Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Plataforma software de ayuda a la toma de decisiones para facultativos durante la prescripción de medicamentos

Estudante: Carlos Sousa González
Dirección: Fernando Bellas Permuy

A Coruña, xuño de 2021.

A mi familia, por ser un apoyo indispensable

Agradecimientos

En primer lugar, quiero dar las gracias a mi familia. Sin ellos no podría haber llegado a donde estoy. Especialmente a mi padre, que me ha ayudado durante todo el desarrollo de este proyecto, participando activamente en las decisiones tomadas. También a mi madre, por enseñarme que sin esfuerzo y constancia no se consiguen las metas más satisfactorias.

Es imposible olvidarme de mi hermano, que me ha enseñado la gran importancia de la autocrítica para mantener los pies en el suelo.

A mi pareja Sara, por ser el apoyo que todo el mundo necesitaría en aquellos momentos de flaqueza. Espero que podamos seguir creciendo juntos.

A mis amigos, los que siempre están ahí, aún con las circunstancias pasadas y actuales seguiremos muy unidos. Ya queda menos para celebrar todo aquello que no hemos podido celebrar.

A Hugo y Lois, por ayudarme tanto en mi desarrollo profesional como personal.

A María Luisa, jefa del departamento de informática del hospital de Monforte, por ayudarme durante todo el proyecto siempre que lo he necesitado, gracias por tu amabilidad y disponibilidad.

Y por último a Fernando, por confiar en mí para este proyecto y brindarme tanto consejos como esas palabras de apoyo en los momentos en los que creía que el proyecto no llegaría a un resultado satisfactorio.

Resumen

El objetivo principal de este proyecto se basa en el desarollo de una aplicación médica, que permita reconocer la problemática inherente al recetado de medicamentos. Esta problemática reside en la incompatibilidad o necesidad de ajuste de dósis de un componente químico, y por consecuencia del medicamento que lo contiene, debido a diversos factores del individuo a analizar (e.g. alergias, intolerancias, enfermedades, edad, sexo, otros medicamentos que está tomando, etc.).

El núcleo de la aplicación se basa en una *API REST*, que contiene la lógica de negocio necesaria para diagnosticar dichos problemas. Este servicio se implementa utilizando tecnologías *J2EE*, apoyandose en el *framework Spring Boot*. Para la persistencia de datos se utiliza *JPA*, en una base de datos *MySQL*.

En cuanto al *frontend*, se basa en una implemetación web del lado servidor, apoyandose en las librerías *Thymeleaf* y *jQuery*.

Abstract

The main objective of this project is based on the development of a medical application, which allows to recognize the problems inherent in the prescription of drugs (chemical substances or trade names). This problem resides in the incompatibility or need to adjust the dose of a chemical component, and as a consequence of the drug that contains it, due to various factors of the individual to be analyzed (eg allergies, intolerances, diseases, age, sex, other medications that are taking, etc.).

The core of the application is based on a *API REST*, which contains the business logic necessary to diagnose such problems. This service is implemented using *J2EE* technologies, relying on the Spring Boot framework. *JPA* is used for data persistence, in a *MySQL* database.

As for the *frontend*, it is based on a web implementation on the server side, relying on the *Thymeleaf* and *jQuery* libraries.

Palabras clave:

- Farmacología
- Prescripción segura
- Integración
- Java
- MySQL
- JPA
- Spring Boot
- Thymeleaf

Keywords:

- Pharmacology
- Safe prescription
- Integration
- Java
- MySQL
- JPA
- Spring Boot
- Thymeleaf

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La farmacología es una ciencia multidisciplinaria que surgió como tal en el siglo XIX, originalmente como una rama de la fisiología experimental, y posteriormente como una ciencia independiente dentro la medicina. Sin embargo lleva ligada a la propia medicina desde sus orígenes, denominándose ”Materia médica” en lugar del término actual, y reflejándose en pequeños intentos de aplicarla ya desde el 3000 a.C. en la medicina del imperio chino, pasando por otras civilizaciones como la mesopotámica, la egipcia, la romana o la griega, de la que hereda uno de sus símbolos más representativos, la copa de Higía.

Esta ciencia ha estado ligada a continuos cambios y descubrimientos tanto accidentales como producto de investigaciones destacadas, desde su estandarización hasta la actualidad. Por esta razón tanto médicos como farmacéuticos continúan en formación permanente una vez acabado su aprendizaje académico.

Con los avances tecnológicos y gracias a la sociedad de la información, se ha conseguido compartir estos descubrimientos a nivel global y estandarizado.

Aun con toda esta información disponible, los profesionales médicos no disponen de un tiempo ilimitado para dedicarle a cada paciente. Teniendo que realizar manualmente las comprobaciones de incompatibilidades, debidas a diversos factores (alergias, intolerancias, enfermedades, medicamentos que ya consume el paciente, edad...), siempre estarán sujetos a fallos humanos.

Según un estudio realizado en 2020 por el Sistema Español de Notificación en Seguridad

de Anestesia, los errores de medicación son la tercera causa de muerte en España, solo por detrás de las causadas por las enfermedades cardiacas y el cáncer, y el 80% de ellos podrían prevenirse. [1]

En aras de reducir esta problemática, la Organización Mundial de la Salud impulsó en el año 2017 el reto "Medicación sin daño", en el que se fijó como objetivo reducir a la mitad los daños graves y evitables relacionados con la medicación en todos los países en un plazo de cinco años.

Sin embargo, los médicos rechazan los programas del tipo de *"algoritmos terapéuticos"* (el software es el encargado de la prescripción) porque limitan y coartan su libertad e independencia a la hora de recetar mediación a un paciente [2] [3].

Hasta donde he podido indagar, debido a que no son públicos, los sistemas informáticos de salud no proporcionan una aplicación que permita al médico comprobar si el medicamento a recetar presenta algún tipo de incompatibilidad con respecto a su historial médico. Por contra, el peso de esta comprobación recae totalmente en dicho profesional.

1.1 Objetivos

El objetivo de este proyecto se basa en el desarrollo de un software que permita a los médicos, de forma sencilla y automatizada, comprobar las incompatibilidades de un fármaco, cotejándolo con la historia clínica del paciente.

Esta aplicación se desarrolla en un marco académico y totalmente ajena a cualquier sistema sanitario real. Por tanto, la solución presentada actúa como una prueba de concepto, que requeriría de los debidos ajustes necesarios si se tuviese la ocasión de integrarla dentro de un sistema de salud concreto.

Para ello, se desarrolla un *backend* con toda la lógica de negocio necesaria y una aplicación web que permita su uso vía navegador, rellenando manualmente los datos del historial médico y seleccionando el medicamento que se pretende recetar, por su nombre comercial o genérico para facilitar el uso.

Para el control de permisos, se plantea una dinámica de roles. El rol más básico sería el

de *"Invitado"*, este rol está orientado al uso pedagógico de la información, por ejemplo, un estudiante de medicina que desease comprobar algún caso hipotético. El rol *"Administrativo"* permite el control de la información de la base de datos, pudiendo gestionar las tablas de alergias, intolerancias, enfermedades, medicamentos, etc. Por último, el rol *"Médico"*, está pensado de cara a la integración con otros sistemas de salud, éste permitirá realizar operaciones que usen información delicada del sistema ajeno.

La aplicación web descrita anteriormente es completamente funcional sin necesidad de integrarse en un sistema de salud. Sin embargo, si existiese esta posibilidad, evitaría que el médico tuviese que introducir los datos del paciente manualmente. Para ello se proponen dos escenarios de integración.

El primer escenario, está orientado al uso de la aplicación web. Pero con el objetivo de evitar que el médico tenga que rellenar todo los datos manualmente, como se propone en el uso básico de la aplicación. Se pretende que mediante el uso de la interfaz gráfica de la aplicación, se rellene algún tipo de dato identificativo del paciente (e.g. DNI, numero de historia, número de seguridad social, NASI...). Tras esto, el sistema consumirá un servicio web externo proporcionado por el sistema de salud, que recibiendo este dato identificativo devolverá los datos necesarios de su respectivo historial médico. Posteriormente, se mostrarán dichos datos en la interfaz gráfica del sistema a desarrollar.

El segundo escenario, está orientado al uso por parte de un médico, de una aplicación sanitaria externa que, con los cambios necesarios, pudiese consumir el servicio web proporcionado por el sistema a desarrollar. La aplicación externa, por sus propios medios, obtendría los datos necesarios del historial clínico y junto con un medicamento seleccionado, los enviaría al servicio web. Tras esto, el servicio web procesaría la información dada y le devolvería una serie de alertas, que posteriormente serían mostradas en la interfaz gráfica de dicho sistema de salud.

1.2 Visión global del sistema

El sistema consistirá en una aplicación web del lado servidor implementada utilizando las tecnologías Thymeleaf y jQuery, con la que el usuario podrá interactuar para acceder a las distintas funcionalidades disponibles.

Por último, se usará MySQL como base de datos.

En la figura 1.1 se representan los casos descritos a continuación.

Se comenzará explicando el uso de la aplicación sin ningún tipo de integración. Esto englobaría 2 casos:

El primero, representado por las flechas azules. Un usuario con el rol ”Invitado” o ”Médico” que accede a la aplicación web utilizando la interfaz médica, rellena los datos del historial médico del paciente manualmente en un formulario. Tras esto, se le muestran la lista de alertas relacionadas con el caso propuesto.

El segundo, representado por las flechas grises. Un usuario con el rol ”Administrador” que accede a la aplicación web utilizando la interfaz administrativa, realiza algún tipo de gestión sobre la base de datos (alergias, intolerancias, enfermedades, medicamentos, etc). Dichos cambios se persisten y los cambios se ven reflejados en dicha interfaz.

Como se menciona en el apartado anterior, si se pudiese integrar con un sistema de salud concreto, el sistema propuesto está preparado para soportar dos tipos de integración.

El primero, la importación de la información del historial médico a partir de un dato identificativo, representado por las flechas amarillas. Un usuario con el rol ”Médico” accedería a la aplicación web utilizando la interfaz médica. Tras esto, tendría la posibilidad de introducir un dato identificativo, se ha implementado con el DNI. La aplicación web consumiría el ”Servicio web DNI” externo, al cual le facilitaría el dato identificativo dado. Este servicio externo, le devolvería la información necesaria del historial médico del paciente dado. Por ultimo, dicha información se rellenaría en la interfaz médica, permitiendo al médico realizar cambios posteriores a la importación.

El segundo, que una aplicación de salud externa consuma el servicio web de incompatibilidades, representado por las flechas rojas. Un médico accedería a una aplicación de salud externa con su propia interfaz gráfica. Allí, seleccionaría el medicamento que quiere recetar, posteriormente esa aplicación consumiría el servicio web de incompatibilidades. Éste, recibiría la información necesaria del historial médico del paciente junto con el medicamento a valorar, lo procesaría, y devolvería la lista de alertas resultante. Finalmente, la interfaz gráfica de la aplicación de salud mostraría estas alertas al médico.

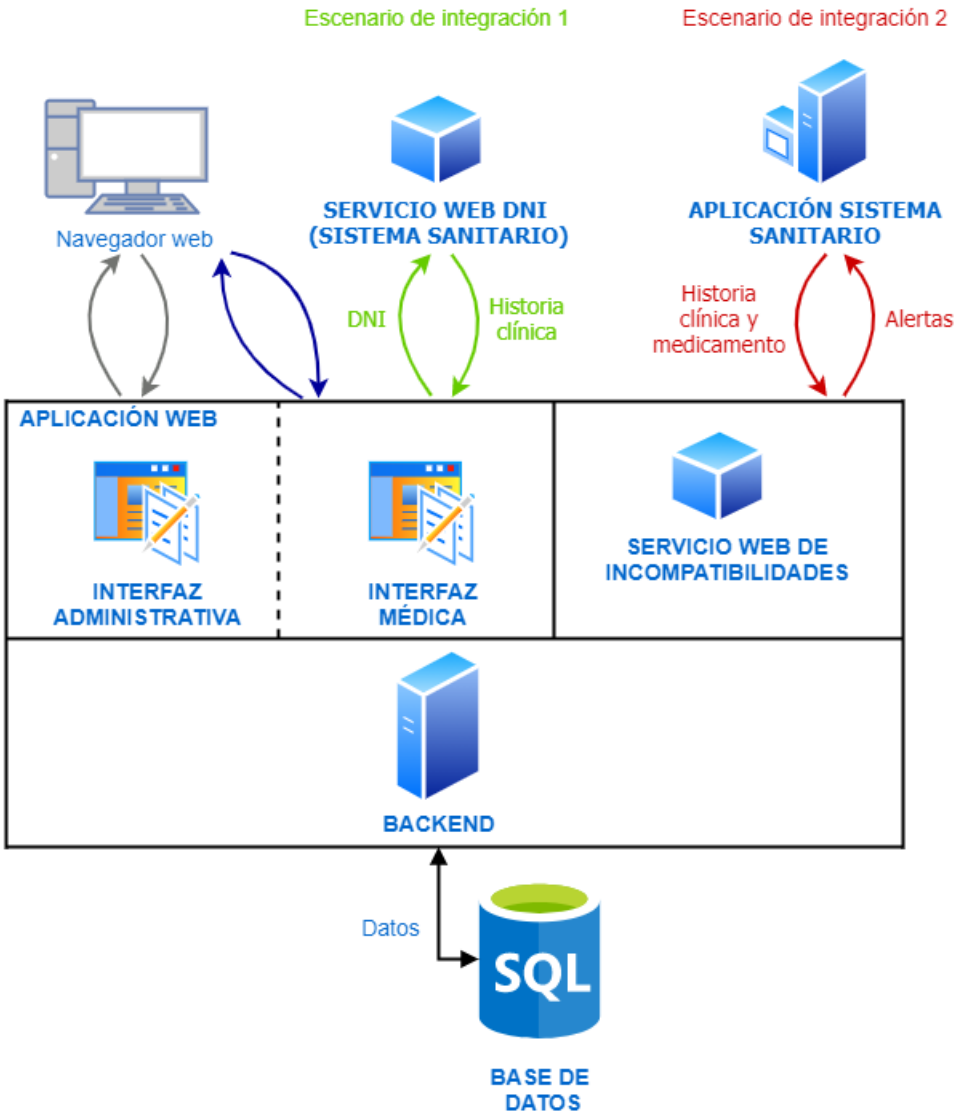

Figura 1.1: Arquitectura del sistema

‘

Capítulo 2

Modelo de Incompatibilidades de Componentes Químicos en Medicamentos

En el ámbito de la farmacología existen múltiples factores a tener en cuenta en el recetado de un medicamento. En este capítulo se describe el estudio realizado sobre dicho campo y los distintos esquemas obtenidos para el desarrollo de la aplicación.

