
# Universidad de Valladolid

## Escuela de Ingeniería Informática

## Trabajo de Fin de Grado

Grado en Ingeniería Informática

Mención en Ingeniería del Software

# Alzheimer Platform

Autor:

**D. Guillermo Pastor Diez**


# Universidad de Valladolid

## Escuela de Ingeniería Informática

## Trabajo de Fin de Grado

Grado en Ingeniería Informática

Mención en Ingeniería del Software

# Alzheimer Platform

Autor:

**D. Guillermo Pastor Diez**

Tutor:

**Dña. Yania Crespo González-Carvajal**

# Índice general

# Índice de figuras

# Índice de cuadros

# Agradecimientos

Agradecer a mis padres por estar siempre a mi lado en los peores momentos y aguantar el temporal conmigo. A Juan, la guía en el camino y a Jennifer por insistir para que no bajara los brazos

Yania, mi tutora, que ha sido un referente y guía durante todo mi tiempo en la universidad fuera y dentro del aula. Y sobretodo por su comprensión con un mal estudiante como yo.

Y por último al equipo del TDAF en Telefónica, que ahora son 4ª Plataforma casi todos, porque me han llevado de la mano todo este tiempo y me han hecho crecer como persona y profesional. Pero sobretodo a Guido García Bernardo, la primera persona que apostó por mi y que me hizo creer que igual yo si valía para esto.

# Resumen y Abstract

## Resumen

En este proyecto se abordará el análisis, diseño y desarrollo de una plataforma destinada a desarrolladores, basada en ejercicios para pacientes con Alzheimer, para los propios pacientes y, finalmente, para investigadores de dicha enfermedad.

Esta plataforma permitirá a los pacientes y/o sus cuidadores, acceder fácilmente desde sus hogares, a un conjunto de ejercicios proporcionados por diversos desarrolladores. Por otro lado, permitirá a los investigadores conocer más sobre la enfermedad y su evolución a través de los datos obtenidos por la realización de ejercicios por parte de los pacientes.

## Abstract

The scope for this project will be the analysis, design and development of a platform for developer of patient exercicises, the patients of Alzheimer and finally for the researchers in this field.

This platform allows patients and/or carer, to have access from their homes in an easier way. These daily exercises is created by developers. Also, this platform allows researchers to get more knowledge about the disease and how it evolves through the data collected from the exercices done by the patients.

*Capítulo 1*

# Introducción

## 1.1. Objetivos

Este **Trabajo de Fin de Grado** tiene como objetivo, desarrollar una plataforma que satisfaga las siguientes necesidades:

- Acceso para pacientes y/o cuidadores de pacientes de Alzheimer, a ejercicios desde cualquier dispositivo móvil u ordenador con acceso a internet
- Publicación y gestión por parte de desarrolladores de ejercicios para pacientes de Alzheimer.
- Proveer de datos y métricas sobre el Alzheimer a investigadores a través de datos obtenidos de los ejercicios de los pacientes.

## 1.2. Motivación

La motivación de este proyecto, surge tras visitar en varias ocasiones un centro de día para personas mayores, en el cual se encontraban varios pacientes de Alzheimer. Al mismo tiempo, tuve un trato diario con una persona encargada de los cuidados de un paciente de Alzheimer en su ámbito familiar.

Tanto en los centros de día, como en los cuidados dentro del hogar a estos pacientes, se realizan ejercicios para tratar de ayudar en su tratamiento. Estos ejercicios, en la mayor parte de los casos, se encuentran en papel, en unos cuadernos de ejercicios similares a los utilizados en el colegio o, en algunos centros de día, o en sistemas informáticos de pago cerrados.

Para los centros de día públicos, es difícil acceder a dichos programas informáticos por el coste asociado que conllevan. Con esta plataforma, se ofrece la posibilidad de que cualquier persona que tenga un paciente de Alzheimer bajo su cuidado, o bien, centros de día y residencias con estos pacientes, puedan acceder a estos ejercicios a través de un dispositivo móvil o desde un ordenador, requiriéndose simplemente una conexión a internet.

Otro aspecto importante es conseguir variedad en los ejercicios para los pacientes y un flujo continuo de nuevas publicaciones dentro de la plataforma, ya que, de este modo, se consigue estimular al paciente y se evita caer en la monotonía. Para conseguir estos dos objetivos, es vital conseguir involucrar a los desarrolladores y crear una comunidad entorno a la plataforma. Ya que, en un principio, el desarrollo de estos juegos no reportará ningún beneficio al desarrollador más que el de proporcionar una herramienta más para mejorar la vida de los pacientes.

Dado que se trata de un acto voluntario por parte de los desarrolladores, se ha buscado que la interfaz para la publicación de juegos sea lo más amigable posible. Siempre ofreciendo la posibilidad de operar directamente con el API REST.

Por estos motivos, la idea de realizar como **Trabajo de Fin de Grado** una plataforma Open Source que pueda facilitar un poco la vida de pacientes, familiares/cuidadores e investigadores, es la base del proyecto.

La posibilidad de tener datos reales sobre los ejercicios realizados por los pacientes, resulta de utilidad para los investigadores de esta enfermedad, ya que pueden tener una foto de la evolución de un grupo de pacientes según la población, edad, etc.

Para desarrollar una versión lo más útil posible del proyecto, se visitó y consultó a varias personas encargadas de cuidar a pacientes de Alzheimer en el ámbito familiar, así como a profesionales que trabajan en clínicas con este tipo de pacientes. El objetivo de estas visitas es la priorización de las funcionalidades que resultarían más útiles en relación con los ejercicios diarios que los pacientes tienen que realizar.

Fruto de estas entrevistas y visitas, se confirmó la hipótesis de que es **complicado la renovación de estos ejercicios** dentro de los centros de día y en el ámbito familiar. Añadido a esto, se detectó que en muchas ocasiones, estos ejercicios se encuentran en cuadernillos de papel y **no pueden ser llevados a los hogares de los pacientes** porque tienen que estar disponibles para otros pacientes en el centro.

También los cuidadores, confirmaron la hipótesis de que los pacientes tiene **poca motivación a la hora de realizar estos ejercicios**.

Por lo tanto, podemos enumerar tres requisitos básicos a satisfacer por la plataforma:

- Conseguir que los ejercicios puedan ser realizados desde cualquier lugar.
- Facilitar el acceso a un abanico más amplio de ejercicios. Más variabilidad.
- Hacer más atractiva esta actividad diaria para los pacientes.

Como solución a estos tres requisitos planteados, la realización de una plataforma *Open Source* que permita a los cuidadores poder acceder a una mayor variedad de ejercicios, con un formato más interactivo y desde cualquier dispositivo móvil, parece una buena aproximación al problema.

Al tratarse de ejercicios que pueden influir en el tratamiento diario de los pacientes, la plataforma debe tener un sistema de validación para asegurar que los ejercicios son adecuados para el uso por parte de los pacientes.

## 1.3. Contextualización

¿Por qué es importante realizar avances es este campo? ¿Qué impacto tiene sobre la población? ¿Es útil una plataforma de este tipo? Para dar respuesta a estas preguntas y determinar la misión de este proyecto, se ha realizado una búsqueda a través de distintas organizaciones y publicaciones online.

### 1.3.1. ¿En que consiste el Alzheimer?

Según la *Alzheimer's Disease International* el Alzheimer supone la causa más común de demencia, siendo entre el 50 % y el 75 % de los casos. El Alzheimer destruye neuronas y nervios cortando la capacidad de transmitir información dentro del cerebro, teniendo un especial impacto en la zona donde se almacenan recuerdos.

Según la *Alzheimer's Association* [1], la principal organización sanitaria voluntaria sobre dicho tratamiento, ayuda e investigación del Alzheimer en Estados Unidos, en 2018 el Alzheimer presenta los siguientes datos sobre la población estadounidense.

- Supone la 6ª causa de mortalidad y la 5ª en personas mayores de 65 años.
- Afecta a 5.7 millones de estadounidenses y se espera que en 2050 este número pueda alcanzar los 14 millones.
- Desde el año 2000, el número de muertes asociadas a esta enfermedad se ha incrementado en un 123 %.
- 1 de cada 3 personas mayores fallecen a causa de esta enfermedad.
- El 83 % de los cuidados que reciben estos pacientes, provienen de familiares, amigos o asociaciones sin ánimo de lucro.

Actualmente, el Alzheimer no tiene cura, pero según instituciones como *Alzheimer's Disease International* hay ciertas prácticas que pueden ayudar a reducir el riesgo asociado como se puede ver en la Figura1.1.

Como se muestra en la Figura 1.1, estas prácticas no son complejas. Ya que, el hecho de comer de una manera saludable o realizar actividad física de forma habitual son realizables por parte de todos. Destacan también, las buenas prácticas relacionadas con evitar el consumo excesivo de alcohol y el no fumar, pueden reducir los casos de demencia.


Figura 1.1: Prácticas para la reducción de riesgo en demencia. Tomado de Alzheimer's disease international.Tomado de [5]

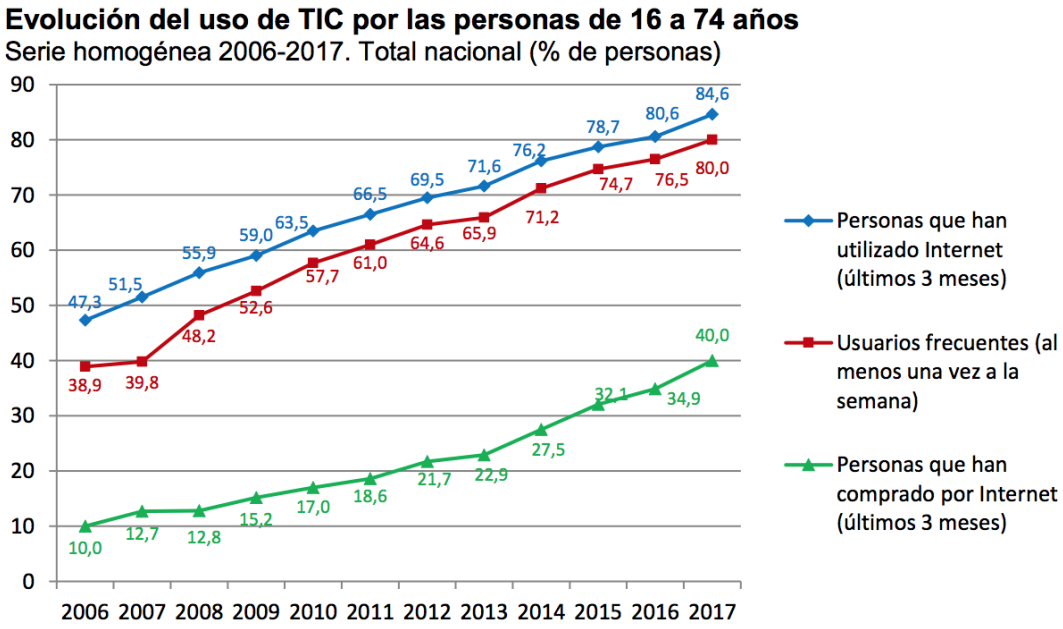


Figura 1.2: Gráfico sobre el uso de las TIC en 2017.

Realizar ejercicios para mantener una buena actividad cerebral, es el principal punto de acción sobre el que la plataforma puede tener influencia.

### 1.3.2. ¿Formato digital para los ejercicios?

La Figura 1.2, extraída de las notas de prensa del Instituto Nacional de Estadística, muestra un estudio realizado relacionado con el uso de las TIC en la población española.

El número de hogares equipados con ordenador, se ha incrementado un 1.3 % en 2017 con

respecto a 2016. Y el 53.4 % de los ordenadores, son tabletas.

Dentro de este estudio, también destaca que, en 2017, el 43,7 % de las personas ente 65 y 74 años habían usado internet en los últimos 3 meses y un 38,0 % lo usa al menos una vez por semana.

Aunque las personas mayores de 65 años no sean las que inicialicen la aplicación para ejercitarse de manera diaria, este papel está pensado para los cuidadores y familiares, que serán los que realicen los ejercicios. Este estudio demuestra, que cada vez hay una mayor predisposición en el uso de estas, ya no tan nuevas, tecnologías y la curva de aprendizaje no será tan pronunciada.

## 1.4. Estado del arte

De cara a la realización del proyecto, y una vez establecido el objetivo que se quiere alcanzar, merece la pena ver cómo en la industria se están resolviendo problemas similares.

Alzhup, desarrollada por una *startup* española, es una aplicación, que, según los autores, retrasa hasta 3 años el deterioro cognitivo de los pacientes. La terapia introducida dentro de la aplicación está validada por la Universidad de Salamanca y el Imserso.

Dentro de esta aplicación móvil, se proporcionan una serie de ejercicios de motivación, aeróbicos, de cálculo, de memoria, de relajación, etc. que trabajan distintas áreas cognitivas del cerebro. Cuenta con un banco de recuerdos personales, que simula la manera en la que el cerebro cataloga los recuerdos. Este banco de recuerdos, ayuda a que la familia y allegados del paciente se involucren más en el tratamiento del paciente con el fin de evitar que todo el peso recaiga sobre el cuidador principal.

Otra de las aplicaciones destinadas a ayudar a pacientes y familiares, sería, YoTeCuido. Se trata de una aplicación móvil disponible para Android y IOS.

La aplicación de YoTeCuido proporciona rutinas diarias, información para ayudar a los familiares y recursos cercanos como asociaciones, etc. Esta aplicación está más orientada a ser una guía para ayudar a los familiares principalmente.

Dado los problemas que pueden surgir cuando un paciente de Alzheimer sale sin supervisión a la calle, se ha creado una aplicación para plataformas móviles, Android e IOS, llamada Tweri.

Esta aplicación busca mejorar la calidad de vida para los pacientes en las primeras fases de la enfermedad. Dentro de la aplicación, se establecen tanto los límites en los paseos que puede recorrer el paciente, como la duración de los mismos. Para hacer este seguimiento de los pacientes, solo es necesario activar la aplicación cuando un paciente salga a dar un paseo. En caso de que se superen los límites de duración o distancia, se envían notificaciones al cuidador.

Esta aplicación está dentro de la mejora de vida para los pacientes y cuidadores, pero en un ámbito distinto, ya que se centra en permitir que los pacientes ganen independencia.

Brany App, es una aplicación destinada a la realización de ejercicios para evaluar y monitorizar la salud del cerebro. Está centrada en promover un estilo de vida saludable para el

cerebro.

El nicho de usuarios para esta aplicación no son en sí, solo los pacientes de Alzheirmer, sino que tiene como objetivo, que los usuarios también sean personas que no estén afectadas por la enfermedad pero quieran mejorar su memoria, habilidades de raciocinio y reducir el riesgo de desarrollar demencia.

Como punto inicial del uso de la aplicación, invita al usuario a realizar un test para determinar el "punto de salud" en el que se encuentra el usuario antes de comenzar a jugar.

