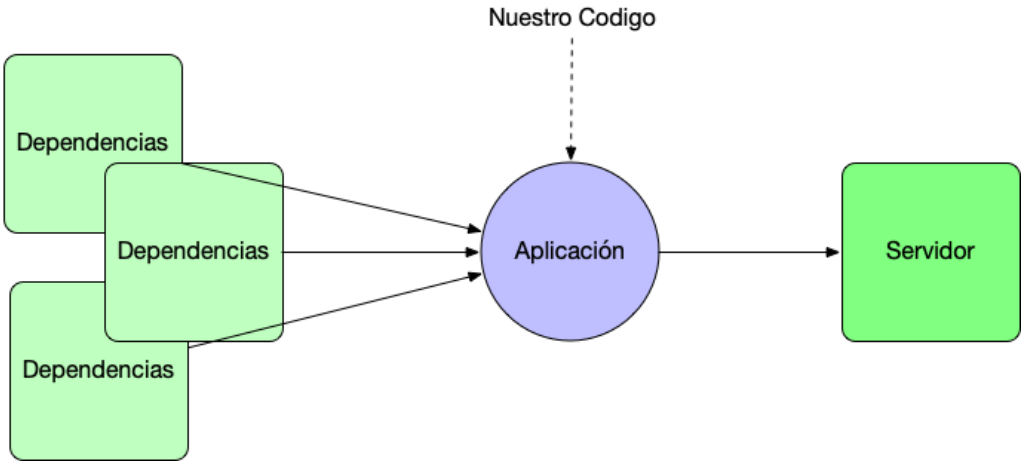

Figura 3.2: Spring Boot - https://www.arquitecturajava.com/

3.3. Seguridad en Aplicaciones Web

La seguridad de una aplicación web es un punto crítico, ya que posiblemente es uno de los puntos que más ataques puede recibir.

Spring Security

Spring Security, otra framework contenido en Spring Framework, proporciona autenticación de usuarios, autorización y permisos, entre otras funcionalidades para frenar las amenazas que puedan entrar. En este proyecto principalmente se utiliza para manejar la autenticación de usuarios y las autorizaciones dentro de la aplicación.

BCrypt

BCrypt es un algoritmo de hashing de contraseñas basado en cifrado Blowfish [23], con derivación de claves para estas contraseñas. Se supone que este método de encriptado de contraseñas es lento, por lo que dificulta mucho el uso de fuerza bruta para adquirir la contraseña. [19].

3.4. Bases de Datos

MySQL

MySQL es un sistema de base de datos relacional (RDBMS - Relational Database Management System) que emplea un modelo cliente-servidor. Es de código abierto y proporciona un buen rendimiento, fiabilidad y sobre todo facilidad de uso.

ORM (Object-Relational-Mapping

La especificación JPA implementa ORM en Java. Mapear un objeto-relacional es convertir los datos entre sistemas que son incompatibles por lenguaje de programación, en programación orientada a objetos. La especificación ORM se ayuda de Hibernate.

Página oficial Hibernate: https://hibernate.org/

3.5. APIs y Servicios Web

API (Application Programming Interface)

Una API viene siendo un conjunto de definiciones y protocolos que se utilizan para crear un software, el cúal luego se integra en las aplicaciones. Así permitiéndonos la comunicación entre diferentes aplicaciones a través de un conjunto de reglas. Hoy en día son fundamentales para que las aplicaciones amplíen sus funcionalidades utilizando otros servicios.

Figura 3.3: Funcionamiento de una API - https://outvio.com/

REST (Representational State Transfer)

REST es una arquitectura que se utiliza para diseñar servicios web. Utiliza HTTP para realizar operaciones CRUD (Create, Read, Update, Delete). Sus características son que es simple y escalable. En el caso de esta aplicación, permite que los clientes realicen solicitudes HTPP para obtener datos o realizar acciones que no están desarrolladas en Java.

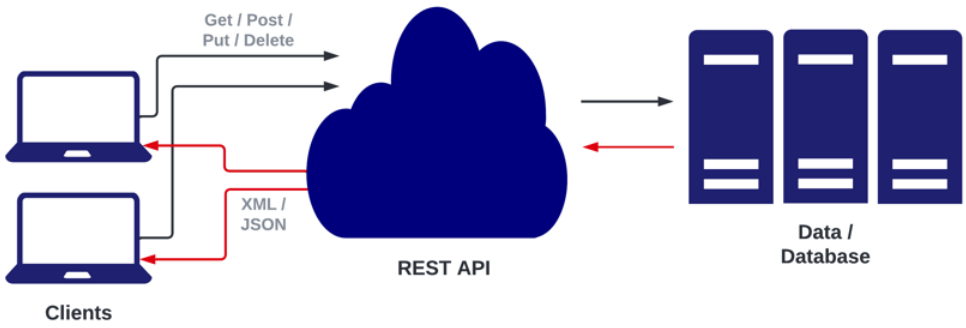

Figura 3.4: API Rest - https://es.player.support.brightcove.com/

3.6. Frontend

Thymeleaf

Thymeleaf es un motor de plantillas, muy eficiente, para Java que se complementa muy bien con Spring. Permite crear vistas dinámicas y renderizar HTML de manera efectiva. El uso de Thymeleaf en este proyecto hace que las múltiples vistas de la aplicación que hay se gestionen de manera más sencilla y que para el usuario sean atractivas y sencillas.

Bootstrap

Bootstrap es una biblioteca de herramientas de código abierto optimizadas para el diseño de sitios y aplicaciones web. Esta plataforma se basa en lenguaje HTML y CSS, e incluye una amplia gama de elementos de diseño, como formularios, botones y menús que se adaptan a diferentes formatos de navegación [12].

4. Técnicas y Herramientas

Esta parte de la memoria tiene como objetivo presentar la metodología técnica y las herramientas para el desarrollo que se han utilizado en este proyecto. Se han estudiado diferentes alternativas de metodologías, herramientas y librerías.

4.1. Tecnologías

En esta sección se van a mencionar las tecnologías, bibliotecas y frameworks que se han utilizado tanto para construir el backend como el frontend.

Backend

Para construir el backend de la aplicación se han combinado diferentes tecnologías, frameworks y bibliotecas. Asegurando de esta manera una arquitectura robusta y escalable.

Java - Robustez y Escalabilidad

La gran mayoría de los apartados del backend, por no decir todos, se han desarrollado utilizando el lenguaje de programación Java. El motivo principal es el uso de una plantilla para la realización del proyecto. Esta plantilla contenía muchas partes adaptables para el proyecto y estaba desarrollada en Java.

A pesar de esto, cabe destacar que Java es un lenguaje de programación conocido por su robustez y la facilidad con la que se puede escalar. La

arquitectura orientada a objetos y la buena manera con la gestiona el uso de la memoria lo hacen una opción eficiente para la creación de una aplicación web.

Haciendo hincapié en la escalabilidad, las funcionalidades de la aplicación que se han desarrollado componen un pequeño porcentaje de lo que va a contener en un futuro, por eso la escalabilidad.

Java - Herramientas y Frameworks

Java, a pesar de ser un lenguaje .ªntiguo", cuenta con un extenso repertorio de frameworks que facilitan tanto el desarrollo, como la integración. El matenimiento, si no se conoce bien el funcionamiento del lenguaje puede ser tedioso, pero no es ni mucho menos complicado.

Se han utilizado diversas tecnologías y frameworks, los cuales son:

Loombok

Loombok es una de las librerías que ofrece Java, con las siguientes ventajas en su uso [17]:

Reducción de código Boilerplate

Es la principal característica de Loombok, reducir el código repetitivo, conocido como Boilerplate. Java, al ser un lenguaje orientado a objetos y fuertemente tipado, es común que sea necesario escribir mucho código para tareas que son entre comillas comunes, como pueden ser los getter y setter o constructores. Lo que esta librería proporciona es la funcionalidad de generar automáticamente este código en tiempo de compilación, reduciendo enormemente la probabilidad de errores y mejorando la legibilidad del código.

Facilidad de Uso

Se integra de manera muy sencilla en el proyecto. Simplemente se añade la dependencia de Loombok al archivo pom.xml y se podría comenzar a utilizar las anotaciones de la librería de inmediato.

Prevención de Errores Comunes

Las anotaciones como @ToString de Loombok hacen que problemas que habiendo implementado manualmente el método no sucedan, ya que trata de manera muy concisa estas operaciones.

En comparaciones de igualdad entre objetos, por ejemplo, también asegura que no haya trampas y se hagan de manera efectiva.

Soporte para Patrones de Diseño

Soporta patrones de diseño, como pueden ser el patrón Builder con @Builder o la creación de Singletons con @Singleton. Hace que el desarrollo sea más simple y que los patrones se implementen de manera correcta.

Más sobre patrones de diseño: https://github.com/

Flexibilidad y Personalización

Loombok puede parecer simple, pero ofrece una gran flexibilidad a la hora de personalizar el código que este mismo genera. Por ejemplo, se pueden especificar nombres de métodos personalizados para los getters y los setters. También excluir/incluir campos en los métodos como toString() o equals().

Todo pueden parecer ventajas, pero tenemos que considerar que el ocultar y reducir el código no tiene por que ser siempre bueno. Alguien que, por ejemplo, vea por primera vez el código de la aplicación puede que le resulte más difícil lograr entender el contenido que hay por debajo al ocultar estos Boilerplates.

Un reto importante al utilizar Loombok, es que si obtenemos un error en alguna parte del código, es más difícil averiguar exactamente donde está y se requiere más experiencia para para solucionarlo que en plataformas más sencillas como podría ser node.js o react.

Más sobre Loombok: https://www.baeldung.com/

JPA

Al haber escogido Java para mapear los objetos, la mejor opción y una de las más utilizadas es implementar JPA para la persistencia de datos. Teniendo en cuenta que la aplicación web esta basada en un Modelo-Vista-Controlador, la cual integra una base de datos MySQL y un API Rest, utilizar la persistencia de datos es óptimo.

La persistencia de datos viene siendo un método para que una aplicación pueda recuperar información desde un sistema que es de almacenamiento no volátil, y hacer que esta información persista. JPA o Java Persistence API, proporciona los mecanismos necesarios para manejar la persistencia de los datos, objetos u otras funciones [8].