Se comenzará explicando el modelado semántico de dicho campo. Posteriormente se analizarán en detalle las distintas categorías de incompatibilidades planteadas.

2.1 Modelado semántico

A continuación, se explicarán cada uno de los elementos representados en el modelo semántico (2.1).

- **Medicamento:** Representa el producto final de la fabricación, dicha fabricación resulta de la preparación de la fórmula a la que se le añaden un conjunto de excipientes.
- **Excipiente:** Sustancia que se mezcla con los fármacos para darles consistencia, forma, sabor u otras cualidades que faciliten su uso. En la aplicación se ha obviado dicha

relación debido a su casi inexistente probabilidad de incompatibilidades.

- **Fórmula:** Denominación científica que se le otorga a un medicamento, en ella se recoge la composición expresada en componentes químicos.
- **Componente químico:** Denominado en la jerga como *"principio activo"*, es la parte del medicamento que le otorga las cualidades terapéuticas.
- **Incompatibilidad:** Define el problema del uso de un medicamento al ser utilizado en un caso determinado. Este problema, puede derivar en la privación del uso de dicho componente químico o en la necesidad de ajuste de dosis para aplicarlo al caso dado.

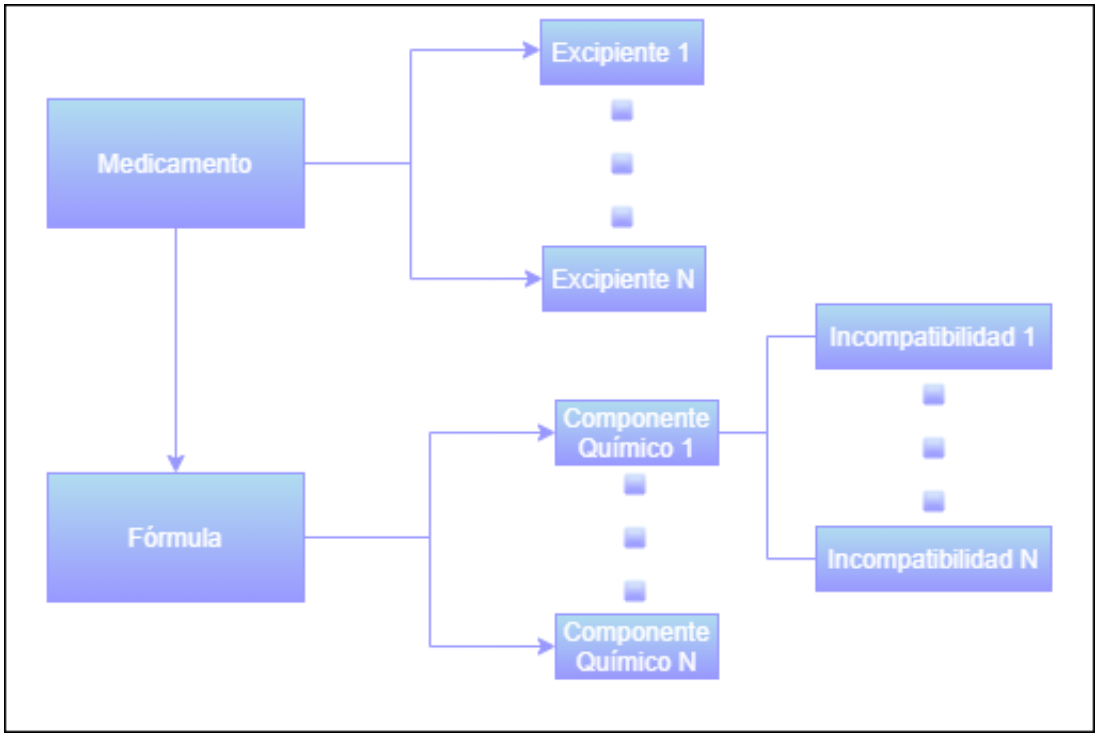

Figura 2.1: Modelo Semántico

2.2 Categorías de incompatibilidades

Un componente químico puede mostrar incompatibilidades por diversos factores a los que está sometido el paciente. Para intentar abarcar la mayoría de casos se han definido los siguientes tipos:

- **Incompatibilidad genérica:** Recoge todos los casos comunes por los que un componente químico puede presentar una incompatibilidad (edad, sexo, embarazo, lactancia, problemas renales e insuficiencia hepática).
- **Incompatibilidad con enfermedad:** Recoge aquellos casos en los que no se puede administrar cierto componente químico debido a que el paciente pacede o ha padecido cierta enfermedad.
- **Incompatibilidad con alergia:** Recoge aquellos casos en los que no se puede administrar cierto componente químico debido a que el paciente padece alergia hacia dicho componente.
- **Incompatibilidad con intolerancia:** Recoge aquellos casos en los que no se puede administrar cierto componente químico debido a que el paciente padece una intolerancia relacionada con dicho componente.
- **Incompatibilidad con otro componente químico:** Recoge aquellos casos en los que existe una interacción farmacológica del componente químico con otro dado. Este caso también se aplica a aquellos componentes pertenecientes a la misma familia, ya que no suele ser recomendable su uso simultáneo.

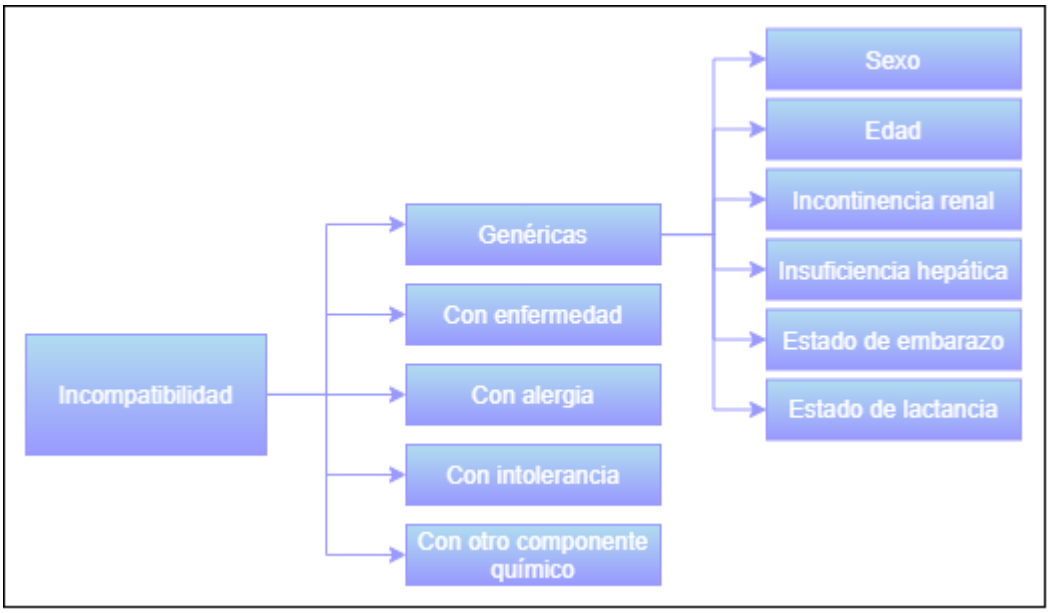

Figura 2.2: Esquema de incompatibilidades

‘

Capítulo 3

Metodología

En el mundo del desarrollo software, la metodología sirve como marco de referencia para la estructuración, planificación y correcto desarrollo de un proyecto. En este caso, como en la mayoría de proyectos actuales, se ha optado por una metodología ágil debido a los siguientes beneficios:

- Mejora la productividad durante el desarrollo y la calidad del producto final.
- Se adapta a cambios durante el desarrollo, minimizando así su impacto en el proyecto.
- Aumenta la participación del cliente interesado en el producto.
- Sigue un proceso iterativo e incremental, por lo cual se obtiene software funcional al final de cada etapa.
- Ofrece una visión estandarizada de las metas a conseguir en cada etapa, esto aumenta la focalización y motivación del equipo de desarrollo.

Para este proyecto se ha elegido una metodología iterativa propia, heredando alguno de los principios de *Scrum* [4]. Ya que esta metodología no es aplicable a un proyecto desarrollado individualmente en lugar de con un equipo.

3.1 Adaptación de Scrum

Scrum es una de las metodologías ágiles más utilizadas en la actualidad, se desarrolló a principios de los 90 aunque su uso se popularizó más tarde. Se basa en un proceso iterativo, en el que se aglomeran un conjunto de buenas prácticas para trabajar colaborativamente y obtener el mejor resultado posible en el desarrollo de un proyecto. Estas prácticas emanan del estudio de equipos de trabajo altamente productivos.

El uso de *Scrum* tiene un gran rendimiento en proyectos con un marco complejo o con requisitos poco definidos y sujetos a cambios. Una de sus principales características es la entrega monitorizada de segmentos del producto final, pactados y priorizados previamente con el cliente.

La base fundamental de esta metodología, y en la que se basa la metodología propia escogida, es la división del desarrollo del trabajo en plazos de corta duración denominados *sprints*, así como la definición de roles, documentos y reuniones que se mantienen en cada una de estas iteraciones.

En los siguientes apartados, abarcaremos mas en detalle los puntos heredados de la metodologia *Scrum*.

3.1.1 Roles

El objetivo de los roles en *Scrum* es crear equipos autoorganizados y multifuncionales, en los que el conjunto de competencias de todos los miembros permita una realización correcta del proyecto. Este método aumenta la flexibilidad, creatividad y productividad a la hora de realizar cualquier tarea.

Scrum define tres roles imprescindibles que todo equipo debe tener: el propietario del producto (*Product Owner*), el equipo de desarollo (*Scrum Team*) y el responsable del equipo (*Scrum master*).

Product Owner

El *Product Owner* es la figura encargada de representar al cliente. Debe transmitir las ne-

cesidades de negocio al equipo de desarrollo para incrementar el valor del producto creado. Tiene la responsabilidad de definir los requisitos del *Product Backlog*, priorizarlos y validarlos.

En el caso de este proyecto, la función de *Product Owner* ha estado repartida entre tres personas, el alumno, que ha realizado la tarea de definir el *Product Backlog*, y tanto un médico como el tutor de este proyecto, que han aportado posibles ideas y validado los requisitos planteados.

Scrum Team

El *Scrum Team* es el equipo de desarrolladores que se encarga, del desenvolvimiento y entrega del producto en cada *sprint*. El tamaño de este equipo es varible, pero la cantidad óptima está entre 5 y 9 personas. Debe ser multidisplinar para abarcar todas las necesidades del proyecto, los perfiles mas comunes son programador, diseñador, arquitecto de sistemas, tester, etc. El propio equipo es el responsable de estimar y coordinar las tareas de cada iteración.

Debido a que el proyecto se ha realizado de manera individual, no se ha incluido el rol de *Scrum Master* en la metodología utilizada. Se ha asumido que el alumno es el encargado de ponerse en contacto con el médico que demandaba el producto y posteriormente analizar y diseñar sus peticiones.

3.1.2 Sprints

Scrum define el *sprint* como un bloque temporal, normalmente entre una y cuatro semanas, en el que se va a desarrollar un incremento.

Las fases de cada *sprint* son: análisis, diseño, implementación y pruebas. Al inicio de cada *sprint* se seleccionan de la lista total de requisitos del proyecto, *Product Backlog*, una serie de tareas que se quieren desarrollar, esto resulta en una sublista llamada *Sprint Backlog*, las tarea que conforman esta sublista serán los objetivos a desarrollar en ese incremento. También se llevan a cabo una serie de reuniones que se detallan en los apartados siguientes. Así, al finalizar la iteración, se obtiene un incremento del producto, es decir, un software funcional para el cliente.

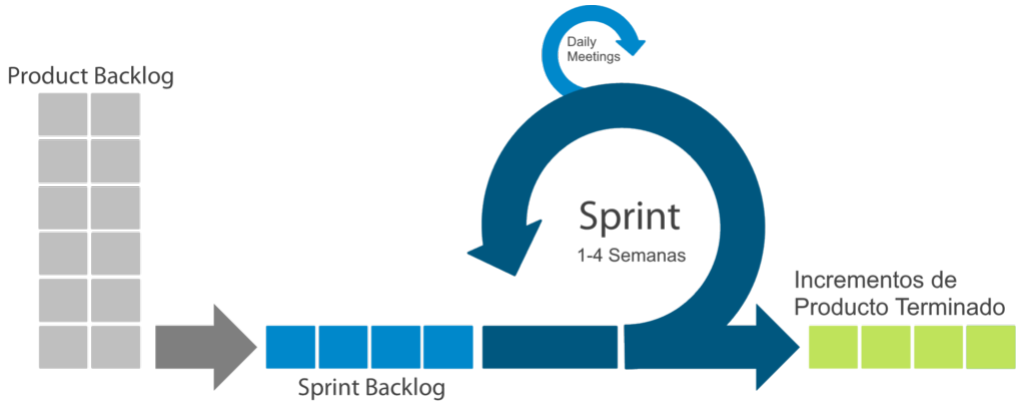

Figura 3.1: Esquema de Scrum

3.1.3 Artefactos

Scrum propone una serie de artefactos con los cuales garantizar la transparencia del estado del proyecto y la conformidad de ambas partes en los objetivos a alcanzar. Facilitando así la organización y toma de decisiones.

Product Backlog

El *Product Backlog* es la lista de requisitos globales del proyecto. Resulta de analizar los requisitos del propio cliente, así como otros de posible interés para los usuarios finales. Al tratarse de una metodología ágil puede cambiar o aumentar su tamaño a lo largo del desarrollo.

Estas necesidades están descritas en forma de historias de usuario, en un lenguaje poco técnico y asequible por ambas partes. El *Product Owner* es el encargado de mantener esta lista actualizada y priorizada, reordenándola si fuese antes de comenzar cada incremento.