Una aplicación que ha ganado mucha popularidad es Timeless, está aplicación fue desarrollada por una joven de 14 años cuya abuela padecía Alzheimer. Cuenta con dos funcionalidades principales:

Por un lado, la funcionalidad de Actualizaciones, la cual permite ver a los pacientes las nuevas cosas que están haciendo sus seres queridos. Esta funcionalidad imita en cierto modo el *timeline* de una red social. Los familiares envían de forma diaria imágenes a la aplicación. Mediante el reconocimiento facial, la aplicación detecta de quién se trata.

Por otro lado, cuenta con otro conjunto de funcionalidades que giran en torno al día a día del paciente, como por ejemplo, una funcionalidad que cuando un paciente llama repetidas veces a una persona, la aplicación le recuerda que acaba de llamar a esa persona y le pregunta si realmente quiere llamarla de nuevo.

Esta aplicación tiene un fuerte componente de inteligencia artificial ya que está basada en el reconocimiento facial.

Estos proyectos enumerados podemos entenderlos como pequeñas iniciativas de Startups o proyectos open source sin ánimo de lucro pero también hay grandes empresas como Samsung involucradas en este tipo de aplicaciones.

Samsung ha desarrollado una nueva aplicación, llamada Backup Memory. Esta aplicación desarrollada para dispositivos Android, permite crear un álbum de recuerdos vitaminados, para que los pacientes puedan recordar mediante imágenes a sus seres queridos. Se trata de un método no demostrado científicamente pero valorado positivamente por la comunidad de pacientes y familiares.

La aplicación también detecta cuando un familiar está cerca y muestra imágenes del paciente y el familiar para facilitar que el paciente se acuerde del familiar y evitar la complicada situación que puede darse en caso contrario.

Estas aplicaciones enumeradas, son solo una parte del total que compone todo el ecosistema de aplicaciones que están surgiendo entorno al Alzheimer, ya que se trata de una enfermedad en auge, en la que, cada vez más, la sociedad está poniendo especial interés en ella.

Como respuesta a este interés, la industria trata de dar soluciones parciales para paliar y mejorar la calidad de vida. Esta soluciones parciales se deben a que aún no entendemos cómo funciona el cerebro humano al 100 %, pero es una afirmación segura decir, que en el camino hacia la futura cura de esta enfermedad, la informática tiene un papel principal, no tanto como aplicaciones para paliar la degradación nerviosa, sino más orientadas al tratamiento masivo de

datos obtenidos a través de los resultados del trabajo con pacientes y la simulación.

La principal diferencia con respecto a las aplicaciones ya en el mercado, es, que este proyecto, se trata de una plataforma en lugar de una aplicación. Y debido a esto, puede plantear una propuesta de valor distinta, centrando su valor añadido, en proporcionar un espacio a los investigadores en el cual poder acceder a datos relacionados con el Alzheimer que puedan ser de utilidad para sus investigaciones (como pueden ser gráficas de actividad para un determinado sector de la población). Por otro lado, la plataforma permite a los desarrolladores generar nuevo contenido en formato de ejercicios para que puedan ser realizados por los pacientes, no es objetivo de la plataforma desarrollar este contenido, sino ser un contenedor para facilitar el acceso y la variedad a los mismos.

*Capítulo 2*

# Planificación del proyecto

Un aspecto fundamental del desarrollo de un proyecto, es la planificación del mismo, de cara tanto a detectar riesgos y trazar planes de acción para solventarlos, así como para una adecuada priorización de tareas para evitar un bloqueo durante el desarrollo.

## 2.1. Metodología

Dentro del desarrollo ágil, hay un gran número de metodologías que una organización puede elegir; y todas ellas pueden y deben ser adaptadas a las necesidades de la organización para contribuir mejor a los objetivos de la misma.

Igual que ocurre cuando se trata de lenguajes de programación, no hay una mejor que otra, sino que cada metodología se adapta mejor a ciertos requisitos que otra. Por lo tanto, para seleccionar una, es necesario entender las necesidades del equipo y ver cuál se adecúa más a ellas.

Como se ha indicado previamente, todas las metodologías para ser 100 % efectivas, tienen que adaptarse a la organización y, muy posiblemente, al equipo.

Hay un amplio abanico de metodologías usadas dentro de la industria: *Cystal Clear*, *Agile Unified Process*, etc. Las más populares son *eXtreme Programming* y *Scrum*.

La metodología elegida para la realización de este proyecto será *Scrum* debido a los siguientes motivos:

- Por un lado, este tipo de metodologías han demostrado ser muy útiles para el desarrollo de prototipos por pequeños grupos de trabajo.
- Otro motivo a tener en cuenta, es la experiencia que el alumno ya tenía previa al desarrollo del proyecto.
- Entregas de funcionalidad al finalizar cada *Sprint*. Tener la posibilidad de entregar pequeños avances del producto al cliente y recibir *feedback* sobre esos avances, permite una rápida iteración y mejora.

- Productividad y flexibilidad. Dos aspectos requeridos por el proyecto y que encajan con esta metodología.
- Capacidad de evolución y ajuste de requisitos. El uso de ésta metodología permite realizar cambios sobre el proyecto y la adhesión de nuevos requisitos para la mejora del mismo, al tiempo que se avanza en el desarrollo.

Como la propia documentación de *Scrum* indica, es necesario adaptar esta metodología para que encaje correctamente con el equipo y el proyecto.

### 2.1.1. Adaptación de Scrum

Un aspecto indicado en [17] es el tamaño mínimo dentro de los equipos. *Scrum* establece que, el tamaño mínimo debería ser de 3 desarrolladores, porque de lo contrario, las interacciones se reducen y esto se traduce en una reducción de la mejora sobre la productividad que se podría obtener aplicando esta metodología a un equipo de mayor tamaño. Sin embargo, para la realización del proyecto, dado que es individual, se adapta el proceso para poder utilizar esta metodología en equipos de 1 desarrollador.

En esta adaptación, la tutora del proyecto, Yania Crespo, toma el papel de *product owner* ayudando a la priorización de tareas y aportando *feedback* continuo sobre el desarrollo realizado, tanto sobre el código como sobre la documentación.

Dado que el equipo se componía solo del estudiante, las *daily meeting*, que son características de esta metodología, se suprimieron al carecer de valor.

Del mismo modo el *sprint planning*, *sprint retrospective* y el *sprint review* se unen en una sola sesión semanal para aprovechar mejor el tiempo con el *product owner*.

En cada una de estas reuniones con el product owner se buscan diversos objetivos:

- Realizar una demo de la funcionalidad conseguida durante el sprint anterior.
- Ver las dificultades afrontadas durante el pasado sprint y cómo se superaron.
- Determinar las siguientes tareas para el sprint en función de cual aporta más valor al producto desde el punto de vista del *produt owner*.
- Planificar las tareas que se hayan quedado fuera del *sprint*.
- Valorar si la planificación ha resultado la adecuada (esfuerzo estimado frente a esfuerzo real).
- Consejos y correcciones sobre el estado del proyecto.

Para el desarrollo del proyecto se ha determinado una duración del *sprint* de 7 días. Esto no supone como tal un cambio, ya que la metodología no establece una duración determinada

(pero si aconseja dos semanas de duración). La decisión de acortar el tiempo por *sprint* se debe, a que permite llevar un seguimiento más continuado del proyecto frente al *product owner* y dar una visión real del estado del proyecto. Por otro lado, presentar los avances y problemas en ciclos de iteración tan cortos facilita recibir *feedback* por parte del *product owner* de manera completamente ágil. Por último, otro punto interesante relacionado con una duración de 7 días dentro del sprint, es, que obliga a determinar un mayor número de tareas de pequeño tamaño, lo cual facilita determinar la duración de las mismas y reduce la desviación que se puede producir durante la planificación del *sprint*.

En el desarrollo de proyectos ágiles, es habitual el uso de medidas para estimar el esfuerzo y tiempo que requerirá el desarrollo de las tareas dentro de un sprint como se indica en [3].

La decisión de la duración de las tareas dentro de las metodologías ágiles, se toma conjuntamente entre todo el equipo, por ello, en una situación normal, lo más común sería que se produjeran discusiones entre varios miembros del equipo. Para evitar esto, se idearon los siguientes métodos:

- Cartas de poker. Consta de 8 cartas, con los siguientes valores: 1/2, 1, 2, 3, 5, 6, 7 e infinito, dentro de la reunión cada miembro del equipo da una carta con relación a la duración que tendrá la tarea.

- Sucesión de Fibonnaci. Basado en el hecho de que al aumentar el tamaño de las tareas aumenta también el margen de error, surgió una variante que consiste en emplear números de la sucesión de Fibonacci para realizar las estimaciones. En este caso, se levanta la carta con el número de Fibonnaci que más se aproxime a la estimación.

- Tamaño de camisetas. Se realiza una estimación de la duración de las tareas en función del tamaño de camisetas (S,M,L,XL...). Para evitar confusiones es necesario asignar un valor concreto a cada talla.

Lo ideal sería, realizar una ronda por tarea, pero en caso de que las valoraciones sean muy dispares, se aconseja que los miembros que hayan dado las valoraciones más extremas den sus motivaciones y se realice una segunda ronda. Otro elemento importante en caso de que sea necesario una segunda ronda, es ponderar con un mayor peso, la valoración realizada por el miembro del equipo con más experiencia sobre la tarea.

Esta estimación de dificultad tiene un carácter empírico y se va ajustando a medida que se avanza dentro del desarrollo del proyecto. En este caso, no se ha sacado toda la funcionalidad posible a esta práctica porque se pierde la propiedad de contraste con el resto de valores que proporcionan otros desarrolladores del equipo, que en este caso es solamente uno.

En este proyecto, se ha decido elegir el uso de tamaños de camisetas para esta tarea de estimación, ya que las desventajas que puede presentar este método [4] se ven paliadas ya que solo hay un único desarrollador y la comunicación de tamaño y tareas es clara. Para manejar dentro del proyecto una nomenclatura común se han establecido las siguientes tallas:

- **Talla S**, esta talla hace referencia a tareas de fácil realización y duración no superior a 1 horas.
- **Talla M**, esta talla hace referencia a tareas de dificultad media y duración no superior a 2 horas.
- **Talla L**, esta talla hace referencia a tareas de dificultad elevada y duración no superior a 4 horas.
- **Talla XL**, esta talla hace referencia a tareas especialmente difíciles por lo que se le asigna una duración mayor respecto de las anteriores. Esta talla hace referencia a tareas que tengan una duración no superior a 6 horas. Dado que engloba un grupo de tareas difíciles de estimar, en algunos casos, esta talla puede usarse para tareas que superen la asignación de horas de trabajo/día dentro de un *spint*.

Dado que durante el proyecto hay que realizar una memoria del desarrollo, hubo *sprints* dedicados propiamente al avance de esta documentación en la cual no se presentó ante el *product owner* nueva funcionalidad sino los avances realizados dentro de dicha memoria.

### Velocity

En la planificación de un proyecto ágil, es necesario determinar una medida denominada *velocity*, o lo que es lo mismo, cuántas tareas pueden realizarse dentro del *sprint*. Este valor no es una constante que se fije al inicio del proyecto y no deba cambiar. Por el contrario, la estimación de trabajos es una tarea complicada que requiere de experiencia previa en la realización de la tarea y puede verse afectada si el equipo cambia, por lo que suele ser recalculada. En el desarrollo del proyecto, la *velocity* será calculada en tres puntos: inicio, mitad y en la parte final del proyecto. El cálculo inicial, nos permite determinar la cantidad de trabajo que estimamos que se podrá realizar dentro de cada iteración. El calculo de la *velocity* a mitad del proyecto, nos sirve como valor de control para realizar una corrección sobre la medida inicial del proyecto. Por último, el calculo final de la *velocity* nos sirve para corregir de nuevo la medida tomada en caso de que las últimas tareas del proyecto, por lo general más costosas, hayan modificado su valor.

- ***Velocity* inicial**: 20
- ***Velocity* media**: 14
- ***Velocity* final**: 18

## 2.2. Alcance del proyecto

Para poder estimar de forma correcta la duración del proyecto así como el orden de las tareas, es necesario establecer el alcance del proyecto.

Como se ha introducido anteriormente, el objetivo del proyecto es desarrollar un plataforma que permita a los usuarios iniciar sesión y registrarse. Según su rol, realizarán tareas de índole diferente. Los desarrolladores de juegos, registrarán juegos en la plataforma. Los validadores darán el visto bueno a estos juegos registrados. Adicionalmente para el caso de los pacientes, la plataforma permitirá acceder a distintos ejercicios desarrollados por desarrolladores externos al proyecto. También la plataforma provee de una serie de gráficos y datos destinados a los investigadores.

La realización de este proyecto, implicará realizar tareas de programación en diversos lenguajes de programación, desarrollar una arquitectura basada en microservicios e interconectar todas las funcionalidades mencionadas previamente.

Dentro del proyecto se establecen distintos hitos:

- **23 de Marzo de 2018** Entrega de primera prueba de concepto.
- **6 de Abril de 2018** Inicio de sesión dentro de plataforma.
- **13 de Abril de 2018** Primera versión de documentación(capítulos y definición de contenidos).
- **27 de Abril de 2018** Registro de usuario dentro de plataforma.
- **11 de Abril de 2018** Registro de ejercicio dentro de plataforma.
- **25 de Mayo de 2018** Administración de ejercicios dentro de plataforma.
- **1 de Junio de 2018** Entrega de diagramas.
- **8 de Junio de 2018** Visualización de datos en Grafana.
- **27 de Junio de 2018** Entrega de documentación para el proceso de revisión final.
- **4 de Julio de 2018** Code Complete, entrega de toda la funcionalidad desarrollada.
- **6 de Julio de 2018** QA Complete, resolución de *bugs* encontrados en el entorno de demo.

Es necesario determinar estas fechas finales como objetivos a largo plazo, pero dentro de la planificación del proyecto, contamos con entregas semanales de funcionalidad que permiten ir estimando la desviación sobre esta planificación original.

## 2.3. Calendarización

La realización del proyecto queda acotada entre las siguientes fechas:

- **Fecha de inicio** el día **01/03/2018**
- **Fecha de fin** el día **06/07/2018**

De este modo, desde la fecha de inicio hasta la fecha de QA Complete en el entorno de demo, hay 19 semanas entre las cuales hay que repartir las 300 horas sobre las que está estimado el desarrollo de este proyecto. Dentro de este cálculo, no se han tomado días de descanso ya que es necesario emplear todo el tiempo disponible.

De este modo, tenemos que cada semana el alumno deberá dedicar un mínimo de 15,78 horas semanales para cumplir con el alcance establecido.

### 2.3.1. Gestión de riesgos

Dentro de este apartado se enumeran los riegos detectados previo desarrollo del proyecto. Se estima su impacto y probabilidad de ocurrencia sobre el proyecto y se establece un plan de acción para solventar dicho riesgo en caso de que ocurra.