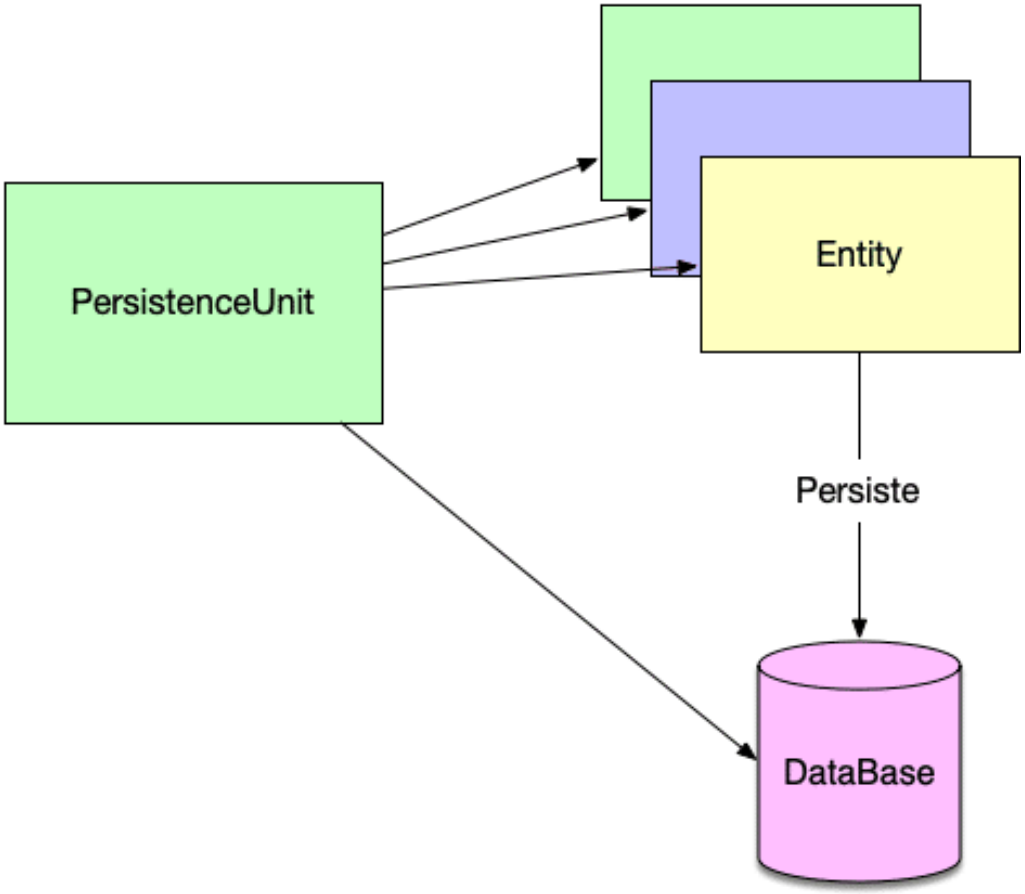

Figura 4.1: Unidad de persistencia - https://www.arquitecturajava.com/

Robustez y Escalabilidad

La naturaleza de Java para la POO hace que se reutilicen los diferentes componentes, algo prácticamente esencial en un MVC. La naturaleza de esta aplicación es que vaya creciendo e implementando nuevas funcionalidades algo que JPA va a proporcionar.

Persistencia de Datos

Volviendo con la persistencia, esta proporciona una capa de abstracción sobre bases de datos relacionales, lo que permite interactuar con la BBDD de una manera sencilla y eficaz, ofreciendo manejo de objetos y sin necesidad de realizar consultas SQL demasiado complicadas.

Integración en el Fronted y Backend

Java y JPA tienen un fácil integración con tecnologías de Frontend y Backend, el uso de Spring Boot, del cual se hablará más adelante, harán que estos pasos sean más sencillos.

Entidades y Transacciones

JPA simplifica en gran medida el manejo de entidades y transacciones, permitiendo un mapeo directo de las tablas que tendremos en la base de datos, logrando consistencia y un sencillo manejo de los datos.

- Permite mapear clases Java a tablas de la BBDD, haciendo sencilla la interacción con esta misma.

- Dispone de soporte para relaciones uno a uno, uno a muchos, muchos a uno y muchos a muchos.

- Hace uso de anotaciones para definir las entidades de las diferentes clases que componen la aplicación, de la misma manera que lo hace para las relaciones, manteniendo el código limpio.

Para la gestión de transacciones hay dos opciones, las automáticas y las declarativas. En las transacciones automáticas JPA actúa de manera asegurando la integridad de los datos y proporcionando una fácil recuperación si suceden fallos. Para las transacciones declarativas, se utiliza la anotación @Transactional, simplificando su manejo [21].

Operaciones de Base de Datos

JPQL (Java Persistence Query Language), la cual nos ofrece JPA, es un lenguaje de consultas orientado a objetos. Este en vez de trabajar con tablas como se haría con SQL, trabaja con las entidades.

Otro punto importante es la sencillez que tiene para hacer operaciones CRUD (Create, Read, Update, Delete) en la base de datos.

Se añade al conjunto que JPA gestiona la Caché con el EntityManager, mejorando el rendimiento de esta y evitando accesos simultáneos a la BBDD en una misma transacción. JPA es una especificación, por lo que según el proveedor que se escoja habrá algunos pequeños cambios, pero hay más opciones para elegir la que mejor se adapte al trabajo.

Spring Boot

Spring Boot esta basado en el Framework Spring y su función principal es ayudar y simplificar el desarrollo de aplicaciones en Java. Evita que se profundice en Spring y su complejidad realizando configuraciones automáticas o pre-armadas para no tener que hacerse de forma manual [6].

Sus principales funciones son:

1. **Inicio Rápido**, Al predefinir la estructura del proyecto y sus dependencias, permite que se empiece a desarrollar la aplicación sin tener que realizar configuraciones complejas para ello.

2. **Configuración Automática**, Según las dependencias que estén en el classpath configura la base de datos, los servidores web, entre muchos otros aspectos.

3. **Servidores Web Embebidos**, Incluye servidores web que están embebidos, como lo son Tomcat o Jetty. Esto permite desplegar la aplicación sin necesitar llevarla a un servidor externo.

4. **Spring Initializer** [20], Es un asistente de Boot muy completo. Lo que hace es generar un proyecto Spring Boot con las dependencias y configuración necesaria, al menos la configuración básica. Su uso se divide en 3 pasos:
   - Se elige el paquete o estructura a la se quiere que pertenezcan las clases.
   - Se le da un nombre al proyecto.
   - Se seleccionan las dependencias que se quieren integrar.

5. **Configuraciones**, Tiene soporte para implementar configuraciones externas utilizando el archivo application.properties. Se pueden definir variables de entorno y argumentos en línea de comandos.

6. **Módulos Instalados**, Viene con módulos preinstalados, Spring Data o Spring MVC entre ellos.

Más sobre Spring Boot: https://spring.io/

La estructura de un proyecto hecho con Spring Boot sería la siguiente:

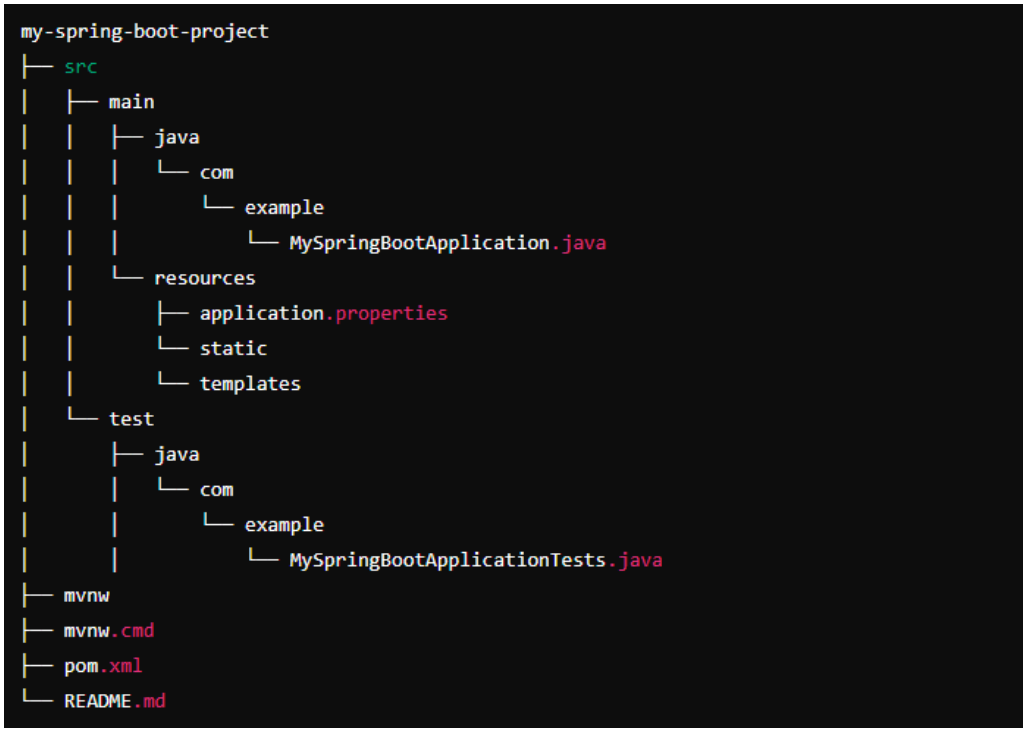

Figura 4.2: Spring Boot Structure

Esta estructura se puede renombrar, pero la configuración básica que Spring Initializer crea es esta.

Las clases que componen una aplicación Spring pueden variar, pero por norma general y en la gran mayoría de casos son las siguientes [24]:

Model

Cada clase Modelo corresponde a una tabla de la BBDD y los campos de la clase equivalen a la columnas de la tabla. Es decir, que representan los modelos de datos de las tablas que utilizamos en la BBDDD.

A los Modelos comunmente se les pone la anotación @Entity, esta anotación es manejada internamente por Hibernate. Cuando Hibernate que la clase contiene la anotación anterior, crea la tabla utilizando el nombre de la clase (se puede especificar un nombre diferente si se quiere).

Más sobre Hibernate: https://www.baeldung.com/

Lo más habitual es especificar las relaciones entre objetos en los Modelos, como podría ser la siguiente:

```
@OneToMany(fetch = FetchType.EAGER, mappedBy = "rol")
private Set<Users> usuarios;
```

Figura 4.3: Relación uno a muchos - Roles y Usuarios

DTO

Los DTOs o Data Transfer Object son clases que tienen una finalidad, la cual es crear un POJO (Plain Old Java Object) con los atributos que se requieren para enviarlos al servidor y luego recuperarlos en una instancia. De esta manera se envían y se recuperan únicamente los que se necesitan.

POJO: https://www.arquitecturajava.com/

Controllers

Los Controladores van a manejar las peticiones que son pasadas por el proveedor de peticiones, y este, retornará la vista correspondiente. Es decir que los diferentes métodos o clases que se definan en el Controlador, devolverán la vista al fichero .html que la contiene.

En los controladores se pueden realizar diferentes ajustes, como definir autenticaciones en las peticiones o utilizar objetos en ellos. En ejemplo podría ser el siguiente, que tiene una anotación de seguridad para que dichas vistas solo sean visibles y accesibles por el usuario que sea administrador:

```java
@Controller
@Secured("ROLE_ADMIN")
public class AdminController {

    @GetMapping("/gestion/usuarios")
    public String gestionUsuarios() { return "gestion/gestionUsuarios"; }

    @GetMapping("/gestion/configuracion")
    public String gestionConfiguracion() { return "gestion/gestionConfiguracion"; }
}
```

Figura 4.4: Controlador para las funciones del Administrador

Services

Los Servicios contienen la lógica de la aplicación. A pesar de que en los Controladores también se puede aplicar lógica, la opción más habitual es implementarla en los servicios, así se consigue separar la lógica y agrupándola para un mejor manejo.