En el caso de este proyecto, el *Product Backlog* ha sido desarrollado por el alumno en consonancia con las ideas aportadas y requisitos aportados por un médico.

Sprint Backlog

El *Sprint Backlog* es una sublista del *Product Backlog* abordada durante un incremento. Para favorecer la productividad y el seguimiento, cada una de las tareas debe ser de corta duración, en torno a una jornada de trabajo, y focalizada en el desarrollo de un objetivo específico. Los

propios miembros son los encargados de seleccionar qué tareas llevar a cabo y estimar su duración, comparando dicha duración con la estimada en la preparación del *sprint* y actualizándola si fuese necesario.

Los *Sprint Backlogs* han sido priorizados por el alumno para facilitar el desarrollo, debido a que los resultados de cada incremento solo serían validados y no utilizados.

Historias de usuario

Las *historias de usuario* son la técnica más utilizada para la recopilación de requisitos en las metodologías ágiles. Su utilidad reside en la fácil comprensión por parte del cliente, así como en evitar gran cantidad de documentación formal y su fácil modificación.

El objetivo de las *historias de usuario* es recoger cada funcionalidad que el producto debe incorporar. Se determinan desde el punto de vista del usuario, de forma breve y empleando un lenguaje coloquial. Siguen la estructura "Como **[rol]** quiero **[acción]** para **[objetivo]**".

3.1.4 Reuniones

Para asegurar la transparencia en el avance y la comunicación asertiva dentro del proyecto, *Scrum* propone una serie de reuniones. En ellas se pretende una comunicación eficaz entre los distintos roles.

Sprint Planning

Esta reunión se realiza al inicio de cada *sprint*. Tiene dos partes claramente diferenciadas. En la primera, se revisa el *Product Backlog*, por si es necesario reordenarlo, y se decide que requisitos se van a desarrollar en dicho *sprint*. En la segunda, se desglosa cada requisito escogido en tareas y se estima su duración. Como resultado se obtiene el *Sprint Backlog* y una duración aproximada del sprint.

Este proceso se ha llevado a cabo con la revisión con el tutor, basándose en el *product backlog* ya definido junto con el médico.

Daily Scrum

Al principio de la jornada se realiza esta reunión. Debe ser de corta duración y cada miembro debe responder a tres preguntas:

- ¿Qué hice ayer?
- ¿Qué haré hoy?
- ¿Qué dificultades encontré?

Con estas breves explicaciones, todos los miembros conocen el estado actual del *sprint* y ayudarse entre sí identificando los impedimentos.

Debido a que no existe un equipo de desarrollo, esta reunión se ha obviado, aunque se ha aplicado la filosofía que reside en las tres preguntas anteriores a la hora de organizar el trabajo diario.

Sprint Review

La *Sprint Review* tiene lugar al final del *sprint*, su objetivo reside en la revisón del cumplimiento de objetivos del mismo. Normalmente por medio de una demo, se muestra al *Product Owner* el producto obtenido de ese incremento.

Este tipo de reunión se realizó previa al *Sprint Planning* en cada reunión con el tutor, para mostrarle los avances obtenidos en la anterior iteración.

Sprint Retrospective

El objetivo de la *Sprint Restrospective* es revisar como se abordaron los problemas ocurridos durante el *sprint*, con el fin de identificar áreas de mejora y los aspectos positivos de ese *sprint* para aplicarlos posteriormente. Con esto se consigue una mejora continua a lo largo del proyecto.

Esta reunión se ha realizado junto con la *Sprint Review* con las opiniones dadas por el tutor en dicha validación. Posteriormente, se ha sido reflexionada e interiorizada por el alumno antes del comienzo del siguiente incremento.

Capítulo 4

Análisis de Requisitos

En este capítulo se pretende obtener una visión global de las objetivos planteados para el sistema. Para ello se comenzará describiendo los distintos roles que interactúan con la aplicación y posteriormente se definirá el *Product Backlog* acompañado de las historias de usuario que describen cada funcionalidad, tal y como se explica en el capítulo 3.

Resulta importante aclarar que, el software resultante de este proyecto, no pretende limitar ni coartar la libre dispensación de medicamentos por parte del médico. Ni siquiera pretende ser un sistema experto que proponga un medicamento al médico, solo le advierte de posibles incompatibilidades y sirve de punto de referencia para su posterior toma de decisiones.

4.1 Roles

De cara a la seguridad de la aplicación, se ha planteado una dinámica de roles. Éstos vienen derivados de los permisos relacionados con cada uno de ellos.

Se ha evitado el uso de la aplicación por parte de usuarios sin identificar. Esta decisión tiene como objetivo el evitar crear una opinión contraria a la del recetante en usuarios sin una formación académica en el campo. No se pretende que un paciente ponga en duda las decisiones tomadas por su médico.

La creación de cuentas y asignación de roles, se reserva a decisión del administrador.

Existen cuatro categorías diferenciadas:

- **Invitado:** Este rol está dirigido a usuarios sin el respaldo de una institución médica, pero con unos conocimientos suficientes para el uso de la aplicación. El ejemplo más claro, un estudiante de medicina o farmacia que quiera comprobar sus hipótesis. Podrán rellenar los datos de un paciente anónimo así como el medicamento a recetar y obtener las incompatibilidades del caso dado.

- **Médico:** Tras la puesta en práctica del ”Escenario de integración 1” (1.1) con la aplicación perteneciente a una institución médica, además de las funcionalidades accesibles como ”Invitado”, este rol tiene como objetivo permitir al usuario el uso de dicha comunicación entre aplicaciones. Como se ha comentado anteriormente, dicho médico podrá introducir un dato identificativo del paciente para recuperar la información necesaria de la aplicación médica y evitar rellenarla manualmente, tras esto, seleccionará el medicamento deseado y obtendrá los mismos resultados que en el uso manual.

- **Administrador:** Este rol permite el acceso a todas las funcionalidades anteriores. La creación de cuentas y asginación de roles se reserva para este cargo. A parte, este rol será el encargado de gestionar la información de la base de datos (enfermedades, alergias, intolerancias, medicamentos, componentes químicos...) y mantener dicha información actualizada acorde con los avances médicos.

- **Usuario de aplicación externa:** Tras la puesta en práctica del ”Escenario de integración 2” (1.1), el uso de la lógica de negocio del sistema será accesible desde una aplicación ajena. Dichos usuarios se autenticarán y pasarán todos los procesos de seguridad en su propia aplicación.

4.2 Requisitos

A continuación se detallarán los requisitos demandados, expuestos en el *Product Backlog*. Cada uno de los requisitos estará acompañado de una *historia de usuario* para mejorar su comprensión. Debido a que las *historias de usuario* explican muy brevemente la funcionalidad deseada, en el apartado siguiente los requisitos se describen más en profundidad, acompañados de algún mockup para mejorar y facilitar la labor del desarrollador.

4.2.1 Product Backlog

Para simplificar la visualización de la tabla, se ha incluido un caso de uso ”Administrar” que engloba tres subcasos necesarios para su cumplimiento: añadir, borrar, modificar dicho elemento.

ID	Nombre	Historia de usuario
US01	Autenticación	Como usuario no autenticado quiero iniciar sesión en el sistema.
US02	Cerrar sesión	Como usuario autenticado quiero cerrar sesión en el sistema.
US03	Comprobación de incompatibilidades medicamento nuevo	Como invitado quiero comprobar las incompatibilidades del medicamento a recetar.
US04	Comprobación de incompatibilidades medicamentos recetados	Como invitado quiero comprobar las incompatibilidades de los medicamentos ya recetados.
US05	Repetir paciente	Como invitado quiero poder repetir la hipótesis con los mismos datos.
US06	Comprobación de incompatibilidades desde aplicación externa	Como usuario de aplicación externa quiero poder comprobar las incompatibilidades.
US07	Importar información a partir del DNI	Como médico quiero poder importar la información a partir del DNI.
US08	Administrar enfermedad	Como administrador quiero poder administrar una enfermedad.
US09	Listar enfermedades	Como administrador quiero ver la lista de enfermedades.
US10	Administrar alergia	Como administrador quiero poder administrar una alergia.
US11	Listar alergias	Como administrador quiero ver la lista de alergias.
US12	Administrar intolerancia	Como administrador quiero poder administrar una intolerancia.
US13	Listar intolerancias	Como administrador quiero ver la lista de intolerancias.
US14	Administrar componente químico	Como administrador quiero poder administrar un componente químico.

ID	Nombre	Historia de usuario
US15	Listar componentes químicos	Como administrador quiero ver la lista de componentes químicos.
US16	Administrar fórmulas	Como administrador quiero poder administrar una fórmula.
US17	Listar fórmulas	Como administrador quiero ver la lista de fórmulas.
US18	Administrar familia de componentes químicos	Como administrador quiero poder administrar una familia de componentes químicos.
US19	Listar familias de componentes químicos	Como administrador quiero ver la lista de familias.
US20	Administrar medicamento	Como administrador quiero poder administrar un medicamento.
US21	Listar medicamentos	Como administrador quiero ver la lista de medicamentos.
US23	Administrar usuario	Como administrador quiero poder administrar un usuario.
US24	Listar usuarios	Como administrador quiero ver la lista de usuarios.

Cuadro 4.1: Product Backlog del proyecto

4.2.2 Funcionaliddades

US01 - Autenticación

Para iniciar sesión en el sistema, el usuario debe introducir su nombre de usuario y su contraseña. Tras la verificación, si son correctos, el usuario quedará autenticado en el sistema. En caso contrario, se mostrará un mensaje de error. Tras un inicio de sesión correcto el usuario tendrá acceso a las funcionalidades ligadas a su rol.

US02 - Cerrar sesión

Un usuario autenticado con cualquier rol podrá cerrar sesión desde la opción ”Cerrar Sesión” del menú. Tras esto se le redirigirá a la página de inicio de sesión.

US03 - Comprobación de incompatibilidades

Un usuario autenticado con cualquiera de los roles administrador, médico o invitado podrá realizar la comprobación de una hipótesis dada. Para realizar esta operación, el sistema le proporcionará una serie de formularios para rellenar los siguientes datos, como se refleja en la figura 4.1:

- Edad.
- Sexo: Masculino o femenino.
- Lactancia: si se encuentra en estado de lactancia, esta opción solo se mostrará después de seleccionar el sexo femenino.
- Embarazo: si se encuentra en estado de embarazo, esta opción solo se mostrará después de seleccionar el sexo femenino.
- Filtrado glomerular: Estimación de la función renal.
- Hepatopatía: Indica si el paciente ha padecido o padece problemas de hígado. Este campo se diferencia del resto de enfermedades, debido a que tiene alta relevancia en la comprobación.
- Enfermedades: el usuario podrá buscar y seleccionar enfermedades de la lista de enfermedades.
- Alergias: el usuario podrá buscar y seleccionar alergias de la lista de alergias.
- Intolerancias: el usuario podrá buscar y seleccionar intolerancias de la lista de intolerancias.
- Medicamentos ya recetados: el usuario podrá buscar y seleccionar medicamentos de la lista de medicamentos ya recetados, el elemento elegido no puede estar seleccionado como medicamento a recetar, en dicho caso se desmarcará de la lista a medicamentos a recetar.
- Medicamento a recetar: el usuario podrá buscar y seleccionar solo un medicamento de la lista de medicamentos a recetar, el elemento elegido no puede estar seleccionado como medicamento ya recetado, en dicho caso se desmarcará de la lista medicamentos ya recetados.

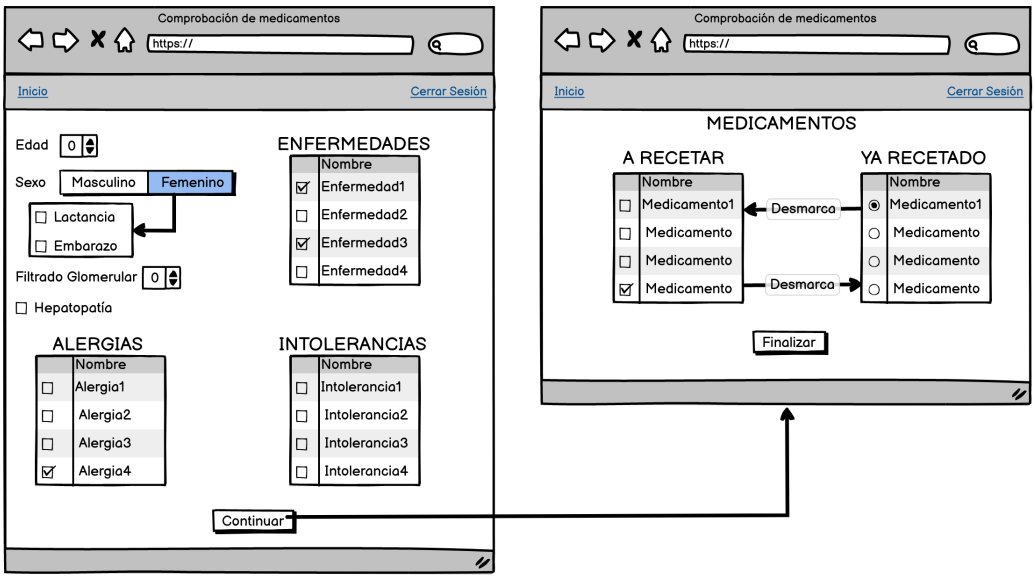

Figura 4.1: Mockup de comprobación de incompatibilidades

Al finalizar el proceso de inserción de datos, la aplicación cargaría la página de resultados mostrando la lista de medicamentos ya recetados y el medicamento a recetar, con las respectivas composiciones de cada medicamento. Así como la lista de alertas resultante de la comprobación.