**Riesgo**: Tiempo de formación en AngularJS escaso.
**Impacto**: Alto.
**Probabilidad de ocurrencia**: 60 %
**Acción preventiva**: Introducir tareas de front end en cada ***sprint*** y dar más tiempo del normal para buscar más documentación y ampliar los conocimientos sobre AngularJS paulatinamente.
**Plan de acción**: Reestructurar la calendarización para añadir tiempo específico dedicado al aprendizaje de AngularJS

**Riesgo**: Jornada laboral del alumno bloqueé el desarrollo del proyecto.
**Impacto**: Alto.
**Probabilidad de ocurrencia**: 70 %
**Acción preventiva**: Tomar un mes de vacaciones dedicado íntegramente al desarrollo del proyecto, así como desgranar las historias de usuario en tareas atómicas para poder repartirlas a lo largo de la semana y no generar picos de trabajo.
**Plan de acción**: Reestructurar las horas dedicadas por semana para recuperar el tiempo perdido de desarrollo.

**Riesgo**: Estudiante caiga enfermo.
**Impacto**: Bajo.
**Probabilidad de ocurrencia**: 50 %
**Acción preventiva**: Reparto de tareas de manera equitativa, para minimizar el impacto que tenga parar el desarrollo durante un tiempo.
**Plan de acción**: Priorizar las tareas de documentación y diseño mientras el estudiante este enfermo.

**Riesgo**: Tiempo de formación en Play Framework escaso.

**Impacto**: Bajo.
**Probabilidad de ocurrencia**: 30 %
**Acción preventiva**: Revisar en profundidad los ejemplos proporcionados por Play Framework y tratar de elaborar una prueba de concepto con las funcionalidades más básicas del proyecto.
**Plan de acción**: Reestructurar la calendarización para añadir tiempo específico dedicado al aprendizaje de Play Framework

**Riesgo**: Estimación del tiempo de desarrollo baja para las funcionalidades de *backend*.
**Impacto**: Medio.
**Probabilidad de ocurrencia**: 50 %
**Acción preventiva**: Espaciar las tareas de *backend* dentro del *sprint*, para poder solventar el exceso de tiempo consumido por alguna de las tareas.
**Plan de acción**: Reestructurar la calendarización para añadir tiempo específico dedicado al *backend*

**Riesgo**: Desarrollo de la documentación insuficiente.
**Impacto**: Alto.
**Probabilidad de ocurrencia**: 80 %
**Acción preventiva**: Priorizar el desarrollo de documentación, dedicando un mayor porcentaje del tiempo de trabajo (relación de 2/3) frente a tareas de desarrollo.
**Plan de acción**: Detener el avance de funcionalidad hasta alcanzar punto deseado dentro de documentación.

**Riesgo**: Pruebas de interacción de los componentes escasas.
**Impacto**: Alto.
**Probabilidad de ocurrencia**: 45 %
**Acción preventiva**: Desplegar todos los componentes utilizando imágenes *docker* y probar su interacción de manera periódica.
**Plan de acción**: Realizar tests de integración de los componentes y probar de manera periódica recorrido completo por la plataforma.

## 2.4. Seguimiento del proyecto

Dentro de este apartado se irán analizando los avances realizados en el proyecto así como los riesgos presentados y si se ha seguido la planificación prevista para ellos.

Aprender de los posibles errores cometidos durante el proyecto y evitar que se produzcan de nuevo dentro del proyecto y en proyectos posteriores es un elemento de gran valor.

### 2.4.1. División de tareas por sprint

#### Sprint 1

Este sprint estuvo centrado en la preparación del entorno para el desarrollo del proyecto, así como realizar una mínima prueba de concepto para determinar si el alcance del proyecto es realista.

Backlog priorizado en la tabla 2.1

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| POC de aplicación Angular | Alta | M | Sí |
| Configuración del entorno | Media | S | Sí |
| Formación Grafana | Alta | XL | Sí |
| Investigación sobre arquitecturas de microservicios | Alta | XL | Sí |

Cuadro 2.1: Backlog de tareas para sprint 1.

#### Sprint 2

En este *sprint* el objetivo era mostrar al *product owner* una primera versión de lo que serían las vistas de la aplicación, para ello era necesario una tarea previa de diseño de interfaces. El elemento seleccionado para la demo fue la funcionalidad de registro de un paciente dentro del servidor.

Backlog priorizado en la tabla 2.2

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Desarrollo vistas en Angular | Alta | L | Sí |
| Esqueleto de servicio Registro | Media | M | Sí |
| Integración Play y Postgres | Alta | XL | Sí |
| Test de servicio registro | Alta | L | No |
| Desarrollo de servicio registro | Media | L | No |
| Diseño de interfaces | Baja | S | No |

Cuadro 2.2: Backlog de tareas para sprint 2.

Dificultades encontradas durante el *sprint*:

Durante la realización de la tarea de Desarrollo del servicio de registro se produjo un problema con la integración en base de datos que realiza el propio Postgres. Play Framework incorpora una funcionalidad denominada *evolutions* que obliga a hacer un *dump* de la base de datos y después realizar todas las inserciones y creaciones de tablas pertinentes cuando el modelo de datos se modifica. Esta funcionalidad es muy útil para evitar fallos durante el desarrollo, pero el desconocimiento de la misma llevó a un retraso sobre la realización de las tareas para este *sprint*.

Dentro del *sprint* hubo tres tareas que se no pudieron completar y por tanto entraron en el *backlog* de tareas para ser priorizadas en futuras iteraciones.

La imposibilidad por falta de tiempo de desarrollar el 50 % de las tareas planificadas para el *sprint* es un indicativo de que la planificación dentro del *sprint* así como la estimación de la talla de las tareas podría estar siendo demasiado optimista.

### Sprint 3

La entrega de funcionalidad planificada para esta iteración era conseguir realizar un inicio de sesión. Dentro del *sprint* se introducen de una manera más notoria las tareas de documentación con el objetivo de no crear una brecha entre el desarrollo y la documentación del proyecto.

Backlog priorizado en la tabla 2.3

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Inicio capítulo Planificación | Media | M | Sí |
| Actualización de registro tareas | Alta | S | Sí |
| Esqueleto de servicio Login | Alta | L | No |
| Test de servicio Login | Alta | L | No |
| Inicio capítulo Introducción | Baja | S | Sí |
| Sección Sistema Asíncrono | Baja | M | Si |
| Patrón Observador | Baja | M | Sí |

Cuadro 2.3: Backlog de tareas para sprint 3.

Dificultades encontradas:

Pese a que la estimación de tareas de este segundo sprint se había mejorado con respecto al primer *sprint*. En esta segunda iteración se manifestó uno de los riegos previstos en la gestión de riesgos: el alumno calló enfermo, por lo que atendiendo al plan de acción se priorizaron las tareas de documentación para impactar lo mínimo posible a la planificación del *sprint*.

De nuevo las tareas que no se pudieron realizar, entraron en el *backlog* de tareas para ser priorizadas en futuras iteraciones.

Al realizar solo tareas de documentación no se pudo realizar una demo al *product owner* con nuevas funcionalidades.

### Sprint 4

Dado que durante el *sprint* anterior no se pudieron completar las tareas necesarias para realizar la demo de inicio de sesión, para este *sprint* se busca poder completar el desarrollo necesario para realizar un inicio de sesión frente al *product owner*.

Backlog priorizado en la tabla 2.4

Dificultades encontradas:

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Esqueleto de servicio de Login | Alta | M | Sí |
| Test de servicio de Login | Alta | M | Sí |
| Desarrollo de servicio de Login | Alta | L | No |
| Metodología Scrum | Media | S | Sí |
| Adaptación de Scrum | Media | S | Si |
| Inicio Capítulo Herramientas | Media | S | Sí |
| Inicio sección User Stories | Alta | L | Sí |
| Apartado Microservicios | Baja | L | No |

Cuadro 2.4: Backlog de tareas para sprint 4.

Durante el desarrollo de este *sprint* no se encontraron dificultades que impidieran o retrasaran el desarrollo del resto, sin embargo dos tareas planificadas: la redacción dentro de la documentación el apartado de Microservicios y el desarrollo de la lógica necesaria para hacer login dentro del servidor, no pudieron ser completadas por falta de tiempo. Esta imposibilidad se debió a que se sobrestimó la capacidad de trabajo realizable durante una iteración, los cambios de contexto necesarios para el desarrollo de la documentación y el código no fueron tenidos en cuenta e imposibilitaron un desarrollo paralelo de las tareas. Este dato se apuntó para tener en cuenta en los sucesivos *sprint planning*.

Ambas tareas pasaron al *backlog* para ser priorizadas en alguno de los siguientes *sprint planning*. Al tratarse una de las tareas de aspectos de documentación no afectó a las demos pensadas para realizar con el *product owner*.

Dentro de este *sprint* se han conseguido completar un mayor número de tareas, eso se debe a dos elementos. Por una parte, las tareas de desarrollo ya eran conocidas del sprint anterior por lo que el esfuerzo para implementarlas fue menor; y por otra parte, varias de las tareas eran de documentación (este tipo de tareas tienen una menor incertidumbre a la hora de estimar duración y dificultad).

### Sprint 5

La principal funcionalidad que quería ser mostrada al *product owner* era realizar un recorrido *end to end* desde la vista en la aplicación Angular hasta el servidor y visualizar el retorno de información por parte del servidor desde la vista.

Backlog priorizado en la tabla 2.5

Dificultades encontradas:

Durante el desarrollo de este *sprint* no se pudieron realizar ninguna de las tareas dado que un pico en la carga de tareas dentro del trabajo del alumno imposibilitó el desarrollo. Este riesgo había sido previamente detectado y reflejado en la documentación, para reducir la posibilidad de que sucediera se dedicó un tiempo en exclusiva al proyecto al comienzo del mismo para avanzar en él y reducir la carga de trabajo.

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Comunicación cliente/servidor | Alta | XL | No |
| Añadir validaciones en Angular App | Media | M | No |
| Test registro paciente en Angular App | Media | M | No |
| Registro pacienteen Angular App | Alta | L | No |
| Apartado API REST | Baja | S | No |
| Apartado Patrón Repository | Baja | S | No |
| Esqueleto de servicio para gestión de ejercicios | Alta | M | No |
| Apartado Microservicios | Baja | L | No |

Cuadro 2.5: Backlog de tareas para sprint 5.

En este *sprint* queda reflejado que la acción preventiva de tomar un tiempo de vacaciones para poder avanzar dentro del proyecto lo máximo posible no tuvo efecto, ya que a la vuelta del trabajo se produjo un bloqueo. Esto se debe a que, a pesar de haber dejado un mayor tiempo inicial para trabajar, las primeras tareas del proyecto suelen tener una menor complicación, por lo que en los sucesivos *sprints* al aparecer tareas de mayor peso, la acción preventiva pierde impacto.

También cabe destacar que las fluctuaciones producidas sobre la carga de trabajo del alumno no siguen un patrón definido por lo que es difícil prever de antemano cuándo se producirán, ya que los requisitos y tiempos dentro del producto dependen de factores externos.

Debido a que no se pudieron avanzar las horas previstas para esta semana, se tomó la acción de reestructurar las horas que estaban planificadas para esta semana para poder recuperar el tiempo perdido y no retrasar todo el proyecto como estaba determinado dentro del apartado de riesgos.

Como no se pudieron realizar tareas, se priorizaron para ser continuadas en los sucesivos *sprints* en función de la determinación conjunta con el *product owner*.

### Sprint 6

El objetivo principal de este *sprint* fue conseguir finalizar una prueba de concepto para la comunicación entre la aplicación Angular y el servidor. Ya que una vez resulta esa problemática se podría abordar la plataforma desde un punto de vista *ent to end*.

Backlog priorizado en tabla 2.6

Dificultades encontradas:

Durante este *sprint* se consiguieron completar todas las tareas excepto la redacción de la documentación sobre Microservicios, la cual no pudo ser realizada por falta de tiempo durante el *sprint*. Dado que esta tarea durante varios *sprints* no ha podido realizarse, en los sucesivos *sprint*

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| POC Comunicación cliente/servidor | Alta | XL | Sí |
| Añadir validaciones en Angular App | Alta | M | Sí |
| Registro en Angular App | Alta | L | Sí |
| Esqueleto de servicio para gestión de ejercicios | Alta | M | Sí |
| Apartado API REST | Baja | S | Sí |
| Apartado Patrón Repository | Baja | S | Sí |
| Apartado Microservicios | Baja | L | No |

Cuadro 2.6: Backlog de tareas para sprint 6.

*plannings* se tomó en cuenta para determinar tanto una posible infraestimación de la tarea como del sprint en general.

Dado que era la primera vez que se trabajaba con Play Framework y Angular JS es realmente difícil estimar la duración de estas nuevas tareas. Además, las tareas de redacción en este sprint ya se toman como un problema real, ya que hasta ahora en la mayor parte de los casos se realizaba una primera versión de las mismas. El tiempo dedicado a la elaboración de contenido para la memoria y corrección de este va en aumento.

### Sprint 7

Para este *sprint* se priorizo la realización de test sobre el servicio de Inicio de sesión y de Registro de usuarios para determinar que diseño se utilizaría posteriormente y como sería implementado. De manera paralela se realizaron tareas de mejora sobre la documentación.

Backlog priorizado en tabla 2.7

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Test de servicio de inicio de sesión | Alta | L | Sí |
| Test de servicio registro | Alta | L | No |
| *Refactor* dentro de Angular App | Media | M | No |
| División de documentación | Baja | S | Sí |
| Corregir documentación | Alta | S | Sí |
| Instalación y configuración de Grafana | Alta | L | Sí |
| Diagrama de Arquitectura | Media | S | Sí |

Cuadro 2.7: Backlog de tareas para sprint 7.

Dificultades encontradas:

Durante este *sprint* no se pudieron realizar las tareas de desarrollo de tests para el servicio de registro dentro del servidor ni la clase *helper* dentro de la aplicación Angular para evitar la repetición de código. De nuevo estas tareas quedaron registradas para ser priorizadas en la siguiente reunión con el *product owner*.

### Sprint 8

Backlog priorizado en tabla 2.8

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Desarrollo de servicio de registro | Alta | L | No |
| Refactor dentro de Angular App | Media | M | No |
| Esqueleto de servicio para gestión de ejercicios | Alta | M | Sí |
| Apartado Patrón Repository | Baja | S | Sí |
| Apartado Microservicios | Media | L | No |

Cuadro 2.8: Backlog de tareas para sprint 8.

Dificultades encontradas:

### Sprint 9

Dentro de este sprint el entrégale negociado con el *product owner* fue los diagramas correspondientes al desarrollo que se venía haciendo desde el comienzo del proyecto.

Backlog priorizado en la tabla 2.9

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Diagrama de modelo de dominio | Alta | L | Sí |
| Diagrama de secuencia para registro de ejercicio (backend) | Alta | M | No |
| Diagrama de secuencia para registro de desarrollador (backend) | Alta | M | No |
| Diagrama de secuencia para registro de paciente (backend) | Alta | M | No |
| Diagrama de secuencia para login de usuario (backend) | Alta | M | No |
| Diagrama de componentes | Media | M | Sí |
| Diagrama de arquitectura | Baja | M | Sí |

Cuadro 2.9: Backlog de tareas para sprint 9.