La base dice que los Controladores no deben interactuar de primera mano con los Repositorios, por lo que estos llaman al Servicio correspondiente para que interactúe con el Repositorio objetivo, realizando las operaciones requeridas y devolviendo los resultados al Controlador.

En resumidas cuentas, el Servicio se encarga de manipular los datos que el Repositorio devuelve. Las operaciones más habituales suelen ser filtrado de datos o validación de estos. En la siguiente imagen se muestra uno de los métodos que puede haber en un Servicio, en concreto este se encarga de guardar un usuario y su contraseña en la base de datos:

```java
1 usage
public UsersDto guardar(UsuarioDtoPsw usuarioDtoPsw) {
    System.out.println("usuarioDto:" + usuarioDtoPsw.getNombreUsuario());
    final Users entidad = getMapper().toEntityPsw(usuarioDtoPsw);
    System.out.println("Entidad:" + entidad.getNombreUsuario());
    Users entidadGuardada = getRepo().save(entidad);
    return getMapper().toDto(entidadGuardada);
}
```

Figura 4.5: Guardar nuevo Usuario - Contraseña

Mapper

Las clases Mapper no son exclusivas de Spring Boot, sino que son comunes en el desarrollo de cualquier aplicación. Estas clases se utilizan para transformar datos entre las diferentes capas de la aplicación, pero donde son realmente eficaces y necesarias es para transformar una entidad a un DTO y viceversa.

Al final una clase Mapper ayuda a mantener la lógica de transformación de datos separada de la lógica de negocio y de la lógica de presentación, ayudando también a manejar y modular los datos.

Un ejemplo de una de las funciones sería el siguiente:

```
9 usages
@Override
public RoleDTO toDto(Roles entidad) {
    RoleDTO dto = new RoleDTO();
    modelMapper.map(entidad, dto);
    return dto;
}
```

Figura 4.6: Transformación de Entidad a DTO

Repository

Los Repositorios son los encargados de acceder a la base de datos para realizar las operaciones CRUD. Estos ayudan a los Servicios a que no tengan que ocuparse de detalles como almacenar los datos proporcionando una capa de abstracción.

Son Interfaces que extienden JpaRepository, la cual ofrece todos los métodos necesarios para realizar las operaciones anteriormente mencionadas.

En el siguiente ejemplo se muestra como el Repositorio de los roles se encarga de hacer una búsqueda del Rol por el nombre:

```
3 usages
Roles findByRolName(String rolName);
```

Figura 4.7: Buscar el Rol por nombre

Resumen Estructura

El resumen de la estructura de una aplicación que utiliza Spring Boot se puede dividir en 6 partes, si se ordenan según el flujo de ejecución quedaría de la siguiente manera:

1. **Controller**, Recibe una solicitud HTTP.
2. **Service**, El Controlador le pasa la solicitud recibida al Servicio para que se aplique la lógica de negocio.
3. **Mapper**, El servicio que recoge la solicitud utiliza los Mapper para transformar los datos.
4. **Repository**, De neuvo el Servicio interactúa con el Repositorio para acceder a la BBDD y realizar las operaciones necesarias.
5. **Entity**, El repositorio maneja las entidades para representar las tablas que hay en la base de datos.
6. **DTO**, Simplemente el DTO transfiere estos datos entre la capa de Dervicio y el Controlador.

Figura 4.8: Flujo de ejecucíon

Fontend

Para desarrollar la parte Frontend de la aplicación se han utilizado diferentes tecnologías, las cuales se describirán en el siguiente orden:

- HTML
- CSS
- JavaScript
- Thymeleaf
- Bootstrap

HTML

HTML significa Hyper Text Markup Language (Lenguaje de Marcado de Hipertexto). Principalmente se utiliza para la creación de páginas web, con aplicaciones para optimizar el SEO (Search Engine Optimization). Las características más importantes de HTML son [10]:

- Utiliza tags HTML para describir y estructurar los elementos de una página.
- Su sintaxis es uniforme y textual, cada sección tiene una etiqueta de inicio y otra etiqueta que indica el fin.

Cabe destacar que HTML no es considera un lenguaje de programación. Esto es porque a diferencia de lenguajes como PHP o Python, en HTML no se pueden crear algoritmos, tareas, condiciones o bucles debido a su falta de estructura de comandos.

Por lo tanto, entra dentro de los lenguajes de marcado. Mientras HTML describe y estructura la web, un lenguaje de programación se adhiere para crear los contenidos dinámicos, implementar algoritmos o y crear tareas con lógica.

```
<!DOCTYPE html>
<html lang="">
<head>
    <title></title>
    <meta charset="">
</head>
<body>
</body>
</html>
```

Figura 4.9: Estructuctura básica de un archivo HTML

Estructura de un código HTML

La gran mayoría de los códigos de este lenguaje de marcado tienen una estructura semejante, siempre constan de los siguientes tres elementos:

DOCTYPE

Declaración de tipo de documento, que indica al navegador de que tipo de documento, sintaxis de código y versión HTML se trata. Con HTML5 se pueden indicar una amplia variedad de datos o referencias, por ejemplo, el lenguaje de la página.

Cabecera - head

LA cabecera transmite al navegador detalles importantes y metainformación sobre el cuerpo del documento, conteniendo elementos como los siguientes:

- Título.
- Descripción.
- Autor.
- Juego de caracteres o codificación.
- Información sobre los estilos del documento.

Cuerpo - body

Esta etiqueta contiene toda la información sobre el diseño visible y sobre los elementos dinámicos con los que cuenta la página. Utilizando etiquetas HTML, el cuerpo describe todo lo que el cliente debe ver en el área de visualización del navegador.

CSS

CSS significa Cascading Style Sheets, y permite tanto crear como cambiar la apariencia o estilos visuales de un documento HTML de forma rápida y sencilla [16].

Es un lenguaje que se utiliza para manejar el diseño, hacer cambios para mejorar la apariencia, usabilidad, ... Es decir, permite personalizar una página web.

Que sea en cascada nos indica que hace posible contar en varias hojas con las propiedades heredadas de otras.

Funciones CSS

Como se ha comentado en el apartado anterior, la función principal de CSS es decirle al sitio que aloja la web los estilos que utilizará la página en cuestión para mostrar los contenidos. De esta manera se puede decir que HTML es la estructura de la página web y CSS es el diseño que define el aspecto de la página.

Al ser independiente, es posible crear diferentes plantillas para implementar las ideas que se quieren integrar, y utilizarlas a conveniencia.

- Hacer cambios en la apariencia de la web sin alterar el contenido.
- Control en el diseño de documentos HTML.
- Realizar modificaciones en elementos similares en forma de cascada.
- Establecer y organizar las preferencias de los estilos que se utilizan.
- Reducir el tiempo a la hora de personalizar las vistas.
- Hacer que una página se adapte a los diferentes tamaños del navegador para que el contenido se ajuste y no haya fallos a la hora de visualizarlos (Responsive).

Algo que no se suele tener en consideración es que también sirve para optimizar el rendimiento de la página, reduciendo los tiempos de espera para la carga del contenido o haciendo este más fluido.

```css
@media (min-width: 992px) {
  #mainNav {
    border-bottom: 1px solid transparent;
    background: transparent;
  }
  #mainNav .navbar-brand {
    color: #fff;
  }
  #mainNav .navbar-brand:focus, #mainNav .navbar-brand:hover {
    color: rgba(255, 255, 255, 0.8);
  }
  #mainNav .navbar-nav > li.nav-item > a.nav-link {
    color: #fff;
  }
  #mainNav .navbar-nav > li.nav-item > a.nav-link:focus, #mainNav .navbar-nav > li.nav-item > a.nav-link:hover {
    color: rgba(255, 255, 255, 0.8);
  }
}
```

Figura 4.10: Utilizar media query para aplicar estilos específicos cuando la anchura de pantalla es menor a 992 píxeles

Componentes de una hoja CSS

La hoja de estilos se compone de dos elementos, el primero sería lo que se conoce como declaración y el segundo, selector. Puede contener más, pero estos dos son los más comunes.

- **Declaración:** Es el elemento que indica lo que debe hacer. Las propiedades de una declaración pueden ir desde el color a utilizar, tipo de fuente para el texto, hasta estilos más complejos como poner sombras.
- **Selector:** El selector indica a que elemento se le aplican las declaraciones. Los selectores pueden ser de diferentes tipos, pero los más habituales son los siguientes:
  - Universal: Elige todos los elementos de la página.
  - De tipo o etiqueta: Aplica los estilos a todos los elementos que tienen el mismo valor de etiqueta.
  - Descendente: Proporciona precisión al selector, ya que permite elegir elementos dentro de elementos.
  - De clase: Se utiliza para buscar secciones basadas en un atributo, se prefija con un punto.
  - De ID: Se emplea para seleccionar componentes a los que se les da un valor ID.

Fuente de CSS: https://www.nextu.com/

JavaScript

JavaScript es un lenguaje de programación de vital importancia en el desarrollo de aplicaciones web. La versatilidad que ofrece y la capacidad para ejecutarse en el lado del cliente (navegador) hacen que sea indispensable para mejorar la experiencia de los usuarios [1, 14].

La necesidad de querer integrar elementos interactivos, animaciones o movimiento de datos, hace que se necesite utilizar lenguajes como JavaScript.

JavaScript es por definición un lenguaje de secuencias de comandos que permite crear contenidos dinámicos, controlar elementos multimedia, animar imágenes y mucho más.

Hay infinidad de opciones a la hora de utilizar este lenguaje, pero las más estratégicas y funcionales son utilizarlo para manejar los datos de la página web y hacer conexiones con las API que se quieran implementar.

Ventajas

- La ventaja principal de JavaScript es que se ejecuta en el lado del cliente, por lo que ahorra una gran cantidad de recursos. También optimiza y acelera la ejecución del programa.
- Una página web tiene que poder ejecutarse o visualizarse en cualquier tipo de navegador (Chrome, Edge, Firefox, ...) y esto se permite gracias a JavaScript, gracias a su compatibilidad con cualquiera de los navegador. Es cierto que según la versión puede haber problemas de incompatibilidad.
- Es una buena forma de dar valor al contenido, integrando elementos dinámicos, efectos, transiciones, etc.
- Permite ejecución de código en respuesta a eventos que vayan ocurriendo en la página web. Por ejemplo, si se usa un evento clic para detectar si un cliente pulsa en un botón, JavaScript ejecutaría el código correspondiente para realizar alguna animación o funcionalidad agregada.
- Gran interacción con APIs.

```
$(document).ready(function() : void  {
    $("#locales").change(function() : void  {
        var selectedOption = $('#locales').val();
        var ref : string  = window.location.pathname;
        if (selectedOption != '') {
            window.location.replace( url: ref + '?lang=' + selectedOption);
        }
    });
});
```

Figura 4.11: Script para el cambio de lenguaje de la página

Desventajas

- Antes se menciona tanto la compatibilidad como la incompatibilidad con los navegadores, según la versión con la que se trabaja es posible que haya diferencias en como un navegador interpreta el código respecto de otro, haciendo necesario modificar ciertas partes del código.
- JavaScript DOM (Document Object Model) puede ser lento al trabajar con HTML.
- Si se obtiene un error en el código, al relacionarse con HTML, la localización y corrección de errores puede llegar a ser problemática.