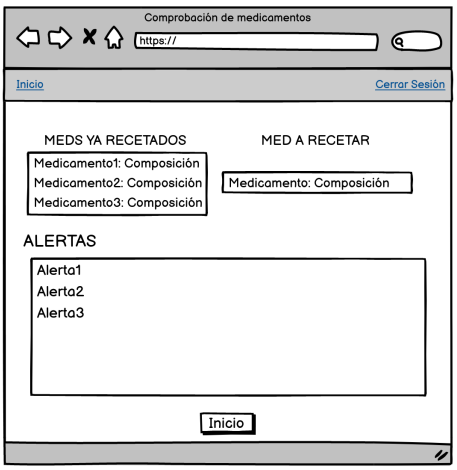

Figura 4.2: Mockup de resultados de la comprobación de incompatibilidades

US04 - Comprobación de incompatibilidades medicamentos recetados

Partiendo del caso anterior, se desea que a la lista de alertas se añadan los problemas que puedan tener los medicamentos ya recetados entre sí y con los datos del historial médico. Esta opción se da como optativa, ya que, el médico puede ser conocedor de estos problemas y aún así tomar la decisión de seguir con dichos tratamientos. Para evitar proporcionarle esta información repetidas veces y ralentizar la lectura de los resultados, se puede desactivar.

Se añadirá a la página de medicamentos una casilla de verificación, inicialmente desmarcada, que indique si se quiere realizar esta comprobación.

US05 - Repetir paciente Una vez terminada una comprobación es posible que se quiera repetir la hipótesis alterando de manera poco significativa los datos introducidos. Para ello se aporta esta opción, que una vez visualizados los resultados, vuelva al comienzo del caso de uso de comprobación de incompatibilidades rellenando automáticamente todos los campos y permitiendo alterarlos si fuese necesario.

Se añadirá un botón en la página de resultados que permita realizar esta operación.

US06 - Comprobación de incompatibilidades desde aplicación externa

Situados en el escenario de integración 2. Un usuario de la aplicación sanitaria externa podrá desde su interfaz gráfica obtener los datos del paciente deseado y seleccionar el medicamento que quiere comprobar. Tras esto, se consultará al servicio web proporcionado por la aplicación a desarrollar, enviándole dichos datos. Para, finalmente, obtener un JSON con las alertas, que la aplicación sanitaria externa traducirá y mostrará vía su interfaz al médico.

US07 - Importar información a partir del DNI

Con el escenario de integración 1 aplicado. Un usuario autenticado con el rol médico tendrá la posibilidad de rellenar un dato identificativo, en este caso DNI, para importar la información de un servicio externo. Evitando así tener que rellenarla, aún así despues de la importación se le permitirá cambiar los datos que considere pertinentes. Dicha funcionalidad está remarcada en la figura 4.3.

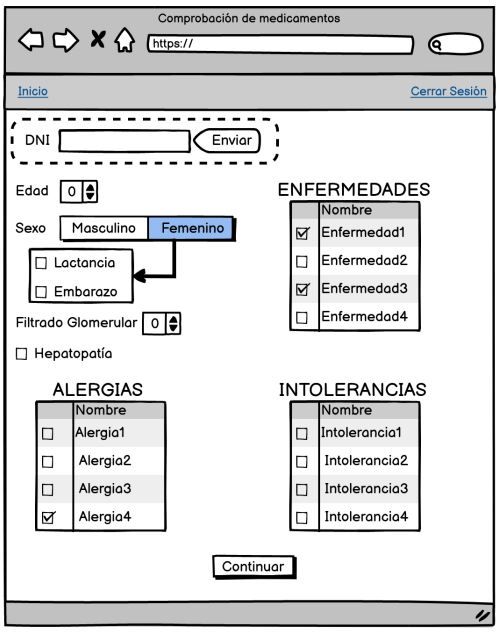

Figura 4.3: Mockup de comprobación de incompatibilidades con importación del DNI

US09, US11, US13, US15, US17, US19, US21, US23 - Listar elementos Estando autenticado como administrador. Estos casos recogen la posibilidad de obtener la lista de enfermedades, alergias, intolerancias, componentes químicos, fórmulas, familias de componentes químicos, medicamentos y usuarios. Mostrándolos en una tabla que permita la búsqueda de un elemento específico.

US08, US10, US12, US18 - Casos administrativos básicos

Estando autenticado como administrador. Estos casos recogen la administración de enfermedades, alergias, intolerancias y familias de componentes químicos. Se agrupan ya que solo necesitan un dato, el nombre que identifica al elemento. Estos casos administrativos deben tener la posibilidad de crear, borrar y modificar un dicho elemento.

US14 - Administrar componente químico

Es importante hacer especial énfasis en la administración de los componentes químicos, ya que componen la parte primordial del esqueleto en el esquema de entidades.

Para la creación o modificación de un componente químico se pedirá nombre, familia a la que pertenece, restricciones genéricas que pueda tener el componente (edad, sexo, necesidad de ajuste de dósis según el filtrado glomerular...) y sus listas de incompatibilidades con enfermedades, alergias, intolerancias u otros componentes químicos. Reflejados en la figura 4.4.

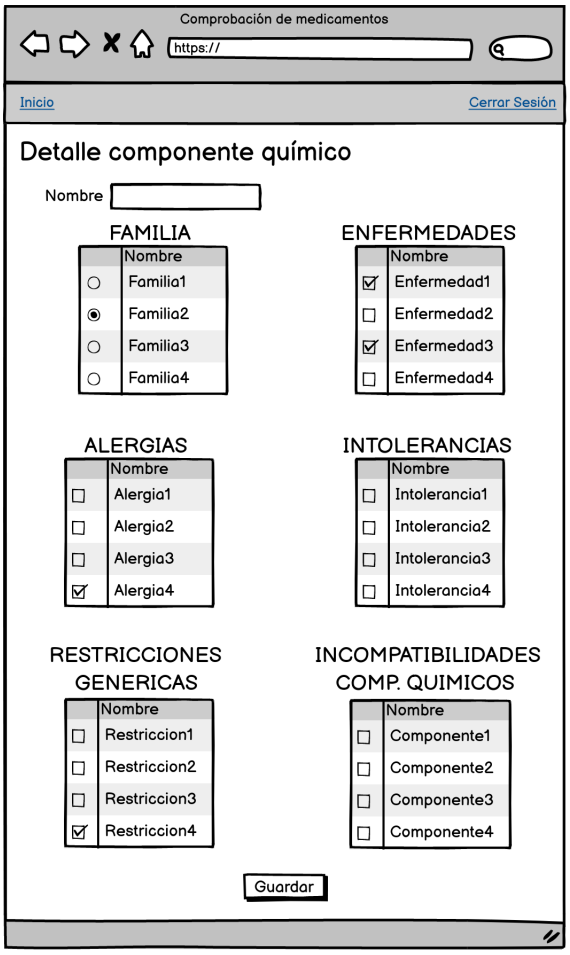

Figura 4.4: Mockup de administración de componente químico

US16 - Administrar fórmula

Una fórmula es la conjunción de componentes químicos que define la composición química de un medicamento. Por tanto, a la hora de añadir o modificar una fórmula debemos aportar un nombre y la lista de componentes químicos que la componen. Permitiendo eliminar todos sus medicamentos derivados al eliminarla.

US20 - Administrar medicamento

Un medicamento, es el producto que se pretende comercializar una compañía, basando su fabricación en la fórmula de la que deriva.

A la hora de añadir o modificar debemos aportar un nombre y seleccionar la fórmula de la que deriva. Esta administración también debe permitir el borrado.

US22 - Administrar usuario Un administrador tiene la posibilidad de añadir, eliminar y modificar cualquier cuenta del sistema. Para añadir o modificar debe aportar la siguiente información:

- Nombre de cuenta
- Email
- Contaseña
- Rol

Capítulo 5

Planificación

En este capítulo se detallarán cada uno de los *sprints* planteados para el proyecto, aplicando la metodología iterativa mencionada en el capítulo 3.

Tras esto, se proporcionará una visión global de la duración y coste que acarreará este proyecto.

5.1 Sprints

A continuación se detallarán los distintos *sprints* planteados durante el proyecto. Siguiendo la filosofía de la metodología comentada en el capítulo ya mencionado, cada uno de ellos fue planificado y estimado en la reunión inicial, *Sprint Planning*.

Como se verá reflejado en las estimaciones, junto con el creciente conocimiento del proyecto aumentó la precisión de las estimaciones para dichos *sprints*.

5.1.1 Sprint 0 Análisis del ámbito de la aplicación

El campo en el que se desarrolla la aplicación no es trivial ni de conocimiento general. Por este motivo, previo al comienzo del desarrollo, se planteó un *sprint* de formación referente a la composición de los medicamentos y las incompatibilidades de los componentes químicos

que los forman.

En este *sprint* se desarrolló el diseño de incompatibilidades mencionado en el capítulo 2.

5.1.2 Sprint 1 - Construcción base del proyecto

El objetivo de este incremento consiste en la creación de un entorno de desarrollo y el diseño del arquetipo para poner en funcionamiento las distintas tecnologías propuestas. Al finalizar, se obtiene una estructura base del proyecto, a la cual se irán añadiendo las distintas funcionalidades.

Así como el diseño del diagrama de entidades del proyecto y su implementación en la base de datos.

5.1.3 Sprint 2 - Seguridad del sistema (gestión de roles)

Previo al desarrollo de funcionalidades que necesitasen ser securizadas y clasificadas por roles, se planteó un esquema claro de estas asignaciones. Para ello, a parte de este diseño de la dinámica de roles, se abordaron las siguientes *historias de usuario*:

- US01 - Autenticación
- US02 - Cerrar sesión

5.1.4 Sprint 3 - Gestión de enfermedades, alergias e incompatibilidades

A partir de este incremento, se comenzó con la gestión de la información de la base de datos. Se planteó una filosofía bottom-up siguiendo el diagrama de entidades, por ello se optó por empezar con la gestión de enfermedades, alergias e intolerancias.

Debido a que las restricciones genéricas, a priori, no se desean modificar, se ha obviado su gestión ya que permanecerán intangibles en la base de datos.

Para conseguir este objetivo se han implementado las siguientes *historias de usuario*:

- US08 - Administrar enfermedad
- US09 - Listar enfermedades
- US10 - Administrar alergia
- US11 - Listar alergias
- US12 - Administrar intolerancia
- US13 - Listar intolerancias

5.1.5 Sprint 4 - Gestión de componentes químicos y familias

En este incremento, siguiendo la filosofía *bottom-up*, se planteó la administración de las familias de componentes químicos y posteriormente de los propios componentes. Para ello, se han implementado las siguientes *historias de usuario*:

- US14 - Administrar componente químico
- US15 - Listar componentes químicos
- US18 - Administrar familia de componentes químicos
- US19 - Listar familias de componentes químicos

5.1.6 Sprint 5 - Gestión de medicamentos y fórmulas

Como últimos dos eslabones del diagrama de entidades contenidos en la base de datos, se encuentran fórmula y, por encima, medicamento. Para abordar la administración de ambas entidades se han implementado las siguientes *historias de usuario*:

- US16 - Administrar fórmulas
- US17 - Listar fórmulas

- US20 - Administrar medicamento
- US21 - Listar medicamentos

5.1.7 Sprint 6 - Comprobación de incompatibilidades

En este incremento, se planteó desarrollar el núcleo de la aplicación, conformado por el análisis de incompatibilidades de los componentes químicos y el diseño de las alertas relacionadas. Para ello, se abordaron las siguientes *historias de usuario*:

- US03 - Comprobación de incompatibilidades medicamento nuevo
- US04 - Comprobación de incompatibilidades medicamentos recetados
- US05 - Repetir paciente

5.1.8 Sprint 7 - Integración con sistemas externos

El objetivo de este *sprint* fue el análisis de necesidades para la integración con sistemas de salud ya existentes. Tras este estudio, se abordaron los dos escenarios comentados anteriormente. Para llevarlos acabo, se implementaron las siguientes *historias de usuario*:

- US06 - Comprobación de incompatibilidades desde aplicación externa
- US07 - Importar información a partir del DNI

5.1.9 Sprint 8 - Gestión de usuarios

Como último *sprint* del desarrollo de la aplicación, se discutió el método de registro de usuarios en la aplicación. En este incremento surgió la discusión referente a no permitir a un usuario casual acceder a la aplicación. Esta idea derivó del posible conflicto de opiniones que generaría aportar esta información a un usuario, sin los conocimientos suficientes en dicho campo, y su médico.

Por lo cual, se decidió dejar esta funcionalidad en manos del administrador, dándole la potestad de decidir a que usuarios darle la posibilidad de acceder a la aplicación. La idea reside en que tanto entidades médicas como formativas que deseen poseer cuentas en dicha aplicación, se pongan en contacto con el administrador para solicitarlas.

Durante este incremento se han abordado las siguientes *historias de usuario*:

- US23 - Administrar usuario
- US24 - Listar usuarios

5.1.10 Sprint 9 - Elaboración de la memoria

El último incremento del proyecto se centra en la elaboración de esta memoria. También se aprovecha para implementar distintas mejoras estéticas en la aplicación.

5.2 Visión global

Como se plantea en la metodología utilizada, la planificación temporal del proyecto se descompone en la suma de las planificaciones temporales de cada incremento. Estos datos resultan de la estimación de la duración de los casos de uso abordados en cada incremento durante cada *Sprint Planning*.

Se podrá observar en los resultados como las estimaciones fueron mejorando a lo largo del proyecto, ya que en las etapas iniciales, debido al desconocimiento del campo a desarrollar, fue imposible realizar una estimación correcta.

Por otro lado, existieron dificultades tanto laborales como personales que desembocaron en interrupciones en el desarrollo del proyecto. Esto se verá reflejado en las fechas de planificación de los *sprints*

5.2.1 Duración

En la siguiente tabla se observan los segmentos temporales en los que se abordó cada incremento, así como la duración, en horas, que consumió el desarrollo del objetivo del *sprint*.

Como se puede observar en los primeros incrementos las estimaciones no eran correctas, esto deriva del desconocimiento del campo y de la propia estructura del software a desarrollar. Al finalizar el *sprint* "4 - Gestión de componentes químicos y familias", se mejoró notablemente la estimación. Este resultado viene dado del incremento del conocimiento base del proyecto, así como de la mejora continua e introspección derivada de la metodología utilizada. Cabe destacar que en los *sprints* 6 y 9 sigue existiendo un desfase notable, esto se debe a que el objetivo del incremento era más amplio y profundo, esto derivó en una peor estimación.