Dificultades encontradas:

Durante el desarrollo de los diagramas surgió una duda relacionada con el uso de funciones lambda dentro del código y su representación entro de los diagramas que impidió la finalización de esas tareas.

Por lo tanto, la planificación fue correcta pero al no saber resolver la duda con las funciones lambda la tarea fue avanzada pero no planificada y pasó al siguiente *sprint* para ser finalizadas por la prioridad de las mismas.

### Sprint 10

El objetivo de este *sprint* era conseguir un entregable con el *feedback* aportado por el *product owner* para la conclusión de los diagramas de secuencia.

Backlog priorizado en la tabla 2.10

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Diagrama de secuencia para registro de ejercicio (backend) | Alta | S | No |
| Diagrama de secuencia para registro de desarrollador (backend) | Alta | S | No |
| Diagrama de secuencia para registro de paciente (backend) | Alta | S | No |
| Diagrama de secuencia para login de usuario (backend) | Alta | S | No |
| Diagrama de secuencia para registro de ejercicio (frontend) | Alta | M | No |
| Diagrama de secuencia para registro de desarrollador (frontend) | Alta | M | No |
| Diagrama de secuencia para registro de paciente (frontend) | Alta | M | Sí |
| Diagrama de secuencia para login de usuario (frontend) | Alta | M | No |

Cuadro 2.10: Backlog de tareas para sprint 10.

Dificultades encontradas:

Durante este sprint no se pudieron completar ninguna de las tares planificadas debido a un pico en la carga de trabajo dentro del trabajo del alumno. Como en otras ocasiones, al ser un riesgo que ya estaba contemplado se aplicó la acción correspondiente: aumentar la carga de horas en los sucesivos *sprints* y las tareas volvieron al *backlog* para ser planificadas en los siguientes *sprints*.

### Sprint 11

Debido al problema surgido durante el sprint anterior, el entregable para este sprint es el mismo: añadir el *feedback* aportado por el *product owner* a los diagramas de secuencia para el

servidor, así como realizar los diagramas de secuencia para el *frontend*.

Backlog priorizado en la tabla 2.11

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Diagrama de secuencia para registro de ejercicio (backend) | Alta | S | Sí |
| Diagrama de secuencia para registro de desarrollador (backend) | Alta | S | Sí |
| Diagrama de secuencia para registro de paciente (backend) | Alta | S | Sí |
| Diagrama de secuencia para login de usuario (backend) | Alta | S | Sí |
| Diagrama de secuencia para registro de ejercicio (frontend) | Alta | M | Sí |
| Diagrama de secuencia para registro de desarrollador (frontend) | Alta | M | Sí |
| Diagrama de secuencia para registro de paciente (frontend) | Alta | M | Sí |
| Diagrama de secuencia para login de usuario (frontend) | Alta | M | Sí |

Cuadro 2.11: Backlog de tareas para sprint 11.

Dado que las tareas relacionadas con la realización de diagramas correspondientes al *backend* ya estaban avanzadas durante el *sprint* anterior permitió añadir las correcciones planteadas sobre las dudas y poder realizar dichos diagramas para el desarrollo del *frontend*.

### Sprint 12

Dentro de este *sprint* se estableció como entregable completar los diagramas restantes y aplicar mejoras sobre los ya presentados. También se estableció como prioridad finalizar el apartador de microservicios para no seguir arrastrándolo entre *sprints*.

Backlog priorizado en la tabla 2.12:

### Sprint 13

Backlog priorizado en la tabla 2.13.

Dificultades encontradas

Carga de trabajo ->mover al sprint siguiente

### Sprint 14

Backlog priorizado en la tabla 2.14.

Dificultades encontradas:

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Mejora de diagrama de componentes | Baja | S | Sí |
| Corrección de diagrama de modelo de dominio | Alta | S | Sí |
| Diagrama de despliegue | Media | M | Sí |
| Diagrama de modelo de base de datos | Alta | S | Sí |
| Diagrama modelo entidad relación | Baja | S | Sí |
| Apartado Microservicios | Alta | L | Sí |

Cuadro 2.12: Backlog de tareas para sprint 12.

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Añadir *sprints* 1-8 | Baja | S | No |
| Desarrollo de servicio de registro | Alta | L | No |
| Refactor dentro de Angular App | Media | M | No |
| Esqueleto de servicio para gestión de ejercicios | Alta | M | Sí |
| Apartado API REST | Baja | S | Sí |
| Apartado Patrón Repository | Baja | S | Sí |
| Apartado Microservicios | Alta | L | No |

Cuadro 2.13: Backlog de tareas para sprint 13.

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Apartado API REST | Baja | S | Si |
| Desarrollo de servicio de registro | Alta | L | No |
| Refactor dentro de Angular App | Media | M | No |
| Esqueleto de servicio para gestión de ejercicios | Alta | M | Sí |
| Apartado Patrón Repository | Baja | S | Sí |
| Apartado Microservicios | Alta | L | No |

Cuadro 2.14: Backlog de tareas para sprint 14.

### Sprint 15

Backlog priorizado en la tabla 2.15.

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Apartado arquitectura de datos | Alta | L | No |
| Apartado influencia del Lean Startup | Baja | S | Sí |
| Apartado Clean Code | Media | L | Sí |
| Apartado despliegue | Media | L | No |
| Mejora cap. Líneas de desarrollo futuras | Baja | S | Sí |
| Mejora cap. Conclusiones | Baja | S | Sí |

Cuadro 2.15: Backlog de tareas para sprint 15.

Dificultades encontradas:

### Sprint 16

En este *sprint* como entregable, dentro del proceso de documentación del proyecto, se estableció la entrega de

Backlog priorizado en la tabla 2.16

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Apartado arquitectura de datos | Alta | L | No |
| Apartado influencia del Lean Startup | Baja | S | Sí |
| Apartado Clean Code | Media | L | Sí |
| Apartado despliegue | Media | L | No |
| Mejora cap. Líneas de desarrollo futuras | Baja | S | Sí |
| Mejora cap. Conclusiones | Baja | S | Sí |

Cuadro 2.16: Backlog de tareas para sprint 16.

Dificultades encontradas:

### Sprint 17

Para este *sprint* el entregable planificado se compone de un la realización de diversos puntos que se habían quedado sin completar dentro de la documentación así como aplicar correcciones vistas con el *product owner* durante la reunión semanal.

Backlog priorizado en tabla 2.17

Dificultades encontradas:

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Apartado de calendarización | Alta | M | Sí |
| Apartado estado del arte | Baja | M | Sí |
| Mejora del cap. Planificación | Media | M | Sí |
| Añadir análisis de tecnologías a cap. Herramientas | Media | S | Sí |
| Diseño interfaz gráfica | Alta | XL | No |
| Apartado integración con Grafana | Media | M | No |
| Corrección sobre capítulo de conclusiones | Media | S | Sí |

Cuadro 2.17: Backlog de tareas para sprint 17.

La realización de la tarea para digitalizar los diseños de interfaz de usuario no pudo ser completada en su totalidad por lo que pasó al siguiente *sprint*. La integración de Grafana dentro de la máquina de demo tampoco pudo ser completada por lo que se planificó para finalizar en el siguiente *sprint*.

#### Sprint 18

En este *sprint* se marcó como objetivo resolver todas las correcciones planteadas por el *product owner* en en *sprint* anterior, con el objetivo de entregar una versión final de la misma. Así como finalizar aspectos como la integración con Grafana dentro del sistema

Backlog priorizado en tabla 2.18

| Tarea | Prioridad | Talla | Completada |
|---|---|---|---|
| Añadir diseño de interfaz gráfica | Alta | XL | Sí |
| Corrección sobre Lean Startup | Baja | S | Sí |
| Integración con Grafana | Alta | L | Sí |
| Apéndice A, Contenido del CD | Alta | S | Sí |
| Apéndice B, Guía de Despliegue | Alta | M | Sí |

Cuadro 2.18: Backlog de tareas para sprint 18.

### 2.4.2. Dificultades surgidas durante el proyecto

Una de las principales dificultades surgidas durante el proyecto para seguir la planificación establecida en el *sprint planning* es la necesidad de compaginar el desarrollo del proyecto con la vida laboral por parte del estudiante, ya que en semanas con picos puntuales de trabajo era imposible la realización de todas las tareas marcadas para esa iteración.

Este era uno de los riesgos detectados, por lo que en primer lugar, se trató de paliar el impacto producido por esta falta de tiempo con la acción preventiva planificada de tener un mayor número de horas libres al inicio del proyecto para agilizar su desarrollo y de manera conjunta con el plan de acción reestructurando las horas de trabajo para las sucesivas iteraciones.

Si bien es cierto que esta acciones han ayudado a paliar el impacto dentro del proyecto, la repetición sucesiva de ocasiones en las que era imposible dedicar tiempo al proyecto han provocado que el plan de acción perdiera impacto a la hora de paliar la desviación que se producía sobre la planificación.

## 2.5. Presupuesto

A la hora de realizar un proyecto, es vital analizar el coste que conlleva su desarrollo, ya que, si no supera el beneficio obtenido por el mismo, no tendría sentido su desarrollo.

En la realización de este proyecto, hay diversos elementos que generan un coste sobre el desarrollo, como por ejemplo el material utilizado, las instalaciones, licencias y el propio gasto derivado de las horas de trabajo. A este conjunto de gastos, en un entorno real, se podría aplicar también el gasto en formación o de transporte que incrementaría aún más este valor.

Hay que prestar especial atención al control de gastos dentro de un proyecto, ya que, un exceso de gasto, puede condenar a un buen proyecto a la ruina o ser cancelado por el equipo directivo de la compañía, pero por otro lado, un déficit en el gasto, puede provocar malestar dentro del equipo de desarrollo o el uso de equipos y licencias anticuadas que retrasen el tiempo de desarrollo.

Para el calculo del coste nos hemos basado en los siguientes parámetros:

- Coste de horas de trabajo: La realización de este proyecto está establecida en 300 horas. Para estimar este coste nos hemos basado en un valor actual del mercado laboral.
- Coste de licencias: Es la suma del coste de todas las licencias de software utilizadas durante el desarrollo
- Coste de instalaciones: Es el coste derivado de alquiler de instalaciones así como agua, luz e impuestos asociados
- Coste de infraestructura: El código debe correr sobre un hardware, ya sea en la nube o sobre una infraestructura propia. Este hecho genera unos gastos sobre el proyecto.
- Coste de amortización de equipos: El coste derivado del uso de los equipos informáticos y periféricos utilizados durante el desarrollo.

$Coste_{Astah} = 12,50€/mes \cdot 5 = 25€$
$Coste_{IntelIJ} = 49,90€/mes \cdot 5 = 250€$
$Coste_{Licencias} = Coste_{Astah} + Coste_{IntelIJ} = 275€$

$Coste_{Ordenador} = 2100€$

$Coste_{Pantalla} = 200€$

$Coste_{Equipos} = Coste_{Ordenador} + Coste_{Pantalla} = 2300€$

$Coste_{Horas} = 300 \cdot 42{,}5 = 12750€$

$Coste_{Alquiler} = 200€/mes \cdot 5meses = 1000€$

$Coste_{Infraestructura} = 0,0928€/hora \cdot 300horas = 27,84€$

$Coste_{Total} = Coste_{Licencias} + Coste_{Horas} + Coste_{Alquiler} + Coste_{Infraestructura} + Coste_{Equipos} = 16390,34€$

## 2.6. Costes

Dentro de esta sección, se detallan los costes reales que se generaron durante el desarrollo del proyecto. También se realiza un análisis de los posibles valores que se han desviado de la estimación prevista en el presupuesto.

### Coste de licencias

En este apartado, la cuantía destinada ha aumentado porque en el presupuesto se estimó que solo sería necesario disponer de la licencia de Astah durante dos meses, pero debido a que se han tenido que ir realizando mejoras sobre los diagramas a lo largo de distintas fases del proyecto, ha sido necesario disponer de ella durante los 5 meses del proyecto.

El coste que implica el uso de la licencia Astah no es un coste real, ya que se está utilizando la licencia académica proporcionada por la Universidad de Valladolid, pero resulta un valor de interés para conocer el coste que tendría realizar este proyecto fuera del ámbito educativo.

$Coste_{Astah} = 12,50€/mes \cdot 5 = 62,5€$

$Coste_{IntelIJ} = 49,90€/mes \cdot 5 = 250€$

$Coste_{Licencias} = Coste_{Astah} + Coste_{IntelIJ} = 312,5€$

Esta modificación supuso un incremento del 13,64 % sobre el valor planificado

### Coste de equipos

Dado que dentro de este apartado se estima el coste de los equipos utilizados para realizar el proyecto y la duración del mismo no requiere un cambio de equipos por la degradación de sus características, el valor de este campo permanece igual que en la sección de Planificación.

$Coste_{Equipos} = Coste_{Ordenador} + Coste_{Pantalla} = 2300€$

### Coste de alquiler

Del mismo modo que ocurre con el valor calculado para los equipos, este valor no varía, ya que, a pesar de que el número de horas ha aumentado, este cálculo se realizó calculando el mes completo.

$Coste_{Alquiler} = 200€/mes \cdot 5meses = 1000€$

### Coste de horas

Este valor ha sido modificado ya que se han dedicado un mayor número de horas de las planificadas para realizar el proyecto. Concretamente, el aumento de horas frente a las previstas, es de 64 horas.

$Coste_{Horas} = 364 \cdot 42{,}5 = 15470€$

El incremento de coste sobre la cifra dada en el capítulo de planificación es del 21 % con respecto al conjunto de horas que se habían previsto.

### Coste de infraestructura

El coste estimado para infraestructura ha sido menos del planificado inicialmente, ya que durante una primera etapa del proyecto se realizó el despliegue en local para minimizar el consumo de recursos y solo en la última etapa cuando era necesario la interacción entre todos los componentes sobre el entorno de demo.

$Coste_{Infraestructura} = 0,0928€/hora \cdot 80horas = 7,43€$

La reducción del coste con respecto al valor presupuestado es del

Según los factores expuestos anteriormente el valor final del desarrollo del proyecto es de:

$Coste_{Total} = Coste_{Licencias} + Coste_{Horas} + Coste_{Alquiler} + Coste_{Infraestructura} + Coste_{Equipos} = 16390,34€$

# *Capítulo 3*

# Herramientas

Para dar una solución al problema planteado es necesario revisar lo que está siendo usado por la industria actualmente y ver cuál es la tecnología que mejor se adecúa al problema.

Antes de definir el resto de herramientas usadas para el proyecto es necesario determinar el lenguaje de programación que será utilizado.

Actualmente, en la industria, para el desarrollo de plataformas online hay varios lenguajes destacados, ya que se ajustan a las necesidades que se presentan al construir una plataforma de este tipo.