Fuente JavaScript: https://developer.mozilla.org/

Fuente JavaScript complementaria: https://keepcoding.io/

¿Que es un JavaScript DOM? https://lenguajejs.com/

Thymeleaf

Thymeleaf es un motor de plantillas que se ejecuta en el lado del servidor, desarrollado para aplicaciones web basadas en Java.

La característica más importante de este motor de plantillas es que ofrece Natural Templates. Esto es que se basa en añadir atributos en lugar de tener que crear y utilizar nuevas etiquetas HTML. Permite que las plantillas se rendericen en local y que esa misma plantilla se procese en el motor de plantillas [5].

Gracias a esto las tareas de diseño y programación se pueden llevar de manera conjunta de una manera bastante sencilla.

Uso de Thymeleaf

El uso de atributos en Thymeleaf funciona de la siguiente manera:

- El atributo text permite mostrar un texto dentro de las etiquetas HTML.

```
<title th:text="#{app.login.titulo}"></title>
```

Figura 4.12: Muestra en la etiqueta h1 el título de la vista Login

- El atributo href genera una URL en un enlace.

```
<a class="btn btn-secondary text-uppercase btn-rounded" th:href="@{/users/registro}" th:text="#{home.login.btn2}"></a>
```

Figura 4.13: Genera el enlace para ir a la página de Registro

- El atributo src genera la URL para definir la fuente de recursos como imágenes o scripts.

```
<img th:src="@{/assets/img/LogoBienSinFondo.png}" alt="Logo" style="height: 55px;"/>
```

Figura 4.14: Genera la URL de imagen que se utiliza para mostrar el Logo

- El atributo sec, en este caso sec:authorize, se utiliza ayudándose de Spring Security para mostrar o no mostrar ciertos elementos de la página según las reglas que se hayan definido.

```
<li sec:authorize="hasRole('ADMIN')" class="nav-item dropdown">
```

Figura 4.15: Muestra las opciones dentro de la etiqueta li si el usuario tiene el rol de administrador

- El atributo layout se utiliza para la creación y gestión de plantillas de diseño que tienen estructuras comunes, como pueden ser las cabeceras o los pies de página. Se combina con la extensión Thymeleaf Layout Dialect, la cual es una dependencia que hay que agregar el archivo "pom.xml".

```
<section class="container" layout:fragment="cabecera"></section>
```

Figura 4.16: Sección donde definiremos las cabeceras de las vistas. En este caso se utiliza con un fragmento, que son partes reutilizables de código

Tipos de Atributos - Expresiones

- **Variables:** Se escriben poniendo un dólar y abriendo llaves, son las más utilizadas. Permiten usar objetos del modelo de datos.
- **Selección:** Permiten reducir la longitud de una expresión prefijando un objeto variable, se escriben con un asterísco y abriendo llaves.
- **Mensaje:** A partir de ficheros de tipo properties o de ficheros de texto, permiten cargar mensajes en la aplicación. Se escribe con la almohadilla y abriendo llaves.
- **Enlace:** Proporciona crear URLs que pueden tener parámetros variables, se escribe con una arroba y abriendo llaves.
- **Fragmentos:** Permiten dividir la plantilla en plantillas más pequeñas para ir cargándolas según se vayan necesitando.

```
- Variables: ${}
    <p th:text="${user.name}">Nombre del usuario</p>

- Selección: *{}
    <div th:object="${user}">
        <p th:text="*{name}">Nombre del usuario</p>
        <p th:text="*{email}">Email del usuario</p>
    </div>


- Mensaje: #{}
    <p th:text="#{welcome.message}">Mensaje de bienvenida</p>


- Enlace: @{}
    <a th:href="@{/user/profile(id=${user.id})}">Perfil del usuario</a>


- Fragmentos: ~{}
    <div th:replace="fragments/header :: header"></div>
```

Figura 4.17: Ejemplos de expresiones en Thymeleaf (escrito en documento de texto plano)

Página oficial: https://www.thymeleaf.org/

Bootstrap

Bootstrap es una biblioteca de herramientas de código abierto optimizadas para el diseño de sitios y aplicaciones web. Esta plataforma se basa en lenguaje HTML y CSS, e incluye una amplia gama de elementos de diseño, como formularios, botones y menús que se adaptan a diferentes formatos de navegación [5].

La historia de Bootstrap radica en la necesidad de solucionar los problemas de navegación móvil, por lo que fue creado por GitHub para ello. Es muy común encontrarse código CSS escrito en Bootstrap o con librerías más grandes, por ello se cataloga como un marco de CSS.

Motivos de Uso

Los principales motivos de utilizar Bootstrap a la hora de desarrollar una aplicación web son los principalmente dos:

1. Querer que el sitio web sea Responsive.
2. Ahorro de tiempo en el diseño y aplicación de estilos.

Cabe destacar que si queremos una página web con estilos o personalización única, utilizar Bootstrap no es una buena idea.

La manera más común de cargar Bootstrap es mediante un CDN (Content Delivery Network). También se suelen descargar los archivos en cuestión e incluirlos localmente en el proyecto.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
```

Figura 4.18: Cargar Bootstrap desde un CDN

Página oficial: https://getbootstrap.com/

API

API o Application Programming Interface, es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones independientes a través de un conjunto de reglas [3].

Se puede decir que una API es una especificación formal que establece como un módulo software se comunica con otro, o interactúa con otro para cumplir diferentes funciones. Todo esto dependiendo de las necesidades de las aplicaciones y los permisos que se le dé [3].

Pueden ser privadas para el uso de una empresa, por ejemplo, abiertas solo para colaboradores específicos, o públicas para que cualquier desarrollador interactúe con ellas. Pueden estar tanto alojadas en local para aplicaciones dentro de un mismo ambiente, o en remoto para cuando hay que acceder desde puntos diferentes.

¿Para qué sirve una API?

La función principal de una API es facilitar el trabajo a los desarrolladores y ahorrar tanto tiempo como dinero. Ejemplos:

- Si se está desarrollando una tienda web, ya no es necesario crear desde cero un sistema de pago o un sistema que compruebe el stock disponible de los productos.
- Se puede utilizar una API que provee un servicio de pago, como podría ser Paypal o Bizum, y pedirle al desarrollador que la integre o integrarla si se tienen los permisos necesarios.

De esta manera no hay que rehacer el trabajo que ya esta hecho. Hablando del ejemplo anterior, si existe un sistema de pagos que los usuarios utilizan habitualmente, es más sencillo implementar este en los diferentes proyectos que se aborden en vez de crear un sistema de pago para cada uno.

También son útiles cuando solamente se quiere utilizar las funciones de un determinado servicio, ofreciendo las ventajas que aporta a los usuarios que utilicen la aplicación en cuestión. Ejemplos:

- Cuando se va a comprar una entrada para cualquier tipo de evento, se pone la información de la tarjeta de crédito. La web donde se compra la entrada, normalmente, utiliza una API para enviar esa información a otro programa que verifica si los datos introducidos son los correctos o si puede ser una tarjeta no válida (Bancos).
- Cuando se verifica, el programa remoto le dice a la web que la información es correcta, valida el pago y esta proporciona las entradas.

Fuente para los ejemplos: https://www.xataka.com/

Rest API

Un API REST (Representational State Transfer) viene siendo un conjunto de restricciones que se deben tener en cuenta en la arquitectura para crear aplicaciones web, normalmente respetando HTTP.

Las restricciones de un API RESTful son:

- Cliente-Servidor: hay un servidor que se encarga de controlar los datos y por otro lado hay un cliente que se encarga de manejar las interacciones del usuario. Debido a esto, se dice que hay un acoplamiento débil entre el servidor y el cliente, ya que el cliente no necesita saber la implementación que hay detrás en el servidor y viceversa con como se usan los datos que el devuelve a los usuarios.

- No tiene estado: cada petición que recibe el servidor debería ser independiente de las demás.
- Cacheable: tiene que tener un sistema de almacenamiento en la caché, así se evitan repetidas conexiones para recuperar el mismo recurso que se utiliza.
- Interfaz uniforme: define una interfaz genérica para administrar cada una de las interacciones que se producen entre el cliente y el servidor, lo que quiere decir que cada recurso del servicio REST debe tener una única dirección o "URI" (Uniform Resource Identifier).
- Capas: el servidor puede disponer de varias capas para su implementación, ayudando en su escalabilidad, rendimiento y seguridad.

Métodos de una API Rest

Son las operaciones que utiliza para manipular los recursos en cuestión, las más comunes son:

- **GET:** se para recuperar un recurso.
- **POST:** este método se suele utilizar para crear un recurso, pero también puede utilizarse para enviar datos a un recurso que ya esta creado y luego procesarlo.
- **PUT:** se utiliza para crear o editar un recurso. Es una operación en la que si el recurso esta creado se actualiza y en caso contrario se crea.
- **PATCH:** método que hace actualizaciones parciales, es decir, se puede determinar en la petición que es exactamente lo que se quiere y utilizar únicamente esa parte del recurso.
- **DELETE:** utilizado para eliminar un recurso.

Y otras menos habituales pero utilizadas también son:

- **HEAD:** tiene un funcionamiento similar el GET, pero este no recupera el recurso. Se utiliza habitualmente en la parte de test, por ejemplo, para comprobar si un archivo existe (un documento Excel o CSV de gran tamaño) sin necesidad de descargarlo.

- **OPTIONS:** método que permite ver las opciones que tiene un recurso para hacer las peticiones.

Los métodos se clasifican como seguros o idempotentes. Es decir, los métodos seguros son los que no modifican los recursos (GET, HEAD y OPTIONS), los idempotentes son los que pueden hacer llamadas varias veces obteniendo el mismo resultado (GET, PUT, DELETE, HEAD y OPTIONS).

Que un método sea idempotente es importante, ya que permita que la API sea tolerante a los fallos. Por ejemplo:

Características de una API Rest

Una API Rest bien diseñada podrá realizar una operación PUT para editar un recurso. Si se obtiene un fallo durante la petición, como podría ser un Timeout, no se sabría si el recurso fue creado o si fue modificado.

Como PUT es idempotente, no hay que comprobar el estado de la petición, ya que, si se vuelve a realizar dicha petición, si en la anterior se hubiese creado, en esta no crearía otra.

Fuente: https://www.idento.es/

Se estudió la opción de utilizar Fast API, pero por la popularidad de Rest API y el uso extendido que tiene, se decidió por este último.

Información sobre Fast API: https://opensistemas.com/

Frameworks y Librerías API

En este apartado se van a destacar los frameworks y librerías que se han utilizado en la API:

Flask

Flask es un framework web de código abierto que se creo en 2010. Flask es minimalista, incluye el motor de plantillas Jinja[7] y una biblioteca llamada

tool. Pero tiene una gran ventaja, y es que permite integrar funciones de terceros [9].

Las características principales de este framework son:

1. **Aprendizaje:** Flask es bastante fácil de aprender. Es sencillo y puede utilizarse tanto para proyectos complejos como para proyectos poco exigentes.
2. **Código abierto:** Es de código abierto y gratuito.
3. **Comunidad:** Flask posee una comunidad enorme que ofrece consejos y soluciones.
4. **Flexibilidad:** Se pueden implementar infinidad de bibliotecas para resolver los problemas que se aborden.

SQL Alchemy

SqlAlchemy es una librería usada en Python para interactuar con bases de datos. Proporciona una interfaz para poder realizar una gran cantidad de operaciones con bases de datos, independientemente de si son de la misma naturaleza. Evita que se tenga que prestar atención a la especificaciones de la BBDD.

Página oficial: https://www.sqlalchemy.org/

Pandas

Pandas es una biblioteca diseñada especificamente para el análisis de datos. Es de código abierto, potente y fácil de utilizar.

Permite realizar funciones como cargar y manipular datos. Es realmente efectivo para procesar datos que estén estructurados en tablas, matrices, series, etc.

Es compatible con muchas bibliotecas de Python, por lo que ofrece una gran flexibilidad en su uso.

Pandas trabaja con DataFrames, que son tablas de datos bidimensionales, donde cada columna contiene ciertos valores de variables y cada fila contiene un conjunto de valores de cada columna. Los datos que se guardan en un DataFrame pueden ser tanto caracteres como números [18].

Proporciona la funcionalidad de exportar o importar datos en diferentes formatos, como pueden ser en formato Excel o JSON, por lo que es realmente útil para trabajar con datos tabulares y estadísticos.

Re

La librería re es un módulo que ofrece la posibilidad de tratar expresiones regulares. Es decir, que permite buscar, coincidir y manipular cadenas de texto.