Sprint	Objetivo	Fecha Inicio	Fecha Fin	Horas estimadas	Horas reales
0	Análisis del ámbito de la aplicación	02/09/19	01/10/19	80	120
1	Construcción base del proyecto	03/10/19	16/10/19	50	80
2	Seguridad del sistema (gestión de roles)	17/02/20	09/03/20	48	32
3	Gestión de enfermedades, alergias e incompatibilidades	30/03/20	23/04/20	40	60
4	Gestión de componentes químicos y familias	04/05/20	21/06/20	80	96
5	Gestión de medicamentos y fórmulas	22/06/20	30/07/20	64	68
6	Comprobación de incompatibilidades	07/09/20	26/02/21	120	130
7	Integración con sistemas externos	01/03/21	11/04/21	80	80
8	Gestión de usuarios	12/04/21	23/04/21	40	42
9	Elaboración de la memoria	26/04/21	17/06/21	100	120
	TOTAL	02/09/19	17/06/21	702	828

Figura 5.1: Planificación temporal estimada y real

5.2.2 Coste

Suponiendo que esta aplicación fuese desarrollada por una empresa real. El coste de la aplicación se calcula multiplicando las horas totales del proyecto por un coste medio/hora. Este coste medio incluye tanto el sueldo de los trabajadores como otros costes relacionados con el proyecto (alquiler del lugar de trabajo, coste de la electricidad...). El coste medio se ha determinado en 35€/hora.

Por lo tanto, el coste aproximado del proyecto sería:

Coste

Se define la fórmula del coste como:

$$\textbf{Coste Total(€)} = HorasRealesTotales(h) * CosteMedio/Hora(€/h)$$

$$\textbf{Coste Total} = 828h * 35€/h = \textbf{28.980€}$$

Capítulo 6

Fundamentos Tecnológicos

En este apartado se expondrán las tecnologías y herramientas empleadas durante el desarrollo del proyecto. Algunas de ellas seleccionadas previamente al inicio del desarrollo y otras al surgir su necesidad específica.

Para su fácil comprensión y análisis se presentarán divididas en los siguientes bloques:

- Empleadas en el *backend*
- Empleadas en el *frontend*
- De soporte o complementarias para el proceso de desarrollo

6.1 Tecnologías empleadas en el *backend*

El *backend* de la aplicación consiste en una lógica de negocio que engloba las funcionalidades necesarias para conseguir los objetivos mencionados anteriormente, tanto el análisis de incompatibilidades de los medicamentos como las necesidades de gestión de la base de datos.

6.1.1 MySQL

MySQL [5] es un sistema de gestión de bases de datos relacional desarrollada por *Oracle*. Este sistema incluye las siguientes características que lo hacen interesante para el proyecto:

- Amplio soporte de sentencias en lenguaje SQL.
- Transacciones, que nos permiten realizar un conjunto de actualizaciones, inserciones y borrados de forma consistente.
- Índices, los cuales nos permiten realizar consultas de forma más rápida y eficaz.
- Posibilidad de selección de mecanismos de almacenamiento.

El mecanismo de almacenamiento seleccionado es *InnoDB* [6] debido a que da el soporte necesario para cumplir con las ventajas mencionadas anteriormente.

La razón principal por la que se optó por este sistema de base de datos y dicho mecanismo de almacenamiento, reside en la experiencia previa adquirida durante la formación académica.

6.1.2 JPA

JPA (Java Persistence API) [7] es un framework de persistencia del lenguaje de programación Java, que maneja datos relacionales de las aplicaciones. Para su implementación en este proyecto se ha escogido *hibernate*.

Hibernate

Hibernate [8] es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones.

6.1.3 Java

Java es el lenguaje de programación más famoso en la actualidad. Entre sus principales características están que es orientado a objetos, multiplataforma y con un *sistema de recolección de basura* más sofisticado que sus predecesores.

La razón principal de la elección de este lenguaje, reside en el amplio conocimiento del mismo por parte del alumno. Así como en el uso de diferentes *frameworks* de *Spring*, que se apoyan en este lenguaje.

6.1.4 Apache Maven

Apache Maven [9] es una herramienta destinada a la gestión y configuración del software en proyectos *Java*. El *Project Object Model (POM)* es la base del funcionamiento de dicha herramienta, se basa en un fichero *XML* en el que se describen las dependencias y el método de construcción de un proyecto software. Sus principales funcionalidades son:

- Automatiza la compilación y empaquetado del software.
- Simplifica y centraliza en un mismo archivo, el *POM*, la gestión de dependencias.
- Permite definir configuraciones alternativas, facilitando la implantación del proyecto en diferentes entornos.
- Implementación orientada a *plugins*, por lo que es fácilmente integrable con otras herramientas.

6.1.5 Spring Boot

Spring Boot [10] es uno de los proyectos más relevantes y famosos de *Spring Framework*. Sus principales objetivos son:

- Permitir la creación de aplicaciones *stand-alone*.

- Permitir la integración en *Tomcat*, *Jetty* o *Undertow* sin la necesidad de la creación de archivos *WAR*.
- Proporcionar *Spring Boot starters* para simplificar la configuración del *POM*.

Uno de los *Spring Boot starters* más relevantes usados en la aplicación es *Spring Data JPA* [11]. Este framework pertenece a la familia *Spring Data* y facilita la implementación de repositorios basados en *JPA*.

6.1.6 Lombok

Lombok [12] es una librería que facilita enormemente el desarrollo básico de entidades en *Java*. Su función es simplificar la creación de entidades, automatizando la implementación de los métodos básicos de cualquier entidad (*getters*, *setters*, constructores vacíos o *equals*).

6.1.7 JUnit

JUnit [13] es un *framework* de pruebas en *Java*. Permite realizar comprobaciones en entornos controlados, verificando el resultado esperado de dicha ejecución.

6.1.8 Rest

REpresentational State Transfer (REST) es un tipo de arquitectura con el objetivo de construir aplicaciones web. Normalmente los servicios REST presentan una serie de recursos, transmitidos mediante *JSON* o *XML*, y una serie de operaciones para gestionarlos.

6.2 Tecnologías empleadas en el *frontend*

Para el desarrollo del *frontend* se utilizaron las siguientes:

6.2.1 jQuery

jQuery [14] es una biblioteca multiplataforma de *JavaScript* que permite simplificar la manera de interactuar con los documentos *HTML*, manipular el árbol *Document Object Model (DOM)*, manejar eventos, desarrollar animaciones y agregar interacciones basadas en *AJAX*. El *plugin* de esta biblioteca, más usado en la aplicación es *DataTable*, esta herramienta permite el encapsulamiento y gestión de listas de datos en tablas.

6.2.2 Thymeleaf

Thymeleaf [15] es un sistema de plantillas *Java* que permite generar cómodamente *HTML* en el lado servidor. Dicha herramienta contiene módulos de integración con *Spring Framework*.

6.3 Tecnologías complementarias para el proceso de desarrollo

6.3.1 Eclipse

Eclipse [16] es una plataforma de software compuesto por un conjunto de código abierto. El *Entorno de Desarrollo Integrado (IDE)* más usado es el adaptado a *Java*. Las dos características que lo convierten en uno de los entornos de desarrollo más usados son:

- Ser un entorno multiplataforma.
- Su capacidad ilimitada de inserción de funcionalidades por medio de *plugins*

6.3.2 Git

Git [17] es un sistema de control de versiones (SCM) distribuido. En este sistema cada uno de los usuarios mantiene una copia en local del historial de versiones. Gracias a la fácil gestión de ramas, ofrece al usuario la posibilidad de desarrollo paralelo o de desarrollo no lineal de un

producto. También ofrece sencillez a la hora de gestionar los conflictos entre archivos, por lo que es recomendable en proyectos de gran tamaño.

6.3.3 Github

Github [18] es una plataforma de alojamiento de proyectos desarrollados colaborativamente basada en *Git*. Permite la creación de repositorios públicos y privados de manera gratuita, fomentando el desarrollo de software libre.

6.3.4 Redmine

Redmine [19] es una herramienta administrativa para la gestión de proyectos. Tiene muchas funcionalidades diferentes entre las que destacan:

- Planificar, organizar y realizar el seguimiento de las distintas tareas.
- Realizar el seguimiento de los distintos usuarios en el proyecto.
- La creación de una *wiki* con estándares sobre dicho proyecto.
- El uso de foros para la comunicación entre los distintos integrantes del proyecto, a nivel de tarea y global.

En el proyecto, la característica más utilizada fue la que permite la planificación y posterior gestión y seguimiento de las distintas tareas. Esto derivó en una constante representación detallada del estado del proyecto, así como en su fácil análisis una vez terminado.

Capítulo 7

Desarrollo

En este capítulo se trataran los aspectos a destacar en el desarrollo de la aplicación. Se detallarán tanto aspectos técnicos como funcionales de los distintos incrementos.

7.1 Modelo de datos

En la figura 7.1 podemos visualizar las entidades usadas por el sistema.

No se han incluido los atributos de las entidades para evitar complicar la comprensión del diagrama.

Para la explicación seguiremos una estrategia *top-down*, se partirá de la información que debe aportar el usuario a las entidades internas que posee la aplicación para realizar las comprobaciones de incompatibilidades.

Se comenzarán explicando las entidades no persistidas en base de datos y que tienen como objetivo encapsular la información aportada por el usuario. Estas entidades son **historial médico** y *tratamiento*.

Un **tratamiento** es la representación del recetado de un **medicamento** al paciente. Por lo cual, el caso que se propondrá para la comprobación de incompatibilidades consistirá en un **tratamiento** y el **historial médico** con el que cotejarlo.

El **historial médico** recoge toda la información relativa al paciente:

- Edad
- Sexo
- Estado de embarazo
- Estado de lactancia
- Filtrado Glomerular
- Hepatopatía
- Lista de **enfermedades** del paciente
- Lista de **alergias** del paciente
- Lista de **intolerancias** del paciente
- Lista de **tratamientos** en vigor del paciente

Continuando con la estrategia *top-down*. Un **medicamento**, ya sea comercial o genérico, contiene una **fórmula**, mientras que varios **medicamentos** diferentes pueden presentar la misma. Dicha **fórmula** está formada por uno o más **componentes químicos**.

El **componente químico** es la unidad fundamental de información del sistema de incompatibilidades. Los **componentes químicos** están categorizados en **familias** según sus propiedades terapéuticas. Para cada componente, se recoge la lista de **restricciones genéricas**, **enfermedades**, **alergias** e **intolerancias** con las que presenta incompatibilidades. Así como la lista de otros **componentes químicos** con los que puede tener interacciones farmacológicas.

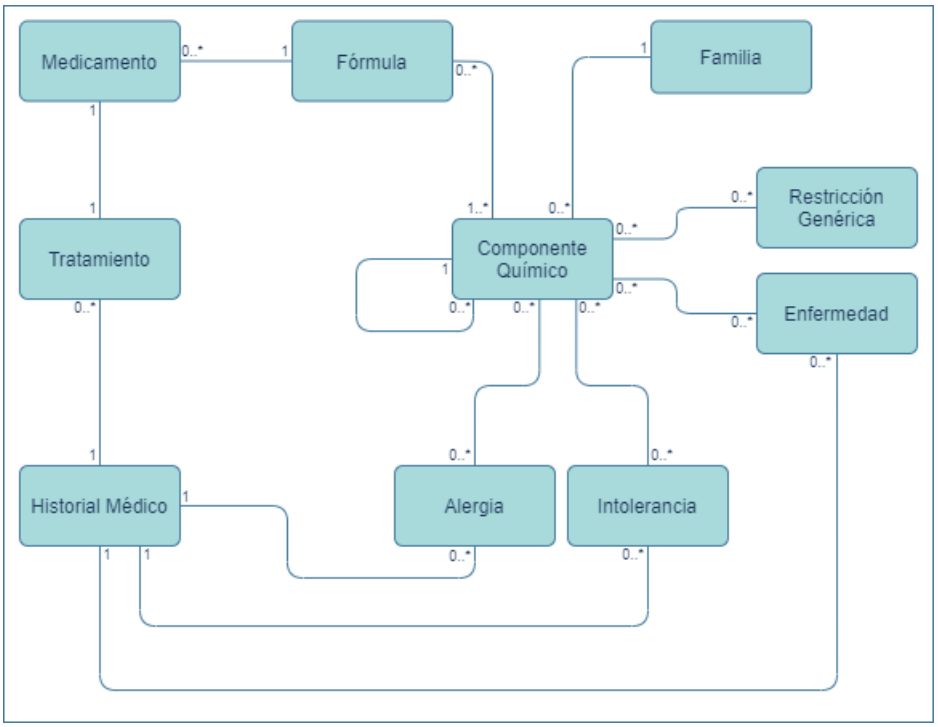

Figura 7.1: Diagrama de entidades de la aplicación

7.2 Sprint 0: Análisis del ámbito de la aplicación

En el sprint inicial se realizaron una serie de reuniones informales con un médico. En ellas se planteó la viabilidad y posibles escenarios de empleo de la aplicación, agudizando el planteamiento de requisitos y escenarios inicial.

La idea inicial se basaba en automatizar el proceso de comprobación de incompatibilidades, que debe realizar manualmente un médico, a la hora de recetar un nuevo medicamento. En estas reuniones, se pulieron los distintos casos en los que un medicamento podía presentar inconvenientes. Se obtuvieron las mencionadas en el apartado ”2.2 Categorías de incompatibilidades”.

Analizando algunas aplicaciones de salud sobre fármacos, se decidió añadir una comprobación adicional, basada en las familias a las que pertenecen los componentes químicos. Ya que dos componentes con la misma finalidad terapéutica pueden resultar contraproducentes. Esta comprobación resulta de ayuda y genera información complementaria en el análisis de

incompatibilidades.

Tras todo esto se conformó un esquema que detallaba los pasos para la generación de alertas sobre las incompatibilidades, como se observa en la figura 7.2.