La página *Stackoverflow*, una de las páginas de referencia dentro de la informática, en la encuesta realizada entre todos los desarrolladores que forman su comunidad, extrajo la gráfica mostrada en la Figura 3.1

Basándonos en estos datos y realizando una búsqueda más detallada sobre qué lenguajes son utilizados para construir plataformas que den respuesta a los problemas planteados en este proyecto son los siguientes:

- **Java**, es un lenguaje de programación de propósito general, concurrente y orientado al objeto diseñado para tener pocas dependencias de implementación. Su intención es que el código se escriba una vez y se ejecute en cualquier dispositivo. Las aplicaciones desarrolladas en este lenguaje son compiladas a bytecode y puede ejecutarse en una *Java Virtual Machine* (JVM) sin tener importancia la arquitectura de la maquina subyacente. Este lenguaje se convirtió desde 2012 en uno de los más populares en uso, especialmente para aplicaciones de cliente-servidor web. La incorporación del recolector de basura permite que los desarrolladores deleguen en el entorno durante el tiempo de ejecución, la gestión de la memoria. Este lenguaje es ampliamente utilizado para el desarrollo de código en el servidor, aplicaciones de escritorio, sistemas de big data, etc. La industria proporciona un gran soporte a este lenguaje y varias de las grandes compañías dentro de la industria lo tienen dentro de su *tech plan*. Por ejemplo, Google lo tiene como uno de sus tres lenguajes de desarrollo. Importantes herramientas como Kafka o Apache Server están programadas con este lenguaje.

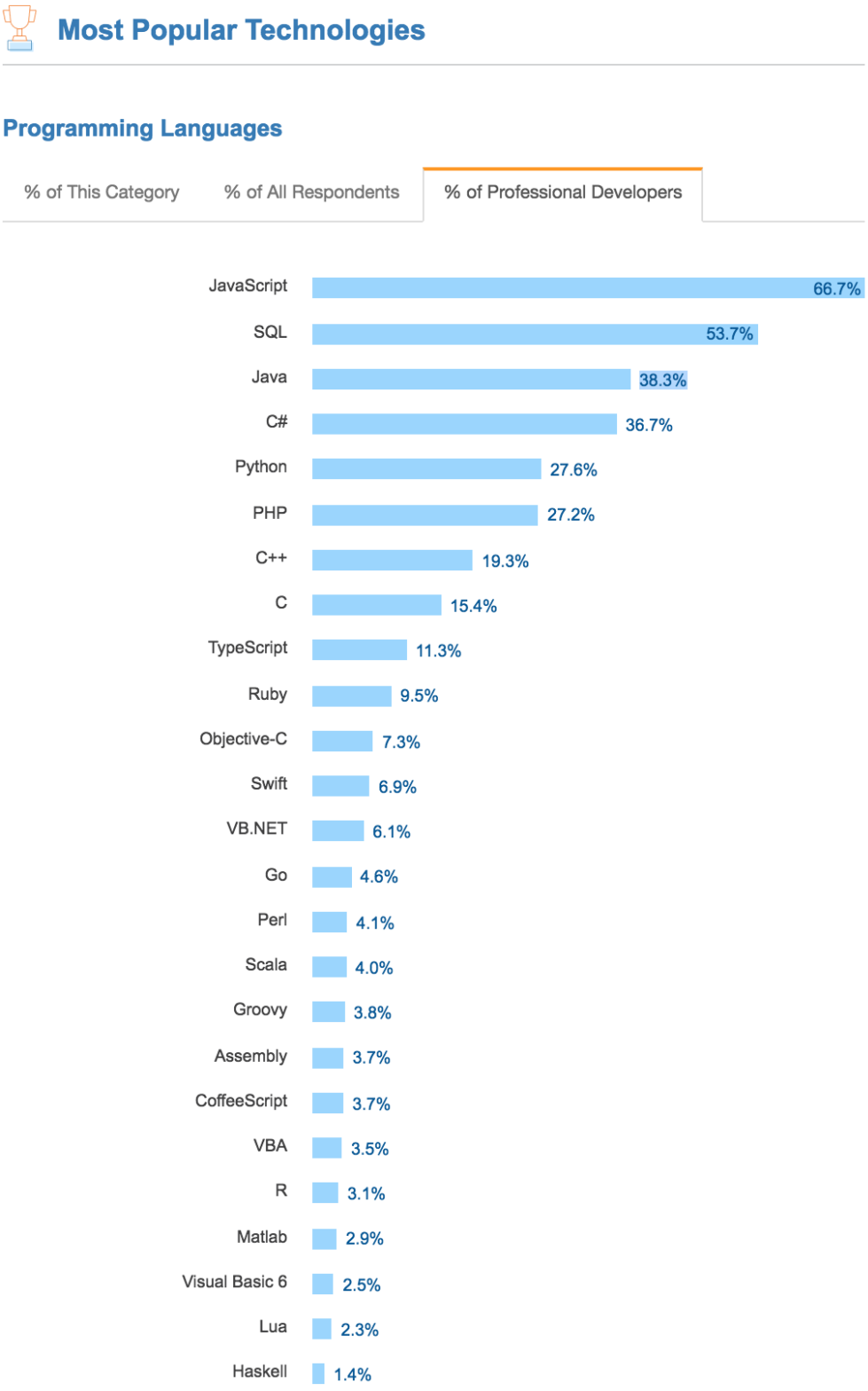


Figura 3.1: Tecnologías más valoradas por los desarrolladores en 2017. Tomado de [14]

- **Scala**, este lenguaje de alto nivel programación permite la programación orientada al objeto así como la funcional.

  Pero tiene especial sentido cuando se usa en su faceta puramente funcional, eliminando los efectos colaterales y haciendo así un código más predecible y fácil de probar. La implementación nativa de un concepto como las promesas, facilita el desarrollo de código asíncrono y su tipado estático es útil para evitar *bugs* durante el desarrollo. Actualmente en la industria, este lenguaje tiene un uso mayoritario para trabajar con grandes volúmenes de datos, ya que tiene una clara sinergia con el *framework* Apache Spark y porque, al incorporar dentro del lenguaje los *Streams* con sus operaciones *map, reduce y filter*, es muy útil para interactuar con grandes colecciones.

- **Ruby on Rails**, el lenguaje de programación se trata de Ruby, pero dentro de la industria no se entiende el uso de Ruby sin su *framework*, Ruby on Rails. Su uso está centrado en la creación de aplicaciones como servidor. Este *framework* sigue el patrón MVC y busca permitir crear grandes aplicaciones minimizando las líneas de código y la configuración necesaria, este principio ha recibido el nombre de *Convention over Configuration*. Una de las bases de su filosofía es el principio DRY (*Don't repeat yourself*). Ruby permite la metaprogramación y al hacer uso de esta propiedad Rails genera un código más legible. Publicando su primera *release* en 2005 ha ido ganando influencia dentro de la industria y grandes compañías de la industria hacen uso de este lenguaje como Github, Zendesk, SoundCloud o Airbnb.

- **PHP**, se trata de un lenguaje de programación de propósito general pensado para ser ejecutado en el lado del servidor, en el desarrollo web de contenido dinámico. Este lenguaje es ampliamente usado, está instalado en más de 20 millones de sitios web y en un millón de servidores. Al tener un gran parecido con lenguajes de programación estructurada reduce la curva de aprendizaje para los crear aplicaciones complejas. Uno de los usos más habituales de este lenguaje es para el desarrollo de módulos dentro del *framework Wordpress*. A pesar de que ha perdido popularidad entre los desarrolladores, los números muestran que sigue habiendo una gran parte de la industria manteniendo sistemas y desarrollando con él.

- **Javascript**, es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Su uso está centrado principalmente en torno a las tecnologías web, ya que permite ser usado tanto en la parte del front-end como en el back-end. Pero su amplio uso y su extensa comunidad han hecho crecer a este lenguaje y extenderlo para ser usado incluso como base para la programación de redes neuronales y sistemas de aprendizaje guiado. En un primer momento concebido para ser usado como un lenguaje orientado al objeto, en recientes versiones del lenguaje se soportan elementos de la programación funcional. Su tipado débil unido a la reducida curva de aprendizaje necesaria para conseguir software

funcional ha hecho granjearse la fama a este lenguaje de difícil de mantener. El contar con varios *frameworks* como Nodejs y AngularJS y su sistema de librerías npm (mantenido por una amplia comunidad) ha contribuido también a su rápida adopción por una gran parte de la industria.

- **Python**, este lenguaje de programación interpretado utiliza una sintaxis que de lugar a un código legible. Este lenguaje, como algunos de los descritos previamente es multiparadigma, dado que soporta la programación orientada al objeto, la programación imperativa Tiene un uso muy extendido entre los DevOps, ya que permite la creación de *scripts* muy legibles y cuenta con muchas funcionalidades que no requieren la descarga de dependencias y que facilitar el tratar con ficheros o con objetos tipos JSON.

Por las características que presenta y el conocimiento previo del alumno en el lenguaje, se ha elegido Java para el desarrollo del proyecto. En su versión 8 se incorporan diversas características como una nueva API de Date y Time y la codificación y decodificación en base64 de forma nativa o mejoras de rendimiento y seguridad sobre la JVM. Pero las características que han dirigido la decisión son la incorporación de las expresiones lambdas ya utilizadas en otros lenguajes como Javascript o Scala y los Streams (una abstracción que aprovecha las arquitecturas de núcleos múltiples sin necesidad de programar líneas de código multiproceso permitiendo así procesar datos de modo declarativo) que orientan el lenguaje hacía un modo más funcional y permiten dentro de la programación orientada al objeto hacer un código más expresivo, menos extenso y más legible.

La elección de Java 8 para el desarrollo de la plataforma abre un amplio abanico de *frameworks* usados por la industria para el desarrollo de software de manera ágil, ejemplos de ello son:

- **Play Framework**, es un framework open-source diseñado para el desarrollo de aplicaciones web y permite su desarrollo en Scala o Java. Su arquitectura está basada en el patrón modelo-vista-controlador (MVC). Esta centrado en optimizar el tiempo necesario para hacer un software productivo usando el paradigma de diseño de software *Convention over configuration*, mediante el cual se minimiza el número de decisiones que el desarrollador tiene que tomar sin perder en el camino el *framework* flexibilidad. Desde su origen Play framework está diseñado para ser asíncrono y *RESTful*, como otros framewoks en el mercado cuenta con una arquitectura modular.

- **Spring Framework**, es un *framework* que tiene como principio la inversión de control y que se ejecuta sobre la Java Virtual Machine. Para el desarrollo sobre Spring Framework, se pude elegir entre Java, Kotlin y Groovy. Un gran funcionalidad que aporta este *framework* es que cuenta con un amplio abanico de módulos que pueden ser añadidos al proyecto de forma sencilla abstrayendo complejidad y agilizando el desarrollo. Como se ha comentado antes, Spring Framework tiene como corazón el principio de inversión de control, esto

se materializa en un contenedor de inversión de control, un componente encargado de gestionar los objetos y sus relaciones dentro de la aplicación, para la obtención de estos objetos, el contenedor se sirve de la inyección de dependencias. Dado el uso extensivo que tiene este *framework* dentro de la industria, ya que fue lanzando comercialmente en 2003, cuenta con una amplia comunidad y una completa documentación. Recientemente ha sido liberada su última versión estable que incorpora mejoras basadas en la implementación del patrón *reactive*.

- **Eclipse Vert.x**, es un *framework* basado en el paradigma *event-driven* y que permite el desarrollo en diferentes lenguajes (Java, JavaScript, Groovy, etc)y se ejecuta sobre la Java Virtual Machine. Tiene un modelo de ejecución similar al que implementa Node.js, todo el código se ejecuta sobre un mismo hilo no bloqueante. El desarrollo para este *framework* es totalmente asíncrono lo que permite el crear aplicaciones escalables y no bloqueantes. Al utilizar el *actor model* y disponer de repositorio público, facilita la reutilización del código. Esta especialmente orientado hacia el desarrollo basado en Microservicios.

En nuestro caso elegiremos Play Framework, dado que nos da la posibilidad de crear una prueba de concepto de manera ágil y sin sacrificar flexibilidad en el camino. El desarrollo del proyecto dentro de este nuevo framework es una oportunidad para experimentar y ampliar el conocimiento. También el uso de este framework por empresas como Samsung, EA, LinkedIn o Telefonica, demuestra que el conocimiento de este *framework* es valorado dentro de la industria.

## 3.1. Herramientas para la gestión del proyecto

### 3.1.1. Pivotal tracker

Herramienta de gestión de tareas en tiempo real, que utiliza la metodología Kanbam. Permite la creación de historias de usuario, *epics* y tareas. Así como una priorización de las tareas y otorgarles un peso determinado. De este modo, resulta más sencillo la planificación de los *sprints* y el seguimiento del estado de las tareas.

La elección de esta herramienta se debe a varios factores: En primer lugar, es gratuita para proyectos de hasta 3 miembros, también es un servicio alojado en la nube por lo que no requiere instalación o configuración previa.

Otra funcionalidad interesante es la posibilidad de determinar la *velocity*, es decir, cuantos puntos de funcionalidad se es capaz de acometer por sprint, para de esta manera evitar sobrecargar con tareas que en base a las métricas de anteriores iteraciones no se podrán cumplir.

Por contrapunto, en esta herramienta no se pueden establecer *User Stories* y visualizar las tareas asociadas a ella dentro de un sprint, lo cual resta un poco de visibilidad a la hora de agrupar tareas en puntos de acción y dificulta mantener el foco para terminar una *User Story*. Esto se

debe a que la herramienta hace énfasis en *features* y la adaptación propuesta por la herramienta es asimilar una *feature* a una *user history*.

Si bien es cierto, que esta herramienta gana especial fuerza cuando se trabaja en equipo.

### 3.1.2. Git

Herramienta de control de versiones diseñada por Linus Torvalds, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Pero en caso de trabajar individualmente también resulta de utilidad ya que cualquier cambio que haya sido registrado en el repositorio. Permite también trabajar en base a ramas, en las cuales podemos desarrollar distintas funcionalidades en paralelo.

Git permite tener un repositorio para trabajar localmente y poder conectarnos en caso de ser necesario con otros servicios como Github o GitLab.

### 3.1.3. Gitlab

Servicio web para control de versiones y desarrollo de software colaborativo basado en Git. Esta herramienta permite administrar el código de los distintos proyectos que se alojan en este servicio.

El servicio puede ser alojado dentro de la propia organización o usarse a través de internet en los servidores de la aplicación.

En su origen este servicio web ofrecía la posibilidad de manejar distintos proyectos y usuarios. Así como ver los *commits* realizados por un usuario o sobre un archivo. También permitía visualizar archivos y los cambios realizados sobre ellos en un formato de historial. Una funcionalidad añadida de este producto es la posibilidad de gestionar proyectos, ofreciendo un tablero Kamban en el que poder introducir tareas y asignarlas a usuarios.

Hasta este punto, nada que no ofrezcan otros *players* del mercado como puede ser Github, pero con el crecimiento del producto y con el afán de buscar un elemento diferenciador, dentro del producto se ofrecen otras funcionalidades extra que no encontramos en otros competidores, o por lo menos tan integradas dentro del producto y tan amigables para el usuario.