```
def limpiar_cadena(cadena):
    return re.sub(r'[^a-zA-Z0-9\s]', '', cadena).strip().lower()
```

Figura 4.19: Limpiar cadena de carácteres

4.2. Herramientas

Herramientas utilizadas en la elaboración del proyecto.

IntelliJ

IntelliJ IDEA es un entorno de desarrollo integrado para el desarrollo de programas informáticos. Esta desarrollado por JetBrains y disponible en diferentes versiones, en este caso se ha utilizado la versión para estudiantes.

El elegir IntelliJ por encima de por ejemplo, Visual Studio Code, es que para Java en particular esta optimizado a niveles que los otros lo están. Integra finalización de código de línea completa sin necesidad de plugins (se pueden implementar). Implementa un asistente basado en IA o inyecciones de lenguaje entre muchas otras funcionalidades.

Página oficial: https://www.jetbrains.com/

Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes y entorno de ejecución (como C++, Java, Python, Go, .NET) [15].

VS Code se ha utilizado principalmente para el desarrollo de la API, su compatibilidad y optimización para el lenguaje Python esta por encima de muchos otros IDEs. A la hora de tratar con grandes cantidades de datos funciona realmente bien y consume una cantidad de recursos moderada.

Características VS Code: https://learn.microsoft.com/

Github Copilot

Github Copilot es un servicio que ayuda a los desarrolladores a autocompletar código a medida que se va programando. Al estar basado en IA, Copilot va adhiriéndose al contexto de archivo que se esta editando y mostrando sugerencias.

Está entrenado con los repositorios públicos, sobre todo los publicados en Github. Este modelo de IA Generativa esta desarrolado por Github, Microsoft y OpenAI, y basado en el modelo GPT 3.5 de OpenAI.

Como permite la integración tanto en IntelliJ como en VS Code, se ha utilizado principalmente para la documentación del código, ya que al no tener el contexto completo del proyecto es dado a dar sugerencias incorrectas o a cambiar nombres de funciones y variables que no debe cambiar.

Documentaciñon Oficial: https://docs.github.com/

ChatGPT

ChatGPT es un sistema con el que se puede chatear para que nos ayude con infinidad de temas. Es un LLM (Large Language Model) desarrollado por la empresa OpenAI, muy usado en el ámbito de la inteligencia artificial. Es un modelo que contiene más de 175 millones de parámetros y entrenado con una enorme cantidad de datos [4].

A pesar de que ChatGPT puede cometer errores, para tareas sencillas o donde no hace falta un contexto demasiado extenso, funciona realmente bien.

Por ejemplo, la imagen de fondo que se utiliza en este proyecto ha sido creada con la versión 4o de ChatGPT, actualmente gratuita (25/05/2024), que utiliza DALLE.

Página oficial: https://chatgpt.com/

WAMP

WAMP es una aplicación que proporciona un entorno de desarrollo local, principalmente para aplicaciones web. WAMP hace de servidor virtual en el ordenador de quien lo instala (WampServer).

Significado:

- **W:** Hace referencia al sistema operativo Windows.
- **A:** Referencia al servidor web Apache.
- **M:** Utiliza MySQL como lenguaje para la gestión de bases de datos.
- **P:** Lenguaje PHP.

Hay opciones quizá más extendidas como XAMPP, pero al ser una aplicación principalmente desarrollada para Windows y la facilidad de uso y configuración de este mismo, hacen que sea la opción elegida.

Página oficial: https://wamp.net/

POSTMAN

Postman es una herramienta que permite realizar pruebas con las APIs, realizando consultas a servicios web y aplicaciones para visualizar sus respuestas y comportamiento. Es decir, se puede utilizar la interfaz que proporciona para enviar solicitudes HTTP a una API y gestionar las respuestas que esta devuelve.

Características:

- **Enviar solicitudes HTTP**: Permite enviar solicitudes HTTP a una API especificando diferentes parámetros o cuerpos en la solicitud.
- **Gestionar entornos**: Hace posible configurar los entornos, ya sea de ejecución, desarrollo, pruebas, etc.
- **Generar documentación de la API**: Permite generar documentación de la API basándose en las consultas que se realizan y las respuestas que se reciben.

Página oficial: https://www.postman.com/

5. Aspectos relevantes del desarrollo del proyecto

En este apartado se anotan los aspectos relevantes en el desarrollo del proyecto, incluyendo las dificultades encontradas en la realización de este y las soluciones que se han aplicado.

A pesar de no haber seguido la metodología SCRUM, si que se pueden diferenciar 5 fases del desarrollo.

1. Hacer que la aplicación Java funcione, es decir, que las funciones básicas o necesarias de la parte de Java no tengan errores y que tenga una interfaz web mínima.

2. Empezar con la API. Descargar las librerías necesarias para desplegar una API y comenzar integrando funcionalidades sencillas, aún sin conexión con la aplicación.

3. Adaptar y modificar la interfaz web, crear las vistas necesarias con las funcionalidades que deben ofrecer. Incluye también su respectivo apartado de codificación en la parte Java de la aplicación.

4. Completar todas las funcionalidades que debe realizar la API.

5. Conectar la API con la aplicación. Arreglar errores pendientes e implementar las últimos servicios que debe ofrecer.

5.1. Inicio

Despúes de la selección del proyecto, se contactó con el tutor. La primera toma de contacto sirvió para hacerse una idea sobre el alcance del proyecto, tecnologías que había que utilizar y servicios que debía ofrecer.

Una vez estructurado el proyecto, se parte de una plantilla para comenzar con el desarrollo. A pesar de que dicha plantilla contenía la mayoría de funciones que se iban a necesitar, al principio ocasionó un desconocimiento absoluto en los diferentes apartados que se tenían que abordar.

Principales inconvenientes

- Al utilizar como plantilla un proyecto que contenia muchos y diferentes ficheros con tecnologías y frameworks desconocidos, hizo que el entender como funciona cada parte de la aplicación fuese algo bastante complicado.
- El cometido era adaptar el código, pero no valía con copiarlo y refactorizarlo. En el comienzo aparecieron infinidad de errores los cuales no se pudieron solventar de manera sencilla al no saber ni entender el funcionamiento general de cada una de las partes que componían el proyecto.
- Falta de entendimiento y conocimiento sobre las tecnologías utilizadas. Derivando en grandes errores de conceptos sobre las estructuras y código.

Durante la primera fase de desarrollo o los primeros meses, se fueron completando las diferentes partes de las estructura de la plantilla y al mismo tiempo se iban haciendo reuniones con el tutor para su explicación. Obteniendo de la misma manera información tanto de las referencias nombradas en el documento como de vídeos colgados en Internet.

En un comienzo, el partir de una plantilla ocasionó más problemas que facilidades. Si que es verdad, que conforme si iba desarrollando la aplicación y solventando los errores y fallos que iban apareciendo, el tener la plantilla fue beneficioso. Ya que una vez comprendidas las tecnologías, frameworks o estructuras que se utilizaban, también se entendía de mejor manera la plantilla.

Un error que se debe destacar, es en la refactorización del código. Se intentó cambiar tanto el nombre de las variables como de diferentes fun-

ciones, las cuales eran comunes en desarrollos de aplicaciones de este tipo, ocasionando fallos que podrían haberse evitado.

5.2. Fase 1

Uno de los mayores retos en esta fase fue entender como funciona una aplicación que utiliza Spring. Es decir, como se relacionan los siguientes módulos:

- Modelos.
- DTOs.
- Servicios.
- Mappers.
- Repositorios.
- Controladores.

Entender Spring Boot al principio parecía una tarea realmente difícil, pero el cometer fallos y solucionarlos hace que se entiendan mejor los problemas que surgen y que pueden surgir. Acabando finalmente por entender el funcionamiento general o estructura del proyecto.

Por otro lado, al no haber utilizado el Spring Initializr sino ir creando la estructura conforme se iba desarrollando cada módulo, hizo que las dependencias de Maven fallaran continuamente. El POM Starter o el fichero pom.xml contenía versiones obsoletas de las dependencias o simplemente no incluía las dependencias necesarias. La solución fue ir buscando las dependencias necesarias y añadirlas al fichero.

Servicios

Los servicios fueron un bastante complicados, debido a que no se comprendía que los mapper facilitan la interacción con los repositorios a los servicios.

Es cierto que entender individualmente los Mappers no es complicado, pero los servicios concretos que se enfocan por ejemplo en los usuarios o roles, al utilizar tanto los mappers como el servicio abstracto, hicieron que la curva de aprendizaje se incrementará.