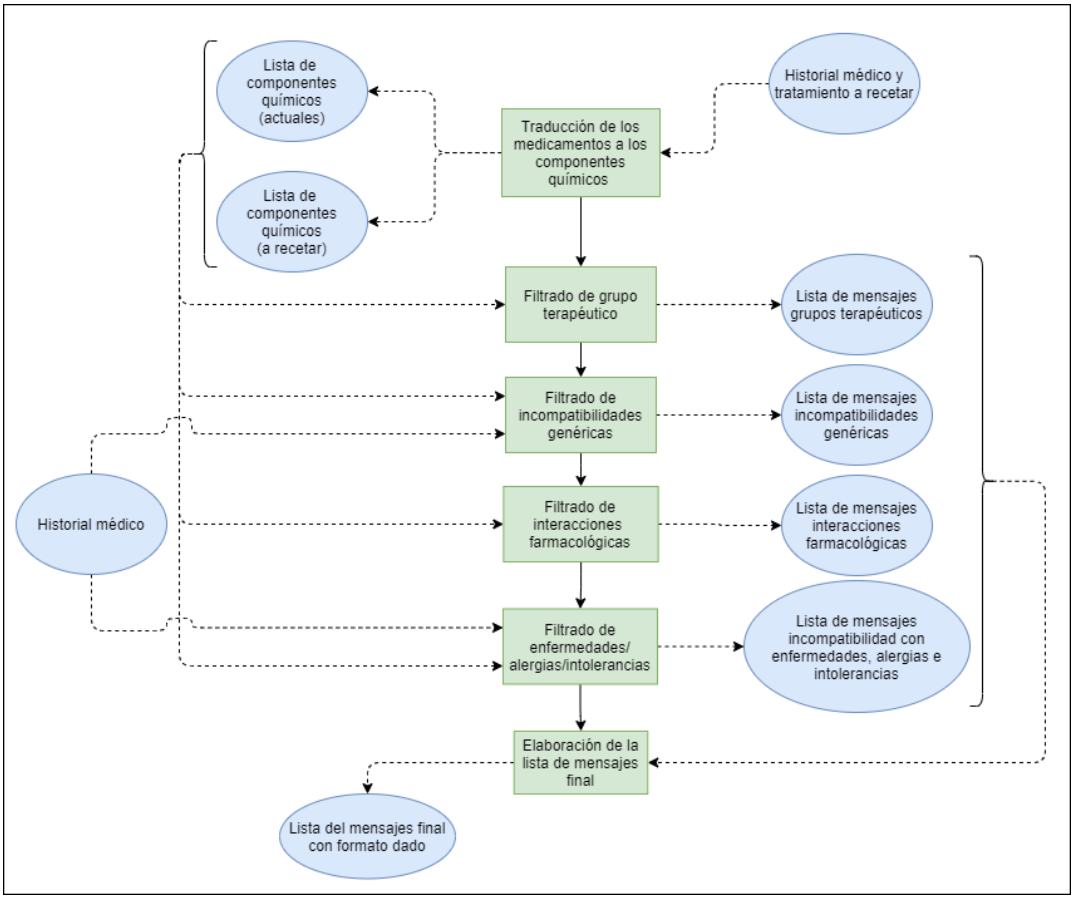

Figura 7.2: Flujo de filtrados

Destacar que en este incremento solo se trataba de definir el núcleo de la aplicación y no sus posibles usuarios ni escenarios de uso.

7.3 Sprint 1: Construcción base del proyecto

Comenzando el desarrollo de la aplicación, se planteó la configuración de los entornos de desarrollo, así como una pequeña funcionalidad de prueba con la que asegurar el funciona-

miento de todas las tecnologías integradas, obteniendo así un escenario base a partir del cual desarrollar los siguientes incrementos.

Para la creación inicial del proyecto, se utilizó la herramienta *Spring Initialzr* [20], de la plataforma *Spring*, con la que se obtuvo un arquetipo Maven inicial. A este arquetipo se le añadieron las dependencias necesarias para el uso de las herramientas y tecnologías mencionadas.

7.3.1 Estructura del *Backend*

Al ser implementado como un proyecto *Maven*, sigue la estructura de directorios por defecto de la herramienta:

- **src/main/java:** Contiene la estructura de paquetes del código fuente de la aplicación. En su directorio raíz se encuentra la clase Application.java con la notación necesaria de *Spring Boot (@SpringBootApplication)*. Siguiendo una estructuración MVC (Model View Controller) se crearon las siguientes carpetas:
  - **Model:** Contiene todas las entidades necesarias para el funcionamiento de la aplicación, categorizadas en directorios según su uso.
  - **Service:** Contiene los distintos servicios con su lógica de negocio necesaria. En este directorio se encuentran todos los servicios de gestión de entidades la base de datos, así como el servicio médico para el procesado de incompatibilidades.
  - **Controller:** Contiene los controladores para los distintos servicios. Así como un subdirectorio con los *DTOs* necesarios para el intercambio de información con el *frontend*.
- **src/main/resources:** Contiene el archivo application.properties, donde se configuran distintos aspectos de la aplicación (e.g. conexión a la base de datos).
- **src/test/java:** Contiene las clases donde se implementan las distintas pruebas de cada servicio.

Para la creación de la base de datos se usó el gestor *MySQL*. Se crearon dos bases de datos idénticas una para el entorno de pruebas y otra para la propia aplicación.

7.3.2 Estructura del *Frontend*

La lógica de las peticiones se desarrolla en los controladores anteriormente mencionados.

Para el desarrollo de las vistas del *Frontend* se crean tres subdirectorios partiendo de **src/-main/resources**:

- **/static/css:** Contiene todos los archivos css.
- **/static/images:** Contiene todas las imágenes utilizadas en el proyecto.
- **/templates:** Contiene los archivos *.html* categorizados según su función (Administración o Médica).

7.4 Sprint 2 - Seguridad del sistema (gestión de roles)

7.4.1 Análisis

Posterior al diseño y elaboración de la base del proyecto, en este incremento comenzaron a desarrollar las historias de usuario del *Product Backlog*. Debido a que en el diseño inicial del *Product Backlog* ya se habían priorizado, se comenzó con la seguridad del sistema. Cada una de las historias de usuario se muestran descompuestas en tareas a continuación:

US01 - Autenticación
Como usuario no autenticado quiero iniciar sesión en el sistema.
Tareas
Creación de la tabla Usuario en base de datos y de la entidad asociada.
Creación del servicio de usuario utilizado por *Spring Security*.
Implementación de la configuración de *Spring Security*.

Cuadro 7.1: Sprint 2: Autenticación

US02 - Cerrar sesión
Como usuario autenticado quiero cerrar sesión en el sistema.
Tareas
Implementación de la configuración de *Spring Security*.
Diseño de un menú con el botón ”Cerrar sesión”.

Cuadro 7.2: Sprint 2: Cerrar sesión

7.4.2 Diseño e implementación

US01 - Autenticación

Para la implementación de la autenticación se optó por usar el token estándar de *Spring Security*, ya que solo serviría como prueba de concepto. Con esto el *backend* no necesita mantener el registro de los tokens, éstos se mantienen en el lado cliente y son enviados junto con cada petición realizada.

Para el desarrollo de esta historia de usuario, tras la creación de la tabla Usuario en la base de datos y su entidad asociada así como el repositorio que las relaciona, se implementó el servicio de usuarios con la función esperada por *Spring Security (loadUserByUsername)*. Adicionalmente se configuró una clase que extendiese *WebConfigurerAdapter*.

US02 - Cerrar sesión

El cierre de sesión se completó derivado de la implementación de *Spring Security*, mencionada en la historia de usuario anterior. Para la visualización del cerrado de sesión se implementó una barra de navegación, con enlace al inicio de la aplicación así como el cerrado de sesión.

7.5 Sprint 3 - Gestión de enfermedades, alergias e incompatibilidades

7.5.1 Análisis

En este incremento, se replanteó el orden del *Product backlog*, siguiendo una metodología *bottom-up*, se decidió implementar las entidades enfermedad, alergia e intolerancia. Así como las funcionalidades necesarias para la gestión de dichas entidades.

US08 - Administrar enfermedad
Como administrador quiero poder administrar una enfermedad.
Tareas
Creación del servicio de enfermedades.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de enfermedad.

Cuadro 7.3: Sprint 3: Administrar enfermedad

US09 - Listar enfermedades
Como administrador quiero ver la lista de enfermedades.
Tareas
Creación del endpoint que devuelva la lista de enfermedades.
Creación de la vista que contiene la lista de enfermedades.

Cuadro 7.4: Sprint 3: Listar enfermedades

US10 - Administrar alergia
Como administrador quiero poder administrar una alergia.
Tareas
Creación de la tabla Alergia en base de datos y de la entidad asociada.
Creación del servicio de alergias.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de alergia.

Cuadro 7.5: Sprint 3: Administrar alergia

US11 - Listar alergias
Como administrador quiero ver la lista de alergias.
Tareas
Creación del endpoint que devuelva la lista de alergias.
Creación de la vista que contiene la lista de alergias.

Cuadro 7.6: Sprint 3: Listar alergias

US12 - Administrar intolerancia
Como administrador quiero poder administrar una intolerancia.
Tareas
Creación de la tabla Intolerancia en base de datos y de la entidad asociada.
Creación del servicio de intolerancias.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de intolerancia.

Cuadro 7.7: Sprint 3: Administrar intolerancia

US13 - Listar intolerancias
Como administrador quiero ver la lista de intolerancias.
Tareas
Creación del endpoint que devuelva la lista de intolerancias.
Creación de la vista que contiene la lista de intolerancias.

Cuadro 7.8: Sprint 3: Listar intolerancias

7.5.2 Diseño e implementación

Ya que el proceso realizado para llevar a cabo las historias de usuario de administración y listado ha sido el mismo para las tres entidades (enfermedad, alergia e intolerancia). Para evitar la repetición de la misma información, se presentarán en dos grupos clasificados por tarea en los que se explican la implementación de dichas historias de usuario.

US08 - Administrar enfermedad, US10 - Administrar alergia y US12 - Administrar intolerancia

Para la administración de dichas entidades, posterior a la implementación de las tablas de base de datos y de sus respectivas entidades así como los repositorios, se desarrolló un servicio común para ellas debido a su gran parecido y utilidad en la aplicación. En este servicio se incluyeron las operaciones necesarias para la recuperación, el guardado y el borrado de dichas entidades.

Con toda la lógica de negocio para realizar la gestión de dichas entidades, se desarrollaron controladores independientes en los que se incluyeron los endpoints para dar cobertura a dichos métodos de los servicios.

Una vez terminados los controladores, se crearon vistas que reflejasen la información de dichas entidades y las opciones necesarias para su gestión.

A continuación, en la figura 7.3, podemos observar las distintas clases encargadas de la administración y listado de dichas entidades.

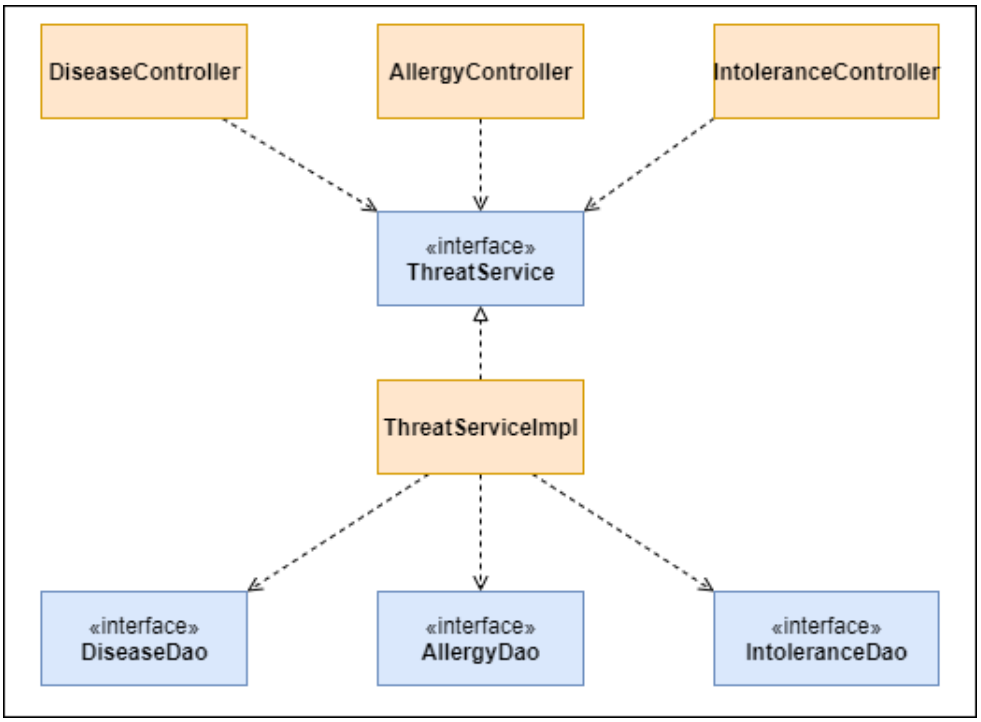

Figura 7.3: Diagrama de clases encargadas de la administración y listado de enfermedades, alergias e incompatibilidades

US09 - Listar enfermedades, US11 - Listar alergias y US13 - Listar intolerancias

Para la visualización de dichas listas visibles en la figura 7.4, se añadieron métodos que devolvían la lista de la entidad, a cada uno de los servicios creados en las historias de usuario anteriores.

Tras esto, se añadieron a cada controlador los endpoints necesarios que devolvían dichas listas. Una vez finalizados, se crearon las vistas que recogían en tablas la información de dichas listas. Adicionalmente, se añadió al menú un desplegable que contiene el acceso a dicha gestión, este desplegable solo se muestra para los usuarios "Administrador".