Las funcionalidades se agrupan por soluciones y cada solución equivale a una fase dentro de el ciclo de vida de *DevOps* o a un atributo de calidad de GitLab. Estas funcionalidades incluyen un tablero Kanban en el que añadir tareas y configurar distintos parámetros como la persona asignada o la duración de la tarea, la posibilidad de publicar las imágenes Docker en un registry privado ofrecido por la propia herramienta o integrar todo el ciclo de Integración Continua y Entrega Continua (CI/CD) sin necesidad de instalar ningún plugin.

Gitlab cuanta con versión gratuita sin límite de repositorios privados y con una versiones Enterprise. Estas versiones de pago están divididas en 3 niveles, tanto para la aplicación web

como para la alojada en la organización. Cada versión tiene un precio superior a la anterior e incorpora más funcionalidades, como por ejemplo soporte 24/7 para el servicio alojado.

La versión del servicio usada para el proyecto se corresponde con la versión educativa proporcionada por la escuela que cuenta con recursos Enterprise. Este Gitlab, donde se encuentra alojado este proyecto, es en gitlab.inf.uva.es. Las credenciales para acceder al Github de la escuela son las proporcionadas por la escuela para acceder a los laboratorios.

Está basada en Ruby y utilizado por organizaciones como el CERN o la NASA. Se trata de una alternativa para la gestión de código frente a otras herramientas como Github, más centrada exclusivamente en el código.

## 3.2. Herramientas para el Desarrollo

### 3.2.1. Play Framework

En la parte inicial del capítulo ya se presentó brevemente Play Framework como una de las alternativas en el desarrollo. El objetivo de este apartado es profundizar más en la herramienta.

Play Framework es un framework open source para el desarrollo de aplicaciones web de alto rendimiento basadas en Scala o Java. Cuenta con un gran número de APIS y componentes para agilizar el desarrollo de dichas aplicaciones.

Play Framework surge como respuesta a los java web framework tradicionales y trata de implementar una mejor solución a los problemas asociados a estos frameworks, debido a ello Play Framework tiene como base la optimización de los recursos, ser ligero, sin estado y una arquitectura *web-friendly* para aplicaciones que busquen una alta escalabilidad.

El corazón de Play Framework es puramente asíncrono al estar centrado en desarrollo para aplicaciones web [8]. Play framework sigue un modelo reactivo y se apoya sobre el API de Akka Stream [9].

Otro de los principios de Play Framework es la flexibilidad. Un ejemplo de ello es la integración con otros servicios, como por ejemplo bases de datos, donde no se limita a un sistema de persistencia concreto o un conjunto reducido, sino que ha realizado desarrollos para dar soporte a varios de estos sistemas como MySQL, Postgres, etc.

Este framework está pensado para ser usado con lenguajes de la JVM (Java o Scala), ofreciendo ejemplos y documentación para ambos lenguajes y la mayor parte de su desarrollo interno en Scala.

Cuenta con una comunidad muy activa que realiza constantemente aportaciones para que el framework siga evolucionando. Otro aspecto importante es que cuenta con un amplio abanico de ejemplos prácticos sobre cómo afrontar y resolver problemas que pueden surgir durante el desarrollo de una aplicación web.

Play Framework permite la construcción de prototipos completamente funcionales en un muy poco tiempo. Lo cual lo hace ideal para pruebas de concepto y modelos ágiles en los que el

proyecto no para de evolucionar.

### 3.2.2. Angular

Angular es una plataforma y framework para crear aplicaciones cliente en HTML y Typescript. Permite desarrollar aplicaciones que serán ejecutadas sobre un navegador web, en un móvil o en un escritorio.

Se trata de un framework de código abierto mantenido por Google. Busca simplificar el desarrollo y las pruebas sobre estas aplicaciones cliente implementando el patrón Modelo Vista Controlador.

Está centrado en el desarrollo de *Single Page Application*, pero puede usarse para el desarrollo de cualquier elemento web.

La estructura que aplica sobre los proyectos, haciendo una división por componentes en lugar de páginas completas, ayuda a evitar la repetición de código y hace el desarrollo más modular.

Las características que proporciona el framework son:

- **Velocidad y rendimiento**, el código generado está optimizado para las máquinas virtuales Javascript y puede ejecutarse en cualquier servidor Node, PHP o .NET.

- **Productividad**, la velocidad en el desarrollo es uno de los principales focos de atención para este framework. Por ello cuenta con Angular CLI, la herramienta por linea de comandos que permite arrancar un servidor y ver los cambios en tiempo real, así como ejecutar los test con cada compilación. También los principales editores de código cuentan con plugins para la sintaxis y el auto completado.

- **Testing**, permite utilizar Karma para pruebas unitarias y Protractor para realizar pruebas end-to-end

Angular y React se han convertido en las opciones por defecto para el desarrollo de aplicaciones web en el ambito empresarial.

### 3.2.3. Grafana

Es una plataforma open source para la visualización de datos a través de gráficas. Grafana es la plataforma por excelencia utilizada para mostrar series temporales.

Cuenta con la capacidad de exportar datos desde un amplio número de fuentes: Prometheus, Postgres, etc. Permite mostrar datos alojados en el servicio de CloudWatch de Amazon, conectar una base de datos para mostrar directamente los datos o mostrar los métricas del servicio a través de *queries* mediante Prometheus. Por este motivo encaja a la perfección con una arquitectura basa en microservicios, en la cual podemos tener alojados los datos en distintos sistemas y verlos en conjunto de una manera rápida.

El uso de esta herramienta fomenta el recabar datos, para visualizarlos, entenderlos y tomar decisiones basadas en dichos datos.

Grafana también aporta la posibilidad de crear alertas en función de los valores que se están recogiendo.

Otro punto fuerte de la herramienta es que cuenta con un amplio número de extensiones que pueden ser añadidas a los *dashboard* para mejorar la experiencia, desarrollados por la comunidad entorno al proyecto. Así como la amplía comunidad que soporta el proyecto.

### 3.2.4. PostgreSQL

PostgreSQL es un potente sistema de base de datos relacional. Es un proyecto open source con una trayectoria de más de 15 años y usado por empresas de alto éxito como CartoDB o Telefónica.

Uno de los principales motivos para la elección de este sistema de base de datos por encima de otros, es que está diseñado para su uso en entornos con un alto volumen de datos. Si bien en una fase inicial, este volumen de datos nunca se dará, diseñar la plataforma con vistas a dicho crecimiento es una decisión acertada.

Otro motivo es su amplio uso en el mundo laboral, como se ha mencionado previamente, y la oportunidad para investigar y profundizar sobre su uso.

### 3.2.5. PgAdmin

Interfaz gráfica de usuario multiplataforma para la interacción con PostgreSQL. Es la opción mayoritaria entre los usuarios de Postgres, ya que ofrece una gran funcionalidad:

- Visualización de rendimiento en tiempo real, lecturas y escrituras sobre la base de datos.
- Generación de script SQL al seleccionar una tabla o toda la base de datos.
- Realización de *queries* desde la propia herramienta contra la base de datos
- Visualización en tiempo real sobre el contenido de las tablas.

Permite conectar con varias bases de datos al mismo tiempo. Otro aspecto interesante, es que posee imagen Docker y podemos lanzarla en nuestros equipos solo con tener instalado Docker, sin llenar el equipo de dependencias.

### 3.2.6. Swagger

Swagger es el framework compuesto por un conjunto de herramientas para trabajar en todas las etapas del ciclo de vida de un API REST.

Esta conjunto de herramientas es complementaria para el desarrollo del código. Pero resulta de gran utilidad ya que muestra de una manera muy visual la definición de un API REST, con

los endpoints que pueden ser atacados, o los objetos que devuelve dicho endpoint (aportando incluso ejemplos).

Dentro del proyecto nos centraremos en dos de estas herramientas:

- **Swagger Editor**, este herramienta permite escribir la especificación en formato yaml o json del API REST y ver en tiempo real cuál es el resultado. También incorpora un validador, para evitar fallos de sintaxis en la formación del yaml o json.

- **Swagger UI**, permite la exposición de un *endpoint* que contiene la definición del API. Esta herramienta resulta de gran utilidad cuando el servicio está abierto al uso por parte de terceros.

Si añadimos dentro de la definición del API REST un endpoint de un servidor arrancado que contenga el código que se expone, los desarrolladores que consulten la definición del API en el Swagger UI pueden realizar pruebas reales hacia el servidor.

Por último, este conjunto de herramientas también ofrece una utilidad, Swagger Codegen,para generar el código cliente y servidor en base a la definición del API REST. Este servicio debido a su complejidad cuenta con un gran número de *bugs* y está siendo desarrollado de manera constante por la comunidad.

### 3.2.7. Docker

Docker es un proyecto open source que permite automatizar el despliegue de apelaciones dentro de contenedores software. Docker permite ejecutar procesos de manera aislada y aprovechando recursos del sistema operativo.

A diferencia de una máquina virtual, los contenedores no requieren incluir un sistema operativo adicional. En lugar de ello, se apoya en funcionalidades del kernel de Linux como *cgroups* o *namespaces* para aislar la vista de una aplicación del sistema operativo.

Usando estos contenedores, los recursos son aislados, los servicios son restringidos y permite ver el proceso como cualquier otro corriendo en la máquina, con su propio identificador, interfaces de red y sistema de archivos. Todos los recursos usados por un contenedor puede restringirse.

Los comandos necesarios para generar una imagen, así como la imagen base, se registran en un fichero denominado Dockerfile. Cuando se construye una imagen por primera vez, Docker almacena en una caché interna los resultados de esta primera ejecución. De este modo, si se vuele a generar la imagen, la construcción será mucho más rápida.

Las imágenes Docker se pueden publicar en registros, lo que facilita el que otros usuarios puedan acceder a ellas. Mediante una estrategia de etiquetas, se pueden mantener distintas versiones de una misma imagen.

### 3.2.8. Kubernetes

Kubernetes es un proyecto open source que permite automatizar los despliegues, contribuye a la escalabilidad del proyecto y facilita la gestión de aplicaciones que corren dentro de contenedores.

Se trata de uno de los principales software de orquestación del mercado conjunto con Docker Swarm y Mesos. Un software de orquestación permite delegar toda la comunicación entre contenedores, balanceo de carga y operación de los recursos.

Para ejecutar Kubernetes es necesario crear un cluster compuesto de sus unidades atómicas, los Pods, que pueden componense de varios contenedores. Estos Pods corren dentro de nodos físicos.

Kubernetes define sus recursos dentro de ficheros yaml y estos ficheros se determina el estado deseado de un recurso. Es decir, si queremos tres instancias de determinado servicio corriendo en todo momento en nuestra aplicación, en el momento en el Kubernetes detecte que una de las réplicas no esta sana, la matará y generará una nueva.

Kubernetes también nos permite delegar el balanceo de carga entre las distintas instancias de un servicio.

### 3.2.9. Intellij IDEA

Intellij es un editor diseñado especialmente para el desarrollo con Java. Cuenta con un gran número de plugins para el desarrollo con Java y la JVM.

Esta herramienta realiza un análisis sobre el código, permitiendo la navegación entre distintos puntos del proyecto, así como recomendaciones sobre el código que se está desarrollando.

Pone al desarrollador en el foco de trabajo por lo que ofrece un gran número de funcionalidades, atajos de teclado y todo el *workflow* está orientado al desarrollo.

### 3.2.10. Visual Studio Code

Visual Studio Code es en editor de código desarrollado por Microsoft pensado originalmente para el trabajo con Typescript, Javascript, HTML y CSS. Incorpora una integración directa con Git, lo que permite añadir ficheros y realizar commits sin salir del propio editor.

También cuenta con todo un ecosistema de plugins que aportan gran funcionalidad al editor permitiendo, por ejemplo, trabajar en otros lenguajes o desencriptar ficheros. Al igual que IntelIiJ IDEA cuenta con un gran número de atajos de teclado.

## 3.3. Herramientas para Testing

### 3.3.1. JUnit

JUnit es el Framework por excelencia para hacer pruebas unitarias sobre aplicaciones Java. Este framework ha sido importante en el desarrollo de la metodología *Text-driven development*. Sus creadores son Erich Gamma y Kent Beck.

JUnit permite la ejecución de métodos de clases Java de manera controlada, evaluando si los valores obtenidos en dicha ejecución son los esperados. JUnit proporciona anotaciones para determinar un método como un Test, también proporciona los *assert statements* que permiten comprobar que el valor obtenido es igual al esperado.

Los principales editores como NetBeans, Eclipse o IntelIiJ IDEA cuentan con plugins que permiten trabajar con este framework.

## 3.4. Herramientas para Documentación

### 3.4.1. ShareLatex

Editor colaborativo online para la creación de documentación. El lenguaje utilizado por esta plataforma para generar el texto es LaTeX. Se trata de un conjunto de macros de TeX (creado por Donald Knuth) creado por Leslie Lamport en 1984 y busca la creación de documentos escritos que presenten una alta calidad tipográfica.

Al tratarse de una plataforma online permite trabajar desde cualquier punto y realizar las compilaciones de LateX sobre la plataforma, lo que evita la instalación de paquetes y problemas asociados.

Esta plataforma cuenta con más de 1.000.000 de estudiantes y académicos en universidades como el MIT, University of Cambridge o la universidad de Valladolid.

### 3.4.2. Astah

Se trata de una herramienta para el diseño de software. Principalmente esta centrada en el desarrollo de diagramas, aunque como se expone a continuación, en su versión profesional incorpora un conjunto de funcionalidades muy útiles.

Cuenta con dos ediciones, una edición llamada Astah community, que permite el desarrollo de diagramas para versión UML 2.x. Por otra parte, una versión de pago llamada Astah profesional, que incorpora un mayor número de diagramas además de los diagramas UML 2.x como son diagramas de flujo o diagramas de entidad relación.

Dentro de la versión profesional, también están incluidas dos funcionalidades realmente útiles como es la generación de código a partir de diagramas y un control de versiones, para poder gestionar el cambio sobre distintos diagramas. Estas dos funcionalidades hacen a esta

herramienta realmente útil ya que permite un desarrollo fluido de los diagramas e incluso puede ahorrar tiempo dentro del desarrollo usando el código auto generado.

El uso de esta herramienta durante distintas asignaturas de la carrera ha influido en la elección de la misma para el desarrollo de los diagramas para representar el modelo del dominio y los diagramas de secuencia que explican la realización en diseño de las *user stories*.

Para el desarrollo de este proyecto, se ha utilizado la licencia de estudiantes que permite el uso de manera gratuita para estudiantes de la versión profesional.

### 3.4.3. Draw.io

Herramienta online para crear diagramas similar a Astah pero con una amplia gama de estereotipos. Esta herramienta posee una amplia variedad de diagramas como diagramas de flujo, diagramas UML, de entidad relación o de diagramas de red. Proporciona la posibilidad de añadir una extensión dentro del navegador Google Chrome, para utilizar la página en modo *stand-alone*. Utilizando Draw.io podemos compartir de manera ágil diagramas dentro de un equipo de software y editarlos de manera compartida.