```java
@Service
public class RolesService extends AbstractBusinessService<Roles, Integer, RoleDTO, RolesRepository, RolesServiceMapper> {

    1 usage
    private final UsersRepository usuarioRepository;

    protected RolesService(RolesRepository repository, RolesServiceMapper serviceMapper, UsersRepository usuarioRepository) {
        super(repository, serviceMapper);
        this.usuarioRepository = usuarioRepository;
    }
```

Figura 5.1: Servicio para los Roles

En este caso por ejemplo, a la hora de llamar al servicio de Roles, se llamaba con atributos erroneos y con un Mapper mal definido. Lo que ocasionaba en errores como NullPointerException o que el Repositorio en cuestión no realizase con que tenía que realizar.

Mappers

En los Mappers no hubo tanto problema, quitando que en muchas ocasiones se codificaron los métodos toDto y no los métodos toEntity.

En definitiva, en esta primera fase, la dificultad principal fue el entendimiento de Spring Boot, el poder operar con los métodos de Loombok (poca visibilidad de código) de manera correcta, la nomenclatura cambiada en ciertas partes y la refactorización del código.

5.3. Fase 2

La primera toma de contacto con este apartado fue el documentar un poco la tecnología o tecnologías que se podían utilizar para crear una API. Una vez seleccionada, se decidió por utilizar Rest API.

Se instala Flask en VS Code y se levanta un punto de API que únicamente devolviese que el API estaba en funcionamiento. El primer error fue no configurar el entorno de desarrollo, se necesitaba también el entorno Python Enviroment para levantar la API.

Se hacen pruebas de conexión con la base de datos, las cuales en primera instancia tampoco funcionaron, otra vez por la ausencia de la instalación de paquetes de SqlAlchemy.

Después de solventar los errores que iban surgiendo conforme se iba desarrollando esta primera versión de la API, se llega al punto donde

se consigue hacer una conexión con la BBDD y que esta devuelva datos introducidos en ella manualmente.

Basado en los vídeos de AnderCode: https://www.youtube.com/

5.4. Fase 3

En la fase 3 se empieza con el desarrollo de la interfaz de la aplicación. Posiblemente uno de los apartados del proyecto donde más tiempo se ha invertido.

Se empieza con un desconocimiento absoluto sobre Thymeleaf, HTML y CSS. Por lo que el entender el funcionamiento de las plantillas descargadas de Bootstrap, el funcionamiento del Layout de Thymeleaf y como utilizar los atributos y expresiones llevó bastante tiempo.

Siguiendo con las plantillas de Bootstrap, el implementar Bootstrap funciono correctamente y se utilizaron las vistas de la plantilla para luego elaborar las necesarias en la aplicación. Sin embargo, el estructurado y etiquetado correcto de cada sección llevo más tiempo del esperado. Se hicieron infinidad de modificaciones en las vistas para crear una estructura lineal en la aplicación.

CSS

Para CSS se repite la situación, el archivo styles.css que viene con Bootstrap contenía 11.000 líneas de código, por lo que hacer cualquier modificación como aumentar el tamaño de la fuente de los enlaces de la barra de navegación, requería de una inversión de tiempo exagerada.

Una vez identificados los selectores y las etiquetas del archivo se hacen pequeños cambios en el apartado visual. A pesar de depender de un archivo CSS, el no tener el tiempo suficiente para entender las herencias de los selectores, atributos y etiquetas de CSS, se opta por implementar la etiqueta de estilos en el head del layout. De esta forma se consigue adaptar al gusto requerido el apartado visual de la página.

Las vistas que se crean van heredando del Layout los estilos, y repitiendo la estructura de las plantillas se consigue homogenizar y armonizar los estilos utilizados.

JavaScript

Los scripts utilizados en un principio iban a ser los que utilizase la plantilla Bootstrap. Pero según se iban cambiado los estilos de la página, hubo que hacer cambios en el script e incluso añadir algunos más.

El script de la plantillo añade la funcionalidad de integrar la cabecera con la barra de navegación. Cuando se hace scroll, por ejemplo hacía abajo, la barra de navegación se esconde dejando una vista limpia, y cuando se hace scroll hacía arriba, aparece la barra de navegación.

Al cambiar la imagen de fondo que se utiliza para la cabecera fija, se tuvo que cambiar los estilos a la hora de hacer scroll, para cambiar por ejemplo el color de la fuente de los enlaces del menú, teniendo que implementar otro script.

Luego, para el cambio de idioma de la página, también se tuvo que implementar un script que permitiese esta opción.

```
$(document).ready(function() : void  {
    $("#locales").change(function() : void  {
        var selectedOption = $('#locales').val();
        var ref : string  = window.location.pathname;
        if (selectedOption != '') {
            window.location.replace( url: ref + '?lang=' + selectedOption);
        }
    });
});
```

Figura 5.2: Script cambio de idioma

Respecto a JavaScript, las últimas modificaciones fueron respecto a la barra de navegación y su contenido. Adaptarlo para que funcionase de la misma manera en ventanas grandes como en ventanas enfocadas a dispositivos más pequeños, como dispositivos móviles.

Por último, se intentó dejar el Layout lo más limpio posible, por lo que se movieron los scripts a un archivo js. Para los estilos implementados en el Layout, se barajaron diferentes opciones, que ninguna funcionaron:

1. Pasar al archivo css los estilos y hacer pruebas. Pero los estilos predefinidos en dicho archivo machacaban los últimos cambios.

2. Añadir el atributo !important para prevalecer los cambios sobre los que ya venían en la plantilla, pero no respetaba algunos de los estilos que si eran necesarios, se descartó por falta de tiempo.

Vistas y Controladores

Las vistas se completaron sin mucho problema. Surgieron algunas dudas con los controladores, sobre todo con el controlador que utiliza parámetros para el registro.

Este controlador tiene tanto un método GET como uno POST, se colocaron banderas en los dos métodos para realizar una traza y ver donde fallaba o hasta donde llegaba. El problema estaba en el método POST, el cual derivó a realizar ciertas modificaciones en archivo de seguridad de Spring Boot. Una vez se le dieron permisos, al menos el registro entraba al método POST del controlador, para así registrar los datos del nuevo usuario en la base de datos.

```
@GetMapping("/users/registro")
public String vistaRegistro(Model interfazConPantalla) {
    // Instancia en memoria del dto a informar en la pantalla
    final UsuarioDtoPsw usuarioDtoPsw = new UsuarioDtoPsw();
    // Obtengo la lista de roles
    final List<RoleDTO> roleDTOList = roleService.buscarTodosAlta();
    // Mediante "addAttribute" comparto con la pantalla
    interfazConPantalla.addAttribute( attributeName: "datosUsuario", usuarioDtoPsw);
    interfazConPantalla.addAttribute( attributeName: "listaRoles", roleDTOList);
    return "users/register";
}