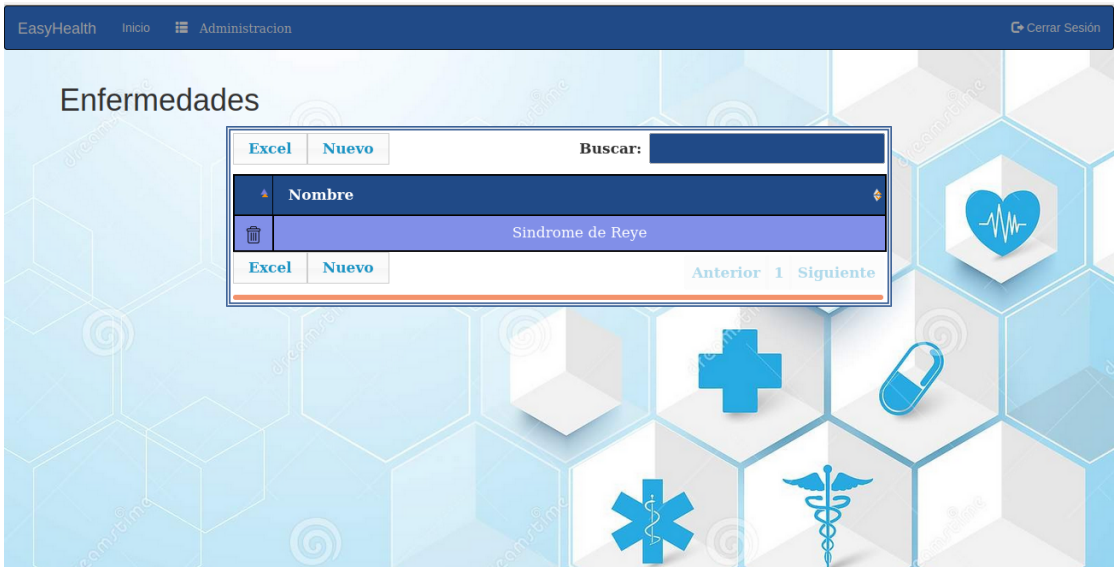

Figura 7.4: Vista de la lista de enfermedades

7.6 Sprint 4 - Gestión de componentes químicos y familias

7.6.1 Análisis

Prosiguiendo con la metodología *bottom-up*, se decidió realizar la gestión de los componentes químicos y sus familias. Para ello se implementaron las siguientes historias de usuario:

US14 - Administrar componente químico
Como administrador quiero poder administrar un componente químico.
Tareas
Creación de la tabla Componente químico en base de datos y de la entidad asociada.
Creación del servicio de componentes químico.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de componente químico.

Cuadro 7.9: Sprint 4: Administrar componente químico

US15 - Listar componentes químicos
Como administrador quiero ver la lista de componentes químicos.
Tareas
Creación del endpoint que devuelva la lista de componentes químicos.
Creación de la vista que contiene la lista de componentes químicos.

Cuadro 7.10: Sprint 4: Listar componentes químicos

US18 - Administrar familia de componentes químicos
Como administrador quiero poder administrar una familia de componentes químicos.
Tareas
Creación de la tabla Familia en base de datos y de la entidad asociada.
Creación del servicio de familias.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de familia.

Cuadro 7.11: Sprint 4: Administrar familia de componentes químicos

US19 - Listar familias de componentes químicos
Como administrador quiero ver la lista de familias.
Tareas
Creación del endpoint que devuelva la lista de familias.
Creación de la vista que contiene la lista de familias.

Cuadro 7.12: Sprint 4: Listar familias de componentes químicos

7.6.2 Diseño e implementación

La implementación de los casos de uso de este incremento se realizó de manera equivalente a la descrita en el *sprint* 3 (7.5.2) para enfermedades, alergias e intolerancias.

A continuación se incluye el diagrama usado en dichas historias de usuario 7.5:

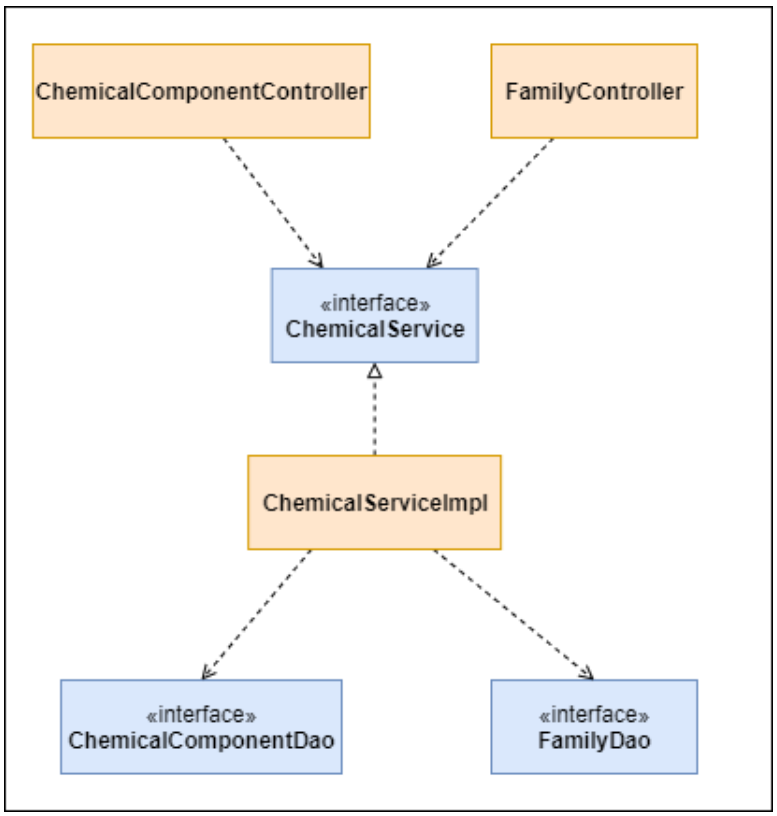

Figura 7.5: Diagrama de clases encargadas de la administración y listado de componentes qímicos y sus familias

7.7 Sprint 5 - Gestión de medicamentos y fórmulas

7.7.1 Análisis

Para finalizar la metología *bottom-up*, se implementarán las historias de usuario relacionadas con la gestión y visualización de medicamentos y sus fórmulas.

US16 - Administrar fórmulas
Como administrador quiero poder administrar un fórmula.
Tareas
Creación de la tabla ”Fórmula” en base de datos y de la entidad asociada.
Creación del servicio de fórmulas.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de fórmulas.

Cuadro 7.13: Sprint 5: Administrar fórmulas

US17 - Listar fórmulas
Como administrador quiero ver la lista de fórmulas.
Tareas
Creación del endpoint que devuelva la lista de fórmulas.
Creación de la vista que contiene la lista de fórmulas.

Cuadro 7.14: Sprint 5: Listar fórmulas

US20 - Administrar medicamentos
Como administrador quiero poder administrar un medicamento.
Tareas
Creación de la tabla ”Medicamento” en base de datos y de la entidad asociada.
Creación del servicio de medicamentos.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de medicamentos.

Cuadro 7.15: Sprint 5: Administrar medicamentos

US21 - Listar medicamentos
Como administrador quiero ver la lista de medicamentos.
Tareas
Creación del endpoint que devuelva la lista de medicamentos.
Creación de la vista que contiene la lista de medicamentos.

Cuadro 7.16: Sprint 5: Listar medicamentos

7.7.2 Diseño e implementación

La implementación de los casos de uso de este incremento se realizó de manera equivalente a la descrita en el *sprint* 3 (7.5.2) para enfermedades, alergias e intolerancias.

A continuación se incluye el diagrama usado en dichas historias de usuario 7.6:

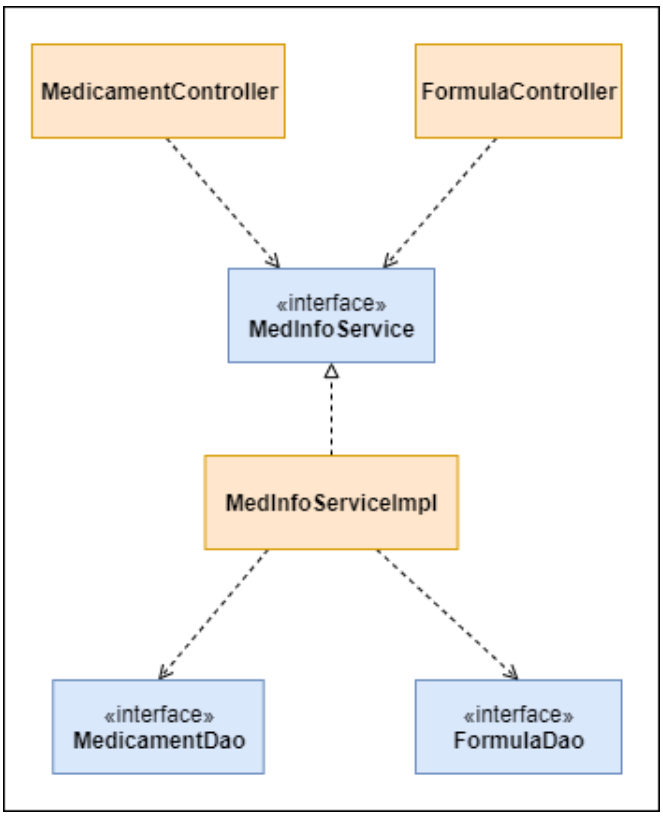

Figura 7.6: Diagrama de clases encargadas de la administración y listado de medicamentos y fórmulas

7.8 Sprint 6 - Comprobación de incompatibilidades

7.8.1 Análisis

Con todo el diseño de entidades ya diseñado e implementado, se comenzó a desarrollar las principales funcionalidades del proyecto, basadas en la comprobación de medicamentos.

US03 - Comprobación de incompatibilidades medicamento nuevo
Como invitado quiero comprobar las incompatibilidades del medicamento a recetar.
Tareas
Creación del servicio médico.
Creación de los endpoints necesarios para rellenar la información del historial médico y el medicamento a recetar.
Creación del endpoint necesario para obtener los resultados.
Creación de las vistas que contienen los formularios para rellenar la información y de la visualización de resultados.

Cuadro 7.17: Sprint 6: Comprobación de incompatibilidades medicamento nuevo

US04 - Comprobación de incompatibilidades medicamentos recetados
Como invitado quiero comprobar las incompatibilidades de los medicamentos ya recetados.
Tareas
Modificación del servicio para permitir comprobar medicamentos contenidos en el historial médico.
Adaptación de los endpoints para permitir esta funcionalidad.
Añadir un checkmark a una vista de las vistas del formulario que permita especificar el deseo de esta funcionalidad.

Cuadro 7.18: Sprint 6: Comprobación de incompatibilidades medicamentos recetados

US05 - Repetir paciente
Como invitado quiero poder repetir la hipótesis con los mismos datos.
Tareas
Adaptación de los endpoints para rellenar los formularios que permita precargar dicha información.
Adaptación de las vistas para mostrar la información cargada previamente.
Añadir un botón ”repetir paciente” en la vista de resultados.

Cuadro 7.19: Sprint 6: Repetir paciente

7.8.2 Diseño e implementación

US03 - Comprobación de incompatibilidades medicamento nuevo

La comprobación de incompatibilidades de un medicamento es el objetivo principal de esta aplicación. Inicialmente se aplicaba de manera única al medicamento a recetar.

Para la implementación de esta historia de usuario se siguió el esquema desarrollado en el *sprint 0: Análisis del ámbito de la aplicación*. A partir de este esquema se implementó un servicio con métodos independientes para cada uno de los filtrados, y un método principal que recogía todas las alertas generadas.

Con el servicio terminado, se implementó un controlador con los endpoints necesarios para el rellenado de la información y finalmente el uso de la funcionalidad implementada.

Con los controladores listos, se procedió a diseñar las vistas que mostraban dichas funcionalidades en la interfaz de usuario.

A continuación se muestra el diagrama de clases que intervienen en el proceso de comprobación de incompatibilidades 7.7.

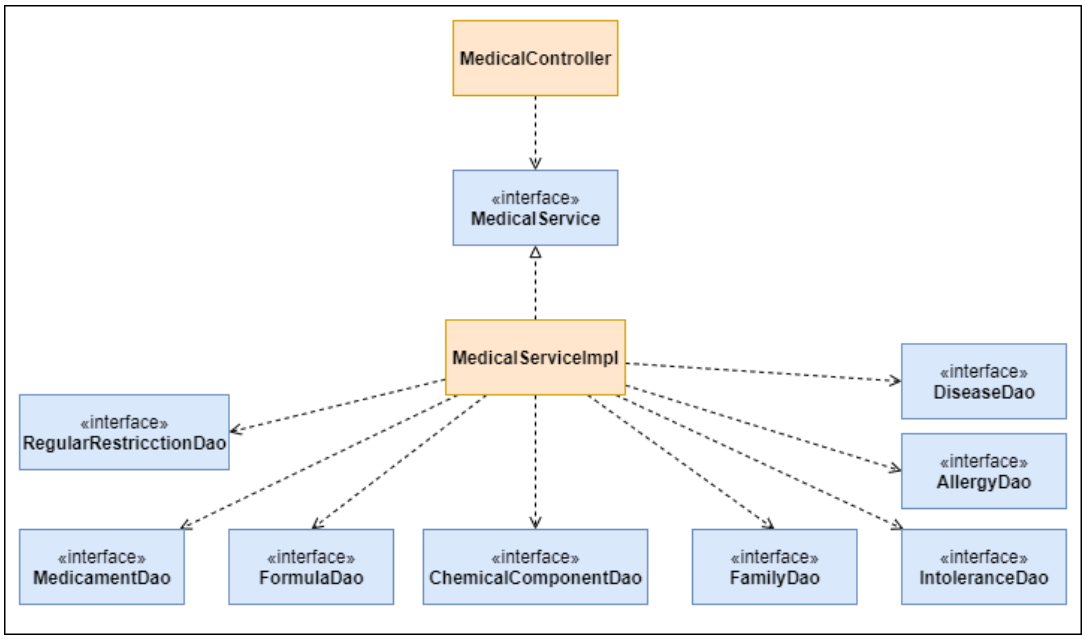

Figura 7.7: Diagrama de clases encargadas de la comprobación de incompatibilidades

US04 - Comprobación de incompatibilidades medicamentos recetados

Para la comprobación de medicamentos ya recetados, se utilizó la misma lógica de negocio

implementada en la historia de usuario anterior, modificándola ligeramente.

También se aprovecharon los endpoints de los controladores.

Finalmente, a la vista del formulario se le añadió una casilla de marcado para permitir el uso de dicha funcionalidad adicional, como se observa en la figura 7.8.

Figura 7.8: Vista del formulario de medicamentos

US05 - Repetir paciente

El requisito de esta historia de usuario, surgió de la idea de facilitar al médico comprobaciones repetitivas con ligeros cambios en las hipótesis planteadas.

Para ello, se modificaron los endpoints del controlador médico.

También se añadió el botón ”Repetir paciente” a la vista de resultados, figura 7.9, llevándolo al inicio del formulario con los datos de la anterior hipótesis ya rellenados. Esto permite que el médico cambie ciertos parámetros y repita la comprobación sin apenas esfuerzo.