Esta herramienta ha sido utilizada para los diagramas del modelo de datos dentro del sistema de base de datos.

### 3.4.4. Balsamiq Mockup

Esta herramienta permite diseñar interfaces para aplicaciones web y aplicaciones móviles. Es muy ágil y está especialmente pensada para ser usada dentro de las primeras fases de la aplicación, cuando lo que prima es poder obtener un prototipo e iterar lo más rápido posible.

Tiene un estilo *cartoon* pero esto no implica que la cantidad de elementos proporcionados para el diseño de las interfaces sea reducido.

También ofrece la posibilidad de compartir entre varios miembros de un equipo los diseños, permitiendo que los clientes aporten *feedback* muy rápido y de manera sencilla.

Se trata de una aplicación de pago, pero ofrece una versión de prueba de 30 días. Para un uso puntual, usar la versión de prueba puede ser suficiente, pero para una compañía es muy rentable la elección de esta aplicación para su uso prolongado en el tiempo.

*Capítulo 4*

# Requisitos y Análisis

## 4.1. Historias de Usuario

En esta sección, se describen las historias de usuario consideradas para la planificación del proyecto y el desarrollo del mismo.

Se han divido las historias de usuario en función del usuario al que están destinadas, dado que hay un gran número de ellas que aplican para varios tipos de usuarios, se ha tomado una de ellos como ejemplo he indicado dentro de la misma que aplica para un conjunto de estos usuarios.

### 4.1.1. Usuario

Dado que esta historia de usuario afecta tanto a pacientes, desarrolladores e investigadores se toma la figura del usuario, la cual es una generalización de todas las anteriores, para que aplique a todas ellas y minimizar la repetición.

#### Inicio de sesión dentro de aplicación

Descripción de la historia de usuario de inicio de sesión:
Como usuario.
Para poder acceder a los servicios ofrecidos por la plataforma.
Quiero poder iniciar sesión dentro del sistema.
**Escenario: Inicio de sesión con éxito**
Cuando introduzco mis datos en la pantalla de inicio de sesión como se muestra en la tabla 4.1

Entonces en función del tipo de usuario que sea, la plataforma presentara un tipo de información concreta:

- Para desarrolladores y pacientes, tras iniciar sesión accedes al conjunto de ejercicios disponibles.

| Campo | Valor |
|---|---|
| Email | johnSmith@gmail.com |
| Contraseña | Long8 |

Cuadro 4.1: Inicio de sesión exitoso.

- Para investigadores, accedes al conjunto de información presentada en forma de gráficos y tablas dentro de *dashboards*.

**Escenario: Inicio de sesión sin éxito**

Cuando los datos introducidos en la pantalla de inicio de sesión no coinciden con ningún usuario registrado en el sistema como se puede ver en la tabla 4.2

| Campo | Valor |
|---|---|
| Email | userNotInSystem |
| Contraseña | Passwd87 |

Cuadro 4.2: Fallo en el inicio de sesión cuando el usuario no está registrado en el sistema.

Entonces debería ver un mensaje de error en la página de inicio de sesión que me indique que el usuario o contraseña no son correctos. Este mensaje es común a todos los errores presentados durante el inicio de sesión con intención de no dar información extra sobre datos reales de un usuario ante un posible atacante.

Cuando los campos de email 4.3 o de contraseña están vacíos 4.4.

| Campo | Valor |
|---|---|
| Email | |
| Contraseña | Passwd87 |

Cuadro 4.3: Fallo en el inicio de sesión cuando el usuario no introduce el email.

| Campo | Valor |
|---|---|
| Email | johnSmith@gmail.com |
| Contraseña | |

Cuadro 4.4: Fallo en el inicio de sesión cuando el usuario no introduce la contraseña.

Entonces el sistema debe mostrar un mensaje de error indicando que el usuario o la contraseña se han dejado en blanco.

Cuando la contraseña introducida no coincide con la del usuario.

| Campo | Valor |
|---|---|
| Email | johnSmith@gmail.com |
| Contraseña | BadPasswd01 |

Cuadro 4.5: Fallo en el inicio de sesión cuando el usuario introduce una contraseña incorrecta.

Entonces el sistema debe mostrar un mensaje de error indicando que esa contraseña no es correcta.

### 4.1.2. Paciente

#### Registro dentro de la aplicación

Descripción de la historia de usuario de registro para paciente:
Como usuario paciente.
Para poder acceder a los ejercicios de la aplicación.
Quiero poder registrarme dentro del sistema.
**Escenario: Registro de paciente con éxito**
Cuando introduzco mis datos en la pantalla de registro:

| Campo | Valor |
|---|---|
| Email | johnSmith@gmail.com |
| Contraseña | Long8 |
| Repetición Contraseña | Long8 |
| Edad | 65 |
| País | España |
| Localidad | Valladolid |

Cuadro 4.6: Registro paciente exitoso.

Entonces debería ver la pagina principal con los ejercicios disponibles.
**Escenario: Registro de paciente sin éxito**

Cuando dejo campos en blanco en la pantalla de registro tabla 4.7.

Entonces debería ver un mensaje de error en la pantalla de registro que me indique que hay un campo vacío.

Cuando hay un campo numérico dentro de un campo de cadena de texto en la pantalla de registro tabla 4.8.

| Email | Contraseña | Rep. de Contraseña | Edad | País | Localidad |
|---|---|---|---|---|---|
| | Passwd12 | Passwd12 | 65 | España | Valladolid |
| johnSmith@gmail.com | | Passwd12 | 65 | España | Valladolid |
| johnSmith@gmail.com | Passwd12 | | 65 | España | Valladolid |
| johnSmith@gmail.com | Passwd12 | Passwd12 | 65 | | Valladolid |
| johnSmith@gmail.com | Passwd12 | Passwd12 | 65 | España | |

Cuadro 4.7: Registro paciente sin éxito por tener campos vacíos.

| Email | Contraseña | Rep. de Contraseña | Edad | País | Localidad |
|---|---|---|---|---|---|
| johnSmith@gmail.com | Passwd12 | Passwd12 | 65 | 1 | Valladolid |
| johnSmith@gmail.com | Passwd12 | Passwd12 | 65 | España | 2 |

Cuadro 4.8: Registro paciente sin éxito por tener números en campos de texto.

Entonces debería ver un mensaje de error en la pantalla indicando que el formato para el campo País o Localidad no es válido.

Cuando introduzco una contraseña con un formato no válido tabla 4.9

| Email | Contraseña | Rep. de Contraseña | Edad | País | Localidad |
|---|---|---|---|---|---|
| johnSmithgmail.com | PASSWD87 | PASSWD87 | 65 | España | Valladolid |
| johnSmith@gmailcom | passwd87 | passwd87 | 65 | España | Valladolid |
| johnSmith@gmailcom | Passwd | Passwd | 65 | España | Valladolid |
| johnSmith@gmailcom | 12345678 | 12345678 | 65 | España | Valladolid |

Cuadro 4.9: Registro paciente sin éxito por email no válido.

Entonces el sistema mostrará un mensaje indicando que la contraseña no tiene el formato correcto. La contraseña debe tener una longitud de 8 caracteres y contener mayúsculas, minúsculas y números.

Cuando la repetición de la contraseña y la contraseña no coinciden4.10

| Email | Contraseña | Rep. de Contraseña | Edad | País | Localidad |
|---|---|---|---|---|---|
| johnSmithgmail.com | Passwd12 | Passwd21 | 65 | España | Valladolid |

Cuadro 4.10: Registro paciente sin éxito por introducir contraseñas que no coinciden.

Entonces debería ver un mensaje de error indicando que las contraseñas no son correctas.

Cuando introduzco un email que no cumple con la expresión regular 4.11

| Email | Contraseña | Rep. de Contraseña | Edad | País | Localidad |
|---|---|---|---|---|---|
| johnSmithgmail.com | Passwd12 | Passwd12 | 65 | España | Valladolid |
| johnSmith@gmailcom | Passwd12 | Passwd12 | 65 | España | Valladolid |
| 1 | Passwd12 | Passwd12 | 65 | España | Valladolid |
| noEsunCorreo | Passwd12 | Passwd12 | 65 | España | Valladolid |

Cuadro 4.11: Registro paciente sin éxito por email no válido.

Entonces el sistema muestra un mensaje de error indicando que el correo no cumple con el formato requerido por el sistema.

#### Selección de un ejercicio dentro de la plataforma

Descripción de la historia de usuario de selección de ejercicio para paciente:

Como usuario paciente.
Para poder realizar los ejercicios.
Tras iniciar sesión en el sistema de manera exitosa.
Quiero poder seleccionar uno de entre los varios que se muestran en el menú principal.

**Escenario: Selección de ejercicio**
Pulso sobre el botón correspondiente a uno de los ejercicios mostrados dentro del menú principal.

A continuación, se muestra una pantalla con la información referente al juego y un botón para acceder al mismo.

Pulso sobre el botón de acceso y el sistema me redirige el ejercicio seleccionado.

### 4.1.3. Desarrollador

#### Registro dentro de aplicación

Descripción de la historia de usuario de registro para desarrollador:
Como usuario desarrollador.
Para poder obtener la key necesaria para publicar servicios dentro de la plataforma.
Quiero poder registrarme dentro del sistema.

**Escenario: Registro de desarrollador con éxito**
Cuando introduzco mis datos en el formulario de registro, como se determina en el ejemplo 4.12

Entonces debería acceder a una pantalla donde se muestre la key necesaria para publicar servicios.

| Campo | Valor |
|---|---|
| Email | steveJobs@gmail.com |
| Contraseña | Passwd89 |
| Repetición Contraseña | Passwd89 |

Cuadro 4.12: Registro desarrollador exitoso.

**Escenario: Registro de desarrollador sin éxito**

Cuando dejo campos en blanco en el formulario de registro. Dentro del ejemplo tabla 4.13 se indican todos los campos que pueden quedar vacíos y provocar error.

| Email | Contraseña | Rep. de Contraseña |
|---|---|---|
| | Passwd89 | Passwd89 |
| steveJobs@gmail.com | | Passwd89 |
| steveJobs@gmail.com | Passwd89 | |

Cuadro 4.13: Registro desarrollador sin éxito por tener campos vacíos.

Entonces debería ver un mensaje de error en la pantalla de registro que me indique que hay un campo vacío.

Cuando las contraseñas introducidas por el desarrollador no coinciden 4.14:

| Campo | Valor |
|---|---|
| Email | steveJobs@gmail.com |
| Contraseña | Passwd89 |
| Repetición Contraseña | Passwd98 |

Cuadro 4.14: Registro desarrollador sin éxito por no coincidir las contraseñas.

Entonces debería ver un mensaje de error en la vista de registro que indique que las contraseñas no son correctas.

### Publicación de un ejercicio dentro de la plataforma

Descripción de la historia de usuario de publicación de ejercicio dentro de la plataforma:

Como desarrollador.
Para que los pacientes puedan acceder a los ejercicios.
Quiero poder registrar ejercicios dentro de la plataforma.

**Escenario: Registro de ejercicio con éxito**

Cuando realizo una petición hacia el servicio con los datos correspondientes al ejercicio 4.15

| Campo | Valor |
|---|---|
| Nombre | Ejercicio de sumas |
| Link | https://ejericiciodesumas.com/home |

Cuadro 4.15: Registro de ejercicio exitoso.

La plataforma retorna un mensaje de éxito sobre la operación de creación de ejercicio.

**Escenario: Registro de ejercicio sin éxito**

Cuando realizo una petición al servicio con un campo vació Tabla 4.16.

| Nombre | Link |
|---|---|
| | https://ejericiciodesumas.com/home |
| Sumas | |

Cuadro 4.16: Registro de ejercicio no exitoso por tener campos vacíos.

El servicio retorna un mensaje de error con el código HTTP 400 (*Bad Request*).

Cuando realizo una petición al servicio con un valor dentro del campo enlace que no es una url válida Tabla 4.17.

| Nombre | Link |
|---|---|
| Sumas | https:/ejericiciodesumas.com/home |
| Sumas | https//ejericiciodesumas.com/home |
| Sumas | https://ejericiciodesumascom/home |
| Sumas | https//ejericiciodesumas.net/home |

Cuadro 4.17: Registro de ejercicio exitoso.

El servicio retorna un mensaje de error con el código HTTP 400 (*Bad Request*).

### Borrado de un ejercicio dentro de la plataforma

Descripción de la historia de usuario de borrado de un ejercicio dentro de la plataforma:

Como desarrollador.

Para poder eliminar el acceso por parte de los pacientes hacía ejercicios publicados por mí.

Tras iniciar sesión dentro de la plataforma.
Quiero poder borrar ejercicios publicados por mí dentro de la plataforma.

**Escenario: Borrado de ejercicio con éxito**
Cuando selecciono uno de los ejercicios en la vista principal.
Accedo a una vista en detalle del ejercicio con un botón para borrar lo.
Al pulsar el botón se muestra una vista de confirmación sobre la acción de borrado.
Una vez realizada la confirmación el juego es borrado del sistema.

### 4.1.4. Investigador

#### Registro de investigador dentro de la aplicación

Descripción de la historia de usuario de registro para investigador:
Como usuario investigador.
Para poder acceder a los servicios proporcionados por la plataforma.
Quiero poder registrarme dentro del sistema.

**Escenario: Registro de investigador con éxito**
Cuando introduzco mis datos en el formulario de registro, como se determina en el ejemplo 4.18

| Campo | Valor |
|---|---|
| Email | albertEinstein@gmail.com |
| Contraseña | Long8 |
| Repetición Contraseña | Long8 |
| Organización | Sacyl |

Cuadro 4.18: Registro investigador exitoso.

Entonces debería poder acceder a la herramienta Grafana para la visualización de datos.

**Escenario: Registro de paciente sin éxito**
Cuando dejo campos en blanco en el formulario de registro. Dentro del ejemplo 4.19 se indican todos los campos que pueden quedar vacíos y provocar error.

Entonces debería ver un mensaje de error en la pantalla de registro que me indique que hay un campo vacío.

Cuando la organización introducida por el usuario no está registrada en el sistema:

| Email | Contraseña | Rep. de Contraseña | Organización |
|---|---|---|---|
| | Long8 | Long8 | Sacyl |
| albertEinstein@gmail.com | | Long8 | Sacyl |
| albertEinstein@gmail.com | Long8 | | Sacyl |
| albertEinstein@gmail.com | Long | Long8 | |

Cuadro 4.19: Registro investigador sin éxito por tener campos vacíos.

| Campo | Valor |
|---|---|
| Email | albertEinstein@gmail.com |
| Contraseña | Long8 |
| Repetición Contraseña | Long8 |
| Organización | Sagrado Corazón |

Cuadro 4.20: Registro investigador sin éxito por utilizar una organización no registrada en el sistema.

Entonces debería ver un mensaje de error en la vista de registro que indique que esa organización no esta registrada en la plataforma.

Cuando las contraseñas introducidas por el investigador no coinciden:

| Campo | Valor |
|---|---|
| Email | albertEinstein@gmail.com |
| Contraseña | Long8 |
| Repetición Contraseña | long9 |
| Organización | Sagrado Corazón |

Cuadro 4.21: Registro investigador sin éxito por no coincidir las contraseñas.