@PostMapping("/users/registro")
public String guardarUsuario(@ModelAttribute(name = "datosUsuario") UsuarioDtoPsw usuarioDtoPsw) throws Exception {
    if (ValidarFormatoPassword.ValidarFormato(usuarioDtoPsw.getPassword())) {
        Users usuario = service.getMapper().toEntityPsw(usuarioDtoPsw);
        String encodedPassword = passwordEncoder.encode(usuarioDtoPsw.getPassword());
        usuarioDtoPsw.setPassword(encodedPassword);
        UsersDto usuario1 = this.service.guardar(usuarioDtoPsw);
        return String.format("redirect:/users/%s", usuario1.getId());
    } else {
        return "users/register";
    }
}
```

Figura 5.3: Controlador para el registro de nuevos usuarios

Ahora, surgía otro problema, registraba el usuario con el Rol null. Esto se soluciona en la fase 5.

5.5. Fase 4

Esta fase esta dedicada completamente al API.

Cargar Tabla de Datos

El primer punto de API que se completa es la carga de datos desde el fichero excel que se puede descargar desde la web de Vademecum [22] o desde la web del ministerio de sanidad [2]. Este fichero contiene todos los medicamentos registrados en el país y su información:

- Código del medicamento.
- Nombre comercial.
- Precio de venta al público.
- Laboratorio que lo produce.
- Principios Activos.
- Etc.

Para realizar esta función, primero se optó por cargarla en el inicio de la aplicación. Se codificaron los ficheros necesarios, modelo, servicios, controlar, etc. Funcionó, pero, no carga los datos en el orden en el que se definieron, y esto llevaría a errores al hacer las consultas luego desde la API.

Entonces se decidió cargar los datos del excel mediante un punto de API como se ha mencionado antes.

En una primera instancia se crea un método llamado cargar_datos_desde_excel(), el cual se iba a utilizar también más adelante para cargar más datos. Luego se completa el punto de API que llama a este método para cargar los datos en la BBDD. Utilizando la librería pandas se realiza de manera sencilla con funciones predefinidas para leer archivos (pd.read_excel()).

Los datos se guardan en un DataFrame, luego utilizando df.to_sql() se cargan los datos del DataFrame en la tabla.

De esta manera se consiguen cargar los datos de la tabla y se procede a realizar consultar, por ejemplo para devolver cierta información de un medicamento o devolver los que tuviesen cierto nombre.

Para el siguiente punto de API, el cual era el que realizase las operaciones para sacar la compatibilidad entre diferentes medicamentos basándose en los principios activos de cada uno, se necesitaba encontrar la mejor manera de hacerlo.

Tabla de Compatibilidades

Después de investigar un poco la mejor manera de realizar la comparaciones, se llega a la conclusión de que la mejor opción es cargar una tabla den la base de datos para hacer las consultas. La primera opción era codificar la tabla en python, pero no era la opción óptima. Iba a consumir muchos recursos e iba a costar mucho tiempo replicarla.

La tabla para las compatibilidades se consigue de la siguiente página: https://es.scribd.com/

Se busca en infinidad de páginas tablas de compatibilidades, pero debe ser algo que no es público o que no es fácil hacerse con ellas. Al final se encuentra esta tabla, que indica la compatibilidad de los principios activos administrados en Y [11].

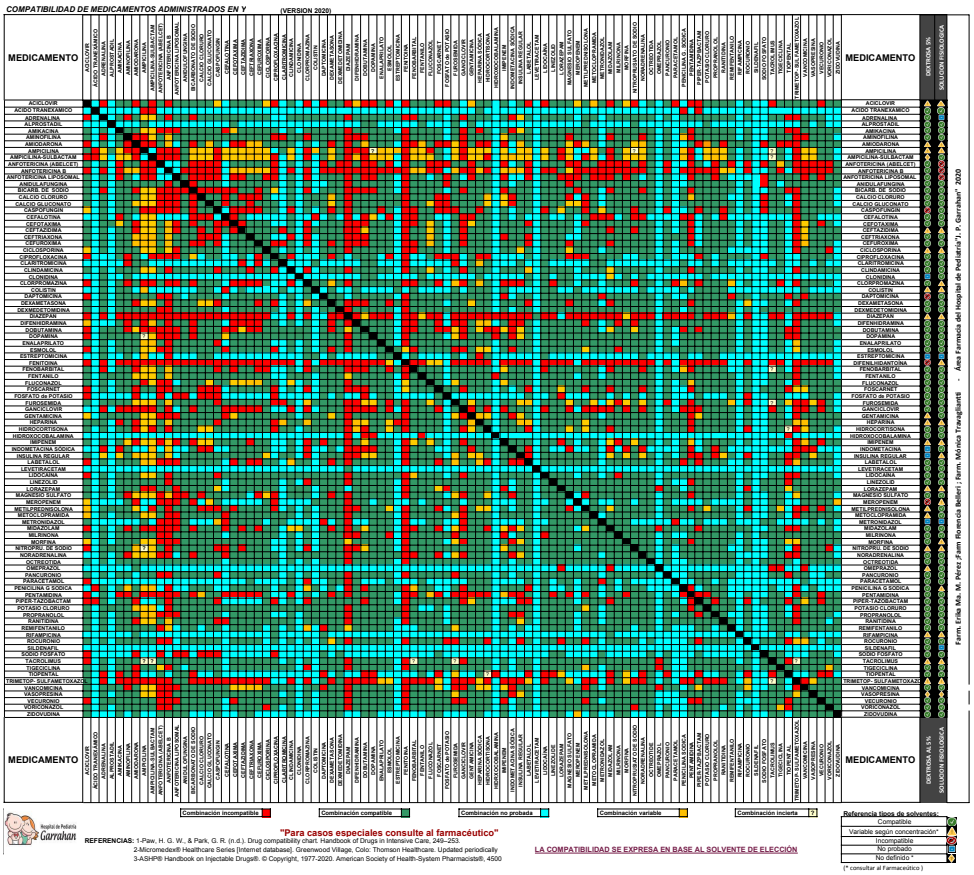

Figura 5.4: Tabla compatibilidades en Y

El siguiente paso es cargar esta tabla en la base de datos, pero, ¿Cómo hacerlo? Contacté con el tutor para barajar las diferentes opciones, pero la más clara era crear una matriz de correlación en la BBDD para replicar la tabla.

El primer intento para convertir la tabla a una matriz de correlación fue crear un método en python que analizase los colores que hay en la tabla, de esta manera según el color que iba encontrando dentro de ciertos parámetros, luego se le podía dar el valor deseado. Esta tabla tiene marcas de agua, por lo que este cometido era muy complicado.

La matriz de correlación R para p variables $X_1, X_2, \ldots, X_p$ se define como:

$$R = \begin{pmatrix} 1 & r_{12} & r_{13} & \cdots & r_{1p} \\ r_{21} & 1 & r_{23} & \cdots & r_{2p} \\ r_{31} & r_{32} & 1 & \cdots & r_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & r_{p3} & \cdots & 1 \end{pmatrix}$$

Al final la decisión pasa por replicar la tabla a mano y luego cargarla en la base de datos. Así que poco a poco se va creando la matriz de correlación en excel asignando los siguientes valores:

- **1** si es verde, compatible.
- **2** si es rojo, incompatible.
- **3** si es amarillo, puede ser peligroso.
- **4** si es azul, no se ha comprobado.

Si se hubiese conseguido la tabla en un formato CSV o en excel, este paso hubiese sido mucho más sencillo. Para cargar la tabla se sigue el mismo proceso que para cargar la tabla de datos de los medicamentos.

En este momento cada punto de API está en su respectivo fichero, luego se unifican en uno solo.

5.6. Comparador Interacciones

El comparador tiene que realizar diferentes funciones, se desglosan en la siguiente lista:

1. Crear una tabla donde se van a guardar los resultados de las comparaciones.
2. Comprobar que puede acceder a los datos tanto de la tabla de medicamentos como de la de compatibilidades.
3. Consultar los medicamentos que se van a comparar y coger los principios activos de cada uno, ya que la comparación se basa en los principios activos.

4. Verificar la compatibilidad de los medicamentos.

5. Devolver los resultados.

Tabla Resultados

Para crear la tabla de resultados, se hace una conexión con la base de datos, en un principio se crea una query para que elimine la tabla si existe para realizar las pruebas. Luego se crear con los campos deseados.

Comprobar Información de Medicamentos

Para consultar los datos simplemente se hace una consulta de todos los datos de la tabla, se devuelve un mensaje JSON para saber si lo ha consultado junto con un mensaje de existo, sino un mensaje de error.

Medicamentos y Principios Activos

En este punto, de los datos de la tabla de medicamentos, únicamente se necesitan los siguientes campos:

- ID
- Nombre del medicamento
- Tipo de fármaco
- Código del laboratorio donde se fabrica
- El estado del medicamento
- Principio activo
- PVP

En el primer intento se comete el error de no utilizar LIKE para que el nombre que le pasamos lo busque en toda la cadena de texto que compone el nombre del medicamento.

Una vez obtiene los campos, hay que realizar una nueva consulta sobre el resultado de la anterior, la cual se encarga de según el nombre que se la ha pasado, guardar el principio activo que contiene.

Se crean listas para almacenar los nombres y los principios activos. Luego se itera sobre todos los resultados para que haga match con el nombre y el principio activo y que vaya guardando en las listas los resultados.

Devuelve un JSON con los datos.

Verificar Compatibilidad

Para verificar la compatibilidad de los medicamentos, se crea una función a la que se le pasan las listas anteriores, se seleccionan todos los datos de la tabla de compatibilidades y se aplica un algoritmo basado en fuerza bruta para compararlos.

Se devuelve una lista con los resultados, si no encuentra la compatibilidad devuelve un error.

Devolver los Resultados

Este último llama a la función de compatibilidad para realizar las comparaciones con los nombres de los medicamentos que se le pasan y llamando a la función de compatibilidades.

Se llama a un método que elimina los duplicados, para reducir la lista, pero no acaba funcionando a la perfección.

Se insertan en la tabla de resultados y devuelve un JSON con los resultado, el cual habrá que pasárselo a la aplicación más adelante.

Añadidos

Uno de los problemas que hubo en esta fase, fue que los principios activos de una fuente y los recogidos de otra, no se llamaban igual, se decide hacer un diccionario para mapear los principios activos.

Otro apartado que se quedó sin solución fue que muchos de los medicamentos se repiten, no exactamente, sino que lo único que cambia es el gramaje, dosis, etc. Se intento normalizar los nombres de los medicamentos cuando se realiza la consulta, pero tampoco dio los resultados esperados.

5.7. Fase 5

Esta última fase del proyecto consiste en realizar las consultas desde la aplicación, que esta comunique con la API y obtenga los resultados.

Para ello toca trabajar en la aplicación Java. Se crea una carpeta de API donde se crear la clase del controlador para el comparador. Es un controlador REST que maneja las solicitudes HTTP del API.

Se cometieron algunos errores, por ejemplo el uso incorrecto del método POST y la falta de permisos en el archivo de seguridad de Spring Boot.

Se configuraron los permisos para CORS [13], que no permitían que se comunicasen.

En la vista de las comparaciones, se crea un formulario para introducir el nombre del medicamento que se quiere buscar, este desencadenando que se ejecute el script para las comparaciones con un evento.

El script, el cual se guarda en un fichero HTML, hace una solicitud GET a la dirección que se le indica, en este caso al punto de API para realizar las comparaciones. Cuando obtiene la respuesta, procesa los datos y los muestra.

Se le añaden estilos para una vista de los resultados atractiva, como símbolos según el resultado obtenido.

5.8. Implementaciones Finales

Las incorporaciones finales que se han hecho en las semanas restantes son las siguientes:

Panel de Gestión de Usuarios

Se implementa un panel de gestión de usuarios al cual solo puede acceder el usuario que tenga el rol de ADMIN. En este panel se puede cambiar datos de los usuarios, habilitarlos y eliminarlos.

Una de las complicaciónes fue que había que integrar lógica utilizando Thymeleaf. Se hace un for each para listar los usuarios. De la misma manera, hubo que cambiar/añadir los controladores necesarios y la seguridad, al realizar los controladores me dieron fallos debido a que los mappers y servicios también necesitaban cambios.

```
<tr th:each="elemento , posicion : ${listausuarios}">
    <td th:text="${elemento.nombreUsuario}"></td>
    <td th:text="${elemento.email}"></td>
    <td sd:authorize="hasRole('ADMIN')">
        <button type="submit" th:classappend="${elemento.active} ? 'btn btn-warning btn-sm' : 'btn btn-success btn-sm'"
                              th:formaction="'/users/'+ ${elemento.id} +'/habilitar'">
            <i class="fas fa-check"></i>
            <span th:text="${elemento.active} ? #{general.deshabilitar} : #{general.habilitar}"></span>
        </button>
    </td>
```

Figura 5.5: Código donde se realiza un for each en Thymeleaf

En este mismo punto, por algo puramente estético, se decidió implementar la funcionalidad para que el botón de habilitar y deshabilitar cambiase según el estado del usuario. En un principio parecía algo bastante sencillo, pero hubo que realizar cambios en cada una de las partes relacionadas, vista, controlador, servicio, etc.

Reset de Contraseña

La última implementación, hacer un reset de contraseña que sea seguro tiene sus pasos a seguir.

1. Incluir las dependencias necesarias de Maven.
2. Definir las variables de entorno necesarias en el archivo application.properties
3. Crear las vistas.
4. Crear el DTO para los correos electrónicos.
5. Crear el servicio.
6. Implementar los controladores (en este apartado se complicó el tratamiento del token).

Google cobra por utilizar sus servicios de correo, por lo que se compra el dominio appactivus.com y se utiliza el servicio Mailtrap (emula un servidor de correo). Se realizan las configuraciones necesarias en el dominio para poder enviar correos en la aplicación.