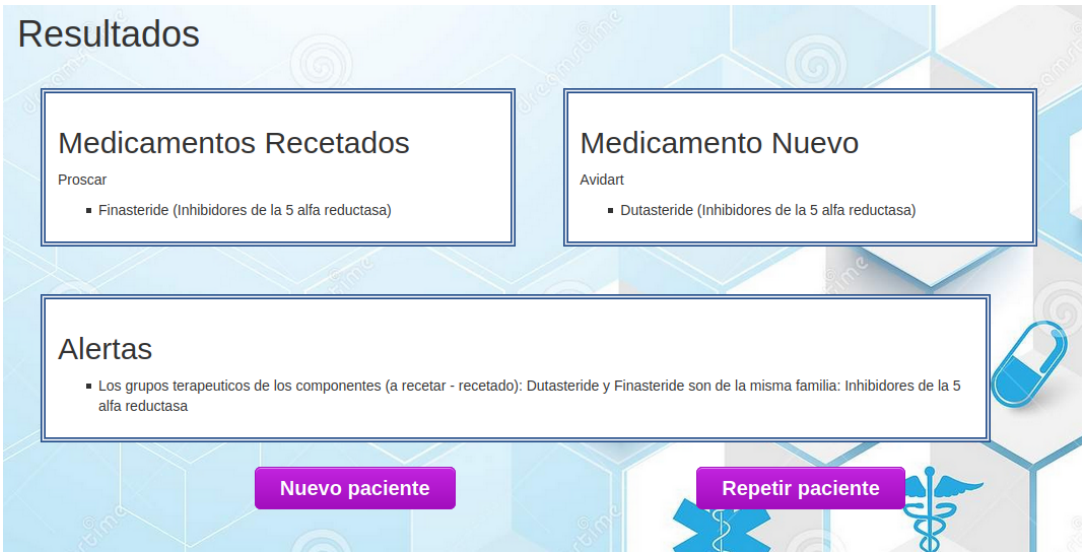

Figura 7.9: Vista de los resultados de la comprobación

7.9 Sprint 7 - Integración con sistemas externos

7.9.1 Análisis

En este incremento con un software funcional, se decidió realizar la prueba de concepto de integración con otras aplicaciones médicas.

Para ello se implementaron las siguientes historias de usuario:

US06 - Comprobación de incompatibilidades desde aplicación externa
Como usuario de aplicación externa quiero poder comprobar las incompatibilidades.
Tareas
Creación de un servicio *REST* accesible desde aplicaciones externas
Creación del endpoint necesario para dar acceso a esta funcionalidad.

Cuadro 7.20: Sprint 7: Comprobación de incompatibilidades desde aplicación externa

US07 - Importar información a partir del DNI
Como médico quiero poder importar la información a partir del DNI.
Tareas
Creación de un endpoint que permita esta funcionalidad en el controlador médico.
Adaptación de la vista del formulario para permitir rellenar el DNI.
Creación de una aplicación mock externa al software que devuelva información según el DNI dado.

Cuadro 7.21: Sprint 7: Importar información a partir del DNI

7.9.2 Diseño e implementación

US06 - Comprobación de incompatibilidades desde aplicación externa

Para desarrollar esta historia de usuario, se desarrolló un servicio *REST* aprovechando la lógica de negocio ya implementada en los *sprints* anteriores.

En este servicio se incluye un endpoint para el acceso a través de un método *POST*, dicho método recibirá un JSON que contendrá la información médica del paciente y el medicamento a recetar. Tras el procesado de dicha información se devolverá un JSON con la lista de alertas generadas.

La gestión de dicha información queda en manos de la aplicación sanitaria externa.

El diagrama de clases encargadas de dicho servicio se muestra en la figura 7.10:

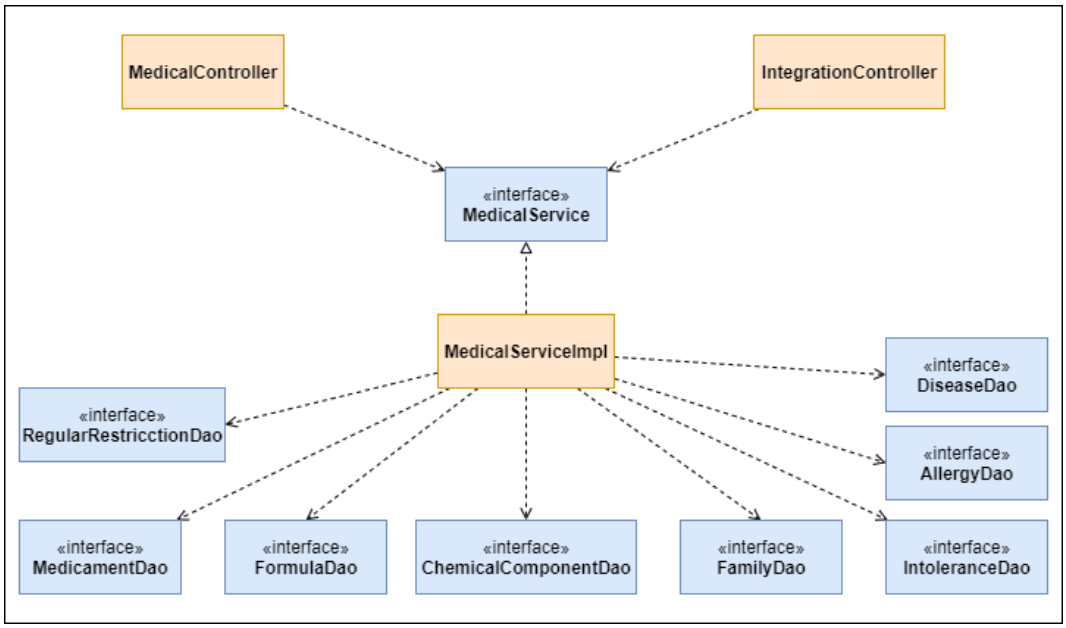

Figura 7.10: Diagrama de clases encargadas del servicio *REST* para la integración con un sistema externo

US07 - Importar información a partir del DNI

Esta historia de usuario surgió a raíz de la petición del médico de poder importar la información desde un servicio externo, perteneciente a una entidad médica, a la aplicación web.

Para la implementación se decidió incluir dicha funcionalidad al comienzo de de la vista de los formularios, con esto se permite la edición de la información tras importarla. También fue necesario añadir un endpoint que permitiese dicha consulta, como se muestra a continuación.

```
1   import org.springframework.web.client.RestTemplate;
2
3   //POST: api/medical/historyFormJsonLoad
4   @PostMapping("/historyFormJsonLoad")
5   public String jsonLoad(@ModelAttribute("dni") DniDto dni, Model
    model) throws URISyntaxException {
6
7       //Creación e inicialización de datos necesarios
8
9       RestTemplate restTemplate = new RestTemplate();
10    final String baseUrl = "http://localhost:8081/dniParser/" +
    dni.getDni();
11    MedicalHistoryJsonDto jsonInfo =
```

```
    restTemplate.getForEntity(baseUrl,
    MedicalHistoryJsonDto.class).getBody();

        //Traducción del objeto jsonInfo
        //Añadir datos al modelo

        return "medicalHistoryForm";
    }

```

Una vez terminada esta funcionalidad, surgió la necesidad de crear una aplicación *mockup* externa al proyecto que completase la prueba de concepto.

7.10 Sprint 8 - Gestión de usuarios

7.10.1 Análisis

US23 - Administrar usuarios
Como administrador quiero poder administrar un usuarios.
Tareas
Creación de la tabla ”Usuario” en base de datos y de la entidad asociada.
Creación del servicio de usuarios.
Creación de los endpoints necesarios para las operaciones (añadir, eliminar, modificar).
Creación de la vista de detalles de usuarios.

Cuadro 7.22: Sprint 8: Administrar usuarios

US24 - Listar usuarios
Como administrador quiero ver la lista de usuarios.
Tareas
Creación del endpoint que devuelva la lista de usuarios.
Creación de la vista que contiene la lista de usuarios.

Cuadro 7.23: Sprint 8: Listar usuarios

7.10.2 Diseño e implementación

La implementación de los casos de uso de este incremento se realizó de manera equivalente a la descrita en el *sprint* 3 para enfermedades, alergias e intolerancias.

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conclusiones

Con la finalización del proyecto se consideran alcanzados los objetivos iniciales planteados.

Por un lado, el objetivo principal del proyecto, la creación de un software que permitiese la comprobación de incompatibilidades de un medicamento cotejándolo con la información de un historial médico dado. Con este objetivo cumplido se agilizará dicho proceso de recetado y se ayudará a evitar posibles errores humanos.

Por otro lado, la gestión de la información referente a los medicamentos por parte de un administrador. Para permitir mantener actualizada dicha información, acorde con los cambios y descubrimientos en el campo de la farmacología.

Por ultimo, la prueba de concepto de integración con sistemas de salud existentes. Permitiendo importar información del historial clínico del paciente a la aplicación web, así como permitiendo a la aplicación de terceros tener acceso al procesado del núcleo de la aplicación por medio de un servicio REST.

Es importante enfatizar que este proyecto es de carácter académico, por tanto, muchas de las necesidades que debería abarcar en su aplicación en el mundo real han sido simplificadas (seguridad, conexiones…). Aún así, demuestra que la idea del proyecto podría aplicarse en un ámbito real y resultaría de gran ayuda.

Para finalizar, se quiere destacar el conocimiento adquirido por el alumno en diversos campos. El primero de ellos, el ámbito médico en el que se desarrolla la aplicación, demostrando conocimientos técnicos médicos para el desarrollo de los esquemas mencionados a lo largo de este documento. El segundo, la estandarización del trabajo a través de una metodología de trabajo incremental y documentada, esto mejoró enormemente la disciplina del alumno a la hora de llevar a cabo cada uno de los incrementos planteados. Por último, la aplicación y desarrollo de cada una de las tecnologías descritas en el capítulo 6, esto ayudó al alumno a mejorar su toma de decisiones en la selección de tecnologías para un proyecto, así como la gestión de las mismas.

8.2 Trabajo Futuro

Este proyecto sienta una base académica para el desarrollo de una aplicación real completa, con interesantes mejoras tanto en la eficiencia de gestión de la información como otras de especial interés en una aplicación médica. A continuación se mencionan algunas ideas que el alumno considera interesantes:

- **Aportación de información adicional de los medicamentos en la aplicación web:** Aportando enlaces a sitios web (e.g. Medimecum o Vademecum), en dicho sitio web se puede visualizar la información del medicamento de manera más técnica e incluso descargar la documentación de dicho medicamento.
- **Proposición de alternativas a los medicamentos con incompatibilidades:** Aumentando la capacidad analítica del software, se podría llegar a ofrecer alternativas viables ante la aparición de una incompatibilidad con el medicamento recetado.

Apéndices

Lista de acrónimos

API Application Programming Interface. 36

DOM Document Object Model. 39

NASI Número Asistencia Sanitaria Individual. 3

REST REpresentational State Transfer. 38

SCM Software Configuration Management. 39

Glosario

backend Capa ”invisible” de una aplicación encargada de la gestión de los datos y el funcionamiento de la lógica de negocio. 2, 35

bottom-up Estrategia de procesamiento de información características de las ciencias de la información, especialmente en lo relativo al software. ”De abajo a arriba”. 28

framework Estructura conceptual y tecnológica de asistencia definida con artefactos o módulos concretos de software para la organización y el desarrollo de software. 36

frontend Capa ”visible” de una aplicación, con la que interactúa el usuario. Interfaz de usuario. 35

mock Simulación de cierta funcionalidad. 60

mockup Modelo utilizado como esbozo o representación del diseño visual de una parte del sistema. 18

numero de historia Identificador usado para representar individualmente el historial médico de cada paciente. 3

plugin Elemento software que añade una funcionalidad adicional a un sistema. 39

top-down Estrategia de procesamiento de información características de las ciencias de la información, especialmente en lo relativo al software. ”De arriba a abajo”. 41

Bibliografía

[1] "Estudio sobre la mortalidad en el recetado de medicamentos." [En línea]. Disponible en: https://www.fidhs.org/noticia-ampliada/los-errores-en-la-medicacion-tercera-causa-de-muerte-en-espana

[2] B. M, "Algoritmos terapéuticos: ¿qué, porqué y porqué no?" *Rev.Soc.Valenciana Reumatología*, vol. 5, no. 4, pp. 1–2, 2014, https://svreumatologia.com/wp-content/uploads/2014/11/revista_svr_vol5_4.pdf.

[3] B. J. y. G. J. Rodríguez E, Puebla V, "El uso de herramientas de ayuda a la prescripción evitaría las intoxicaciones medicamentosas en los ancianos." *An. Sist. Sanit. Navarra*, vol. 37, no. 3, pp. 437–438, 2014, https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1137-66272014000300013&lng=en&nrm=iso&tlng=en.

[4] "Guías de scrum." [En línea]. Disponible en: https://scrumguides.org/

[5] "Pagina web de mysql." [En línea]. Disponible en: https://www.mysql.com/

[6] "Documentación de innodb." [En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html

[7] "Ejemplo de jpa." [En línea]. Disponible en: https://www.arquitecturajava.com/ejemplo-de-jpa/

[8] "Página web de hibernate." [En línea]. Disponible en: https://hibernate.org/

[9] "Página web de apache maven." [En línea]. Disponible en: https://maven.apache.org/

[10] "Documentación de spring boot." [En línea]. Disponible en: https://spring.io/projects/spring-boot

[11] "Documentación de spring data jpa." [En línea]. Disponible en: https://spring.io/projects/spring-data-jpa

[12] "Página web de lombok." [En línea]. Disponible en: https://projectlombok.org/

[13] "Página web de junit." [En línea]. Disponible en: https://junit.org/junit5/

[14] "Página web de jquery." [En línea]. Disponible en: https://jquery.com/

[15] "Página web de thymeleaf." [En línea]. Disponible en: https://www.thymeleaf.org/

[16] "Página web de eclipse." [En línea]. Disponible en: https://www.eclipse.org/ide/

[17] "Página web de git." [En línea]. Disponible en: https://git-scm.com/

[18] "Página web de github." [En línea]. Disponible en: https://github.com/

[19] "Página web de redmine." [En línea]. Disponible en: https://www.redmine.org/

[20] "Página web de spring initializr." [En línea]. Disponible en: https://start.spring.io/