```java
@GetMapping("/users/resetpass/{username}/{token}")
public String cambiopass(@PathVariable("username") String username, @PathVariable("token") String token, ModelMap intefrazConPantalla) {

    Optional<Users> usuario = service.getRepo().findByNombreUsuarioAndTokenAndActiveTrue(username,token ); // Buscar usuario por nombre de
    System.out.println(username + ":" + token );
    UsersDtoPsw usuarioCambioPsw = new UsersDtoPsw();

    // Si el usuario existe, se muestra la vista de restablecer la contraseña
    // Si no existe, se muestra la vista de datos no encontrados
    if (usuario.isPresent()){
        usuarioCambioPsw.setId(usuario.get().getId());
        usuarioCambioPsw.setNombreUsuario(username);
        usuarioCambioPsw.setEmail(usuario.get().getEmail());
        usuarioCambioPsw.setPassword("******************"); // Se oculta la contraseña
        usuarioCambioPsw.setNewpassword("******************"); // Se oculta la contraseña
        intefrazConPantalla.addAttribute( attributeName: "datos", usuarioCambioPsw);
        return "users/resetearPassword";
    }else {

        //Mostrar página usuario no existe
        return "users/datosNoEncontrados";
    }
}
```

Figura 5.6: Captura donde se muestra uno de los controladores para el reset de contraseña, hay que utilizar el token del usuario (es único) para garantizar la seguridad

6. Trabajos relacionados

Antes de comenzar este trabajo era necesaria la consulta y conocimiento de los dos apartados que a continuación se presentan.

6.1. Vademecum

En primera instancia, se conoce que Vademecum es un gran referente en el ámbito farmacológico. Por lo que se investiga las opciones y funcionalidades que ofrece la página.

En el primer análisis se ve que integra opciones como listar medicamentos, consultar la información de estos, orquestar listados, etc.

Tiene funciones de pago y una aplicación móvil, entre estas funciones esta una funcionalidad para consultar las interacciones entre medicamentos, la cual se descubre casi al final de la elaboración del proyecto.

Página oficial: https://www.vademecum.es/

6.2. Proyecto Farmacia

Se estuvo estudiando un proyecto realizado por Borja Moreno, de las asignaturas DAM1 y DAM2.

El proyecto esta realizado en Java, por lo que era perfecto para empezar a hacerse a la idea del futuro desarrollo.

Este proyecto permite recuperar información de medicamentos, borrar medicamentos y añadir nuevos.

Repositorio: https://github.com/

7. Conclusiones y Líneas de trabajo futuras

Este proyecto no es el final, es el comienzo del desarrollo más amplio de todo lo que he presentado en este trabajo. Sin duda, supone una motivación para seguir trabajando en la mejora de las funcionalidades y servicios debe ofrecerse tanto a especialistas del sector como a pacientes del mismo.

7.1. Conclusiones

Dificultad de Estructura

Una de las conclusiones más claras que se saca de la realización de este proyecto es la dificultad para estructurar, comprender y organizar todas las partes que componen una aplicación web, independientemente de las tecnologías utilizadas.

El no comprender las relaciones entre los servicios y los repositorios o la utilidad de los mappers pueden hacer que la curva de dificultad se incremente de sobremanera.

De esta manera el rumbo o progreso en el desarrollo se hace complicado, ya que no se sabe que ficheros o clases hay que completar para poder pasar a las siguientes fases.

Tecnologías Utilizadas

Si se decide utilizar tecnologías nuevas, la dificultad para comprender los errores que se cometen y el por qué se cometen es enorme, teniendo que

buscar información en incontables páginas o repositorios, como puede ser https://stackoverflow.com/.

En este caso la mayor dificultad fue comprender el funcionamiento de Spring o Spring Boot, el método para entenderlo tampoco fue óptimo, ya que se aprendió a base de prueba, error y búsqueda de solución. Todo ello sin entender o seguir el camino del desarrollo en su mejor forma.

Podría decir que el correcto uso de los servicios, repositorios y controladores es donde hay que enfocar la dificultad. En la mayoría de las ocasiones, un pequeño cambio para implementar una nueva funcionalidad requiero de diversos cambios en estas clases.

Bases de Datos

Otra conclusión que es relevante es la necesidad de conocer el funcionamiento de las bases de datos. Esto incluye:

- Conocer la importancia de la nomenclatura de los campos en la base de datos y las buenas prácticas.
- La importancia de las relaciones entre diferentes tablas.
- La capacidad de realizar consultas de manera efectiva, conociendo el amplio abanico de opciones que hay.

En mi caso, se inicia el proyecto con el conocimiento justo para realizar una consulta SELECT de campos de una tabla en específico. Pudiendo realizar que el resultado se devuelva por ejemplo ordenado.

Se adquieren conocimientos como el uso de la clausula DISTINCT (elimina valores duplicados) o de LIKE (compara un valor dentro de una expresión). De gran ayuda en la elaboración de las consultas que realiza la API.

7.2. Líneas de Trabajo Futuras

Actualmente la aplicación se encuentra en una etapa de desarrollo inicial, a falta de cambios significativos.

Comparador Mejorado

El comparador o la lógica que se utiliza para analizar las interacciones entre los medicamentos es básica. En este momento únicamente es capaz de ir comparando parejas de medicamentos que contengan solo un principio activo.

Una aplicación que devuelva unas interacciones completas debería ser capaz de ir analizando la compatibilidad entre medicamentos que contengan más de un principio activo. Una vez completado esto, también debería ser capaz de analizar las interacciones entre más de dos medicamentos.

Estos sería las dos mejoras sustanciales en este aspecto, pero si se requiere algo óptimo se podría añadir información de estudios o prospectos de medicamentos donde se pueda obtener información de las incompatibilidades de estos con diferentes enfermedades o síntomas. Por ejemplo, que un medicamento que contenga azúcar no se pueda administrar o se releve de las interacciones si se indica que el paciente es diabético.

Otro aspecto destacable, es que según la comunidad o provincia donde se utilice la aplicación, habría que darle más peso a ciertos medicamentos. Por encima de los medicamentos, quizá sea el laboratorio o empresa que los produce, ya que la sanidad, hospitales o farmacias de cada provincia suele tener acuerdos o convenios con las farmacéuticas de la zona para suministrar sus productos por encima de otros.

Clasificador de Medicamentos

Implementar una vista que clasifique los medicamentos, por ejemplo, si el usuario busca los medicamentos que sea antiinflamatorios, que devuelva una lista con dichos medicamentos.

Para ello la idea sería disponer de los prospectos de los medicamentos e incluir estas descripciones en la base de datos, luego buscando la palabra en cuestión en estas descripciones que vaya clasificándolos. Se podrían añadir en una nueva tabla para consultas futuras.

Hístorico de Pacientes o Seguimiento

Proporcionar la funcionalidad de ofrecer el acceso tanto al histórico de consultas como al histórico de pacientes. Donde se podrá ver y analizar los datos para un mejor uso del comparador.

Este apartado tiene muchos temas que revisar con la ley de protección de datos.

Más Fuentes de Datos

En el proyecto se han utilizado dos fuentes de datos, la proporcionada por el ministerio de sanidad (listado de medicamentos) y la tabla de compatibilidades (en Y).

El listado de medicamentos es muy completo, pero la tabla de compatibilidades únicamente recoge una pequeña parte de la enorme cantidad de principios activos que hay. Según la forma en la que se administran estas relaciones pueden cambiar, incluir efectos diferentes en los resultados o añadir principios activos que van cambiando según como se administran los medicamentos.

Interfaz de Usuario

La interfaz de usuario no es visualmente agradable, pero si se quiere enfocar en la usabilidad habría que realizar cambios significativos. Entre ellos se destaca la muestra de tablas de interacciones o la posibilidad incluso de ver una imagen del medicamento.

Todo ello acompañado de estilos diferentes, sin dejar atrás la modernidad de las aplicaciones de estos tiempos.

Bibliografía

[1] Keep Coding. El porqué usar javascript: ventajas y desventajas. https://keepcoding.io/blog/porque-usar-javascript/, 8 de abril de 2024, última actualización.

[2] Ministerio de Sanidad. Información sobre los productos incluidos en la prestación farmacéutica del sns (dispensables a través de oficinas de farmacia). https://www.sanidad.gob.es/profesionales/nomenclator.do, Actualizado.

[3] Yúbal Fernández. Api: qué es y para qué sirve. https://www.xataka.com/basics/api-que-sirve, 23 de Agosto de 2019.

[4] Yúbal Fernández. Chatgpt: qué es, cómo usarlo y qué puedes hacer con este chat de inteligencia artificial gpt. https://www.xataka.com/basics/chatgpt-que-como-usarlo-que-puedes-hacer-este-chat-inteligencia-artificial, 9 de Mayo de 2024.

[5] Darwin Galindo. ¿qué es thymeleaf? https://blog.todotic.com/que-es-thymeleaf/, 20 de Septiembre de 2021.

[6] IBM. ¿qué es java spring boot? https://www.ibm.com/es-es/topics/java-spring-boot#:~:text=Spring%20Boot%20permite%20a%20los,durante%20el%20proceso%20de%20inicializaci%C3%B3n., -.

[7] IBM. Plantillas jinja2. https://www.ibm.com/docs/es/guardium/11.4?topic=sdk-jinja2-templates, 12 de Octubre de 2021.

[8] IBM. Java persistence api (jpa). https://www.ibm.com/docs/es/was-liberty/nd?topic=liberty-java-persistence-api-jpa, 30 de Enero de 2024.

[9] IONOS. ¿qué es flask python? un breve tutorial sobre este microframework. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask/, 1 de Marzo de 2023.

[10] IONOS. ¿qué es html (hyper text markup language)? https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-html/, 13 de Octubre de 2023.

[11] G. Castells Lao. Compatibilidad de los fármacos administrados en «y» en las unidades de cuidados intensivos: revisión sistemática. https://www.medintensiva.org/es-compatibilidad-farmacos-administrados-y-unidades-articulo-S0210569118302432#:~:text=La%20administraci%C3%B3n%20de%202%20f%C3%A1rmacos,precipitaci%C3%B3n%20o%20cambio%20de%20color., 4 de Agosto de 2018.

[12] Pablo Londoño. Qué es bootstrap, para qué sirve y cómo funciona. https://blog.hubspot.es/website/que-es-bootstrap, 16 de mayo de 2023.

[13] MDN. Intercambio de recursos de origen cruzado (cors). https://developer.mozilla.org/es/docs/Web/HTTP/CORS, -.

[14] MDN. ¿qué es javascript? https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript, -.

[15] Microsoft. Visual studio code. https://visualstudio.microsoft.com/es/, Fecha de lanzamiento 29 de Abril de 2025.

[16] NextU. ¿qué es css y para qué sirve? https://www.nextu.com/blog/que-es-css-rc22/, -.

[17] Vicente Sancho. Ahorrar tiempo y esfuerzo en desarrollos java con lombok. https://vicentesg.com/ahorrar-tiempo-y-esfuerzo-en-desarrollos-java-con-lombok/#:~:text=Lombok%20es%20una%20biblioteca%20de,escribir%20c%C3%B3digo%20repetitivo%20y%20tedioso., -.

[18] Data Scientest. Pandas : La biblioteca de python dedicada a la data science. https://datascientest.com/es/pandas-python, 19 de Diciembre de 2022.

[19] Skysnag. ¿qué es bcrypt? https://www.skysnag.com/es/blog/what-is-bcrypt/#:~:text=Bcrypt%20es%20una%20funci%C3%B3n%20de,OpenSSH%20emplean%20este%20algoritmo%20hash., 11 de Octubre de 2023.

[20] Spring. Spring initializr. https://start.spring.io/, -.

[21] Spring. Transaction management. https://docs.spring.io/spring-framework/reference/data-access/transaction.html, -.

[22] Vademecum. Página oficial de vademecum. https://www.vademecum.es/, -.

[23] Wikipedia. Blowfish. https://es.wikipedia.org/wiki/Blowfish, 7 de Abril de 2024, última actualización.

[24] Cecilio Álvarez Caules. ¿qué es spring boot? https://www.arquitecturajava.com/que-es-spring-boot/, 7 de Octubre de 2023.