Entonces debería ver un mensaje de error en la vista de registro que indique que las contraseñas no son correctas.

### Visualización de *dashboards*

Descripción de la historia de usuario de visualización de *dashboards* para investigador:
Como usuario investigador.
Para poder utilizar los datos recogidos por la plataforma en la investigación del Alzheimer.
Quiero poder visualizar las métricas obtenidas de los ejercicios realizados por los pacientes de Alzheimer.

**Escenario: Visualización de datos**

Tras introducir las credenciales para entrar en Grafana, puedo acceder a los *dashboards* disponibles y ver la diversa información presentada en ellos extraída de los ejercicios realizados por los pacientes.

## 4.2. Modelo conceptual

En una fase inicial del proyecto se plasmó con un diagrama de clases los principales conceptos manejados por el sistema. Este modelado conceptual o del dominio se muestra en la Figura 4.1

## 4.3. Métricas

Para poder realizar el análisis por parte de investigadores y médicos es necesario establecer los valores que hay que recolectar de cada ejercicio realizado por los pacientes.

Para esta primera fase de la plataforma se plantean una serie de valores que deben almacenarse en persistencia de manera obligatoria por parte de todos los ejercicios.

Estos valores serán:

- Edad del paciente
- País
- Región
- Fecha de inicio del ejercicio
- Numero de fallos
- Tiempo total del ejercicio

La selección de estas métricas se debe a que permiten a un investigador ver resultados relevantes como el número de fallos, así como un progresivo aumento de los mismos en un intervalo de tiempo. Por ejemplo, permitiría mostrar el comportamiento de la media de fallos durante un año. Incluir también valores como la edad del paciente o la región/país desde el que se registro, permiten enriquecer las búsquedas realizadas sobre dichas métricas, dando la posibilidad de mostrar el comportamiento de la media de fallos para mayores de 65 años, durante un año, agrupando los valores por regiones dentro de España.

El establecer un conjunto de métricas comunes permite también resolver un problema de escalabilidad, ya que si cada ejercicio introduce un conjunto distinto de métricas sería necesario realizar un *dashboard* para cada uno de ellos. Este aumento de *dashboards* y gráficas dificultaría la visualización de datos y no haría escalable la plataforma.

Una solución intermedia, la cual permitiera a los desarrolladores de ejercicios nutrir con más valores las métricas almacenadas en el sistema, sería organizar los ejercicios por categorías. De

este modo, tras un estudio para establecer como agrupar los ejercicios, entorno a que valores se toma la decisión para encajar en una categoría o en otra. Para realizar este estudio sería necesario el análisis de los ejercicios realizados por los pacientes en de Alzheimer en la actualidad. Dado que se realiza una subdivisión sobre los ejercicios, se podría establecer también un subconjunto de métricas correspondientes a cada categoría para enriquecer los valores ya almacenados en el sistema.

Sería ideal contar con el apoyo de investigadores dentro de la revisión de las métricas almacenadas en el sistema, para detectar carencias dentro de ellas y añadir nuevos componentes que permitan avanzar en los estudios realizados.

Dado que las métricas y los datos relacionados son la piedra angular de la plataforma, un cambio sobre la manera en la que se tratan o sobre los valores a almacenar puede impactar sobre todo el sistema. Este apartado se trata en profundidad dentro del capítulo de Gestión de datos.

### 4.3.1. Almacenamiento de las métricas

Dentro de este apartado se abordará todo el proceso relacionado con las métricas dentro del sistema.

Las métricas se almacenarán en una tabla dentro de Postgres, por lo que cuando un desarrollador registra un nuevo ejercicio, el sistema genera una tabla para salvar las métricas derivadas de la realización de los ejercicios.

Cuando un usuario finaliza un ejercicio, se realiza una petición POST hacia el microservicio de Gestión de ejercicios, que dentro del cuerpo de la petición se incluye el identificador del ejercicio, el identificador del usuario, fecha de inicio, número de fallos y tiempo total del ejercicio. Posteriormente con esos datos, antes de insertar en la tabla, el sistema realiza una búsqueda sobre la base de datos de usuario para obtener región, país y edad.

Por último el sistema realiza una inserción dentro de la tabla correspondiente al ejercicio con los datos de métricas establecidos al inicio y descartando el identificador de usuario para garantizar la privacidad del mismo.

La consulta de estos datos dentro de la plataforma se realizará desde el componente Grafana.

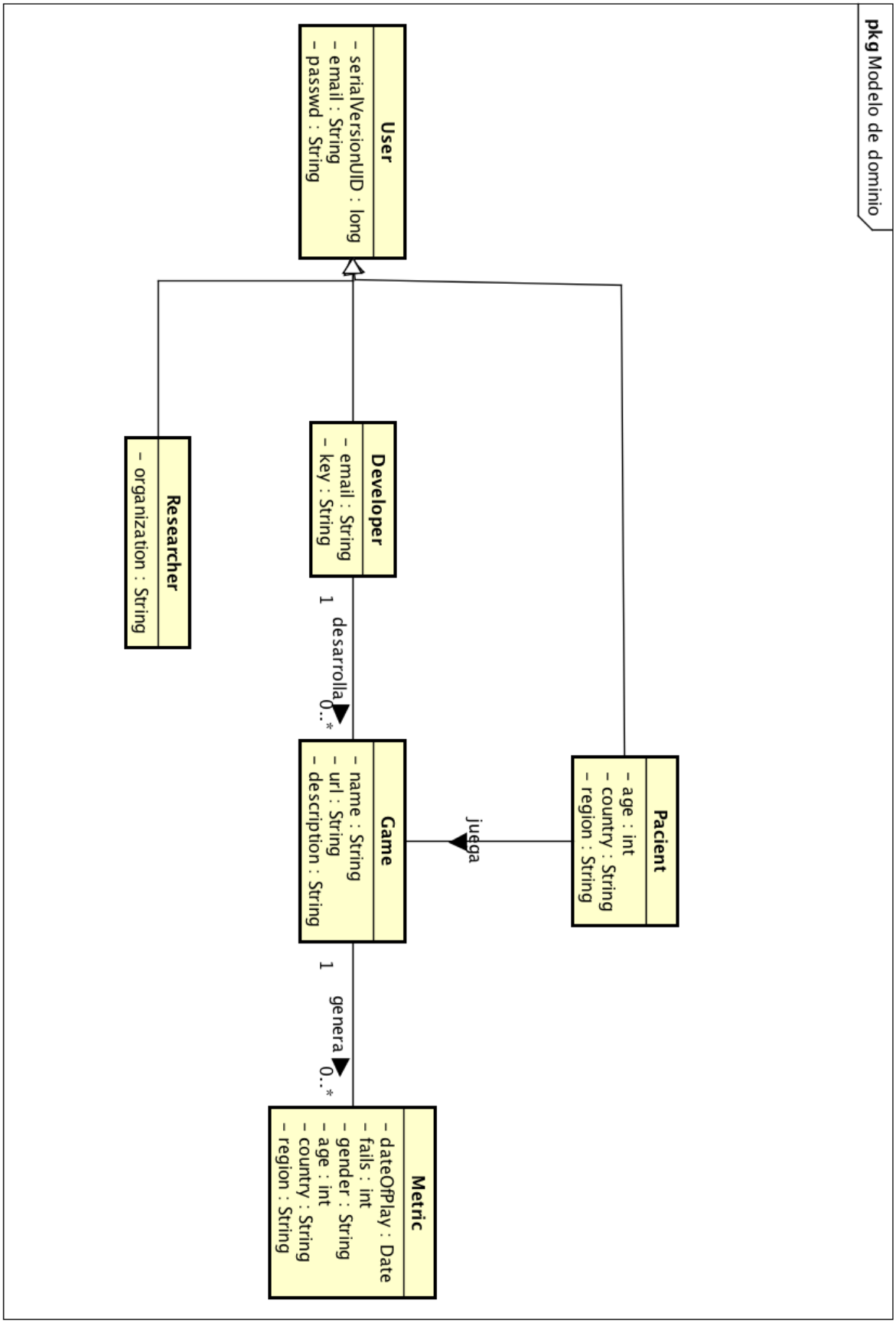


Figura 4.1: Diagrama de modelo de dominio.

*Capítulo 5*

# Gestión de datos

Uno de los propósitos de la plataforma es ayudar al tratamiento de los pacientes de Alzheimer mediante la observación y estudio de los resultados obtenidos por los pacientes a medida que realizan ejercicios a lo largo del tiempo. Para este propósito la plataforma debe poder mostrar esos datos y, sobre todo, almacenarlos.

Lo anterior plantea un problema claro: la privacidad de los usuarios. Por ejemplo, en caso de que un nuevo usuario se registre dentro de la plataforma en una provincia con un número reducido de usuarios activos podría llegar a determinarse su identidad.

Este hecho no solo se da en una pequeña plataforma, sino que grandes corporaciones como Apple, Google o Telefónica han tenido que afrontar y resolver este problema antes. En el caso de Apple, los datos se anonimizan en varios niveles para garantizar que no se pueda identificar al usuario a partir de estos datos. Gran parte del trabajo realizado por esta empresa se ha centrado en la encriptación y protección de datos que son enviados por sus aplicaciones en el *end-to-end*.

Un ejemplo del giro de la industria hacia esta fuerza de los datos y su debido tratamiento es la creación de un gran *data lake* por parte de Telefónica en el cual contener todos los datos de sus usuarios y poder aplicar algoritmos para extraer mejor información de los mismos para la mejora del negocio. Estos datos son anomizados y se requiere del consentimiento del usuario para acceder a ellos y garantizar que no son usados para propósitos indebidos.

Actualmente con el aumento de la capacidad de los sistemas y el crecimiento de los datos generados con el uso de los sistemas ha permitido crear un nicho dentro de la industria para el desarrollo de soluciones Big Data. Este conocimiento extraído de los datos permite a las empresas ofrecer mejores servicios y productos a medida. Sin embargo, ese aumento del conocimiento sobre los usuarios plantea problemas relacionados con la privacidad que deben ser tratados con la debida atención. Dentro de la Unión Europea, la sociedad ha tratado de legislar para proteger a los usuarios.

## 5.1. GDPR

En mayo de 2018 el Reglamento General de Protección de Datos (GDPR) de la Unión Europea entró en vigor con el propósito de mejorar la protección sobre los datos personales de los usuarios.

No es el objetivo de este capítulo detallar en que consiste la ley, sino entender el impacto que esta nueva ley tendrá sobre la industria y a pesar de que se trata de un prototipo, las implicaciones que podría tener sobre la plataforma que se está intentando desarrollar en este proyecto.

La ley de ámbito europeo busca proteger todo el ciclo de vida que puedan tener los datos dentro de la organización (almacenamiento, procesamiento, acceso transferencia y divulgación) y permite sancionar duramente a las organizaciones que no velen por la integridad de dichos procesos.

Las motivaciones para la creación de esta ley son dos principalmente:

- Dar control a los usuarios sobre sus datos.
- Ofrecer un entorno más claro y simple para la operación de las empresas.

### 5.1.1. ¿Qué implicación tiene la GDPR sobre el proyecto?

Tras registrarse, a los usuarios se les informará los datos generados de la realización de sus ejercicios serán vertidos a un sistema para poder ser analizados por investigadores con el propósito de aumentar el conocimiento sobre la patología y tratar de ayudar en su tratamiento.

Se requerirá su consentimiento para que estos datos entren a formar parte del sistema. También se les informará de que en futuras *releases* se proporcionará una entrada para revocar este consentimiento y de ese modo dejar de verter datos dentro de la plataforma. También se indicará en dicho mensaje que los datos que entren en un la plataforma no podrán borrarse cuando se revoque el consentimiento del usuario, ya que pasarán por un proceso de anonimización que no permitirá vincular al usuario con los datos generados.

Dado el perfil de los usuarios de la plataforma, personas generalmente de avanzada edad, es todo un reto la redacción de un mensaje de este tipo, ya que incorpora diversos conceptos que para una persona no familiarizada con el uso de datos y permisos puede ser especialmente confuso.

Para garantizar que el mensaje fuese lo más comprensible posible se realizaron varios borradores y se trató de validar con personas de avanzada edad para iterar y poder dar el mensaje lo más claro posible.

Otra de las implicaciones del proceso es el diseño del trato que recibirán los datos desde que entran la plataforma y durante todo su ciclo de vida.

Cuando un paciente realiza un ejercicio, sus resultados y tiempos serán almacenados en una tabla de resultados, en la que solo se almacenarán cuando, la localización (país y provincia) y resultados derivados de esas prácticas.

Las métricas almacenadas en la plataforma se componen únicamente de número de fallos durante el ejercicio, fecha en la que se realizó el ejercicio, edad del paciente, sexo, país y región. Estos valores, surgidos durante la entrevistas con médicos residentes en centros para la tercera edad, resultan de utilidad para hacer muestreos.

Pero en ningún momento, estos resultados podrán ser relacionados con los usuarios. Al concluir un ejercicio, se realiza la ingesta de estos datos dentro de la plataforma pero debido a que no contienen ningún identificador, no pueden ser relacionados directamente con un paciente. Debido a que los datos de resultados son independientes de los datos que el usuario da al sistema durante el registro, en caso de que un usuario quisiera revocar el consentimiento de almacenamiento de datos dentro de la plataforma, solo serían borrados sus datos de registro. No podrían borrarse los datos derivados de los sucesivos ejercicios porque el sistema no guarda una correlación entre estos resultados y el usuario como tal.

Los datos no son procesados ni enriquecidos durante su estancia dentro de la plataforma ni en la realización de los ejercicios por parte de los pacientes.

El acceso a los datos registrados en cada tabla de resultados tiene un permiso de escritura limitado para el proceso que gestiona los ejercicios, por otro lado solo Grafana tendrá permisos de lectura sobre dichas tablas y los accesos a Grafana están totalmente restringidos ya que se crean bajo demanda en el sistema.

La seguridad del entorno es un aspecto importante que también debe ser tenido en cuenta para garantizar la integridad de los mismos.

Para esta primera fase inicial los datos se almacenan en la instancia de Postgres que forma parte de la plataforma. Para evoluciones posibles de la plataforma este aspecto deber ser considerado y evitar las vulnerabilidades conocidas para Postgres así como restringir el acceso a la base de datos por contenedores Docker corriendo dentro del sistema.

*Capítulo 6*

# Arquitectura y patrones

## 6.1. SaaS

El software como servicio o SaaS es un modelo de distribución de software donde el soporte lógico y los datos que se manejan están alojados en servidores externos al ordenador desde el que se conecta el usuario.

Al estar el software alojado en los servidores de la empresa proveedora, la implementación de nuevas funcionalidades y el soporte es responsabilidad de la empresa proveedora.

Este modelo encaja con el objetivo del proyecto, ya que, aunque en un primer momento podría estar más cercano a una plataforma como servicio (PaaS) en este caso las actividades que los desarrolladores publicarán en la plataforma no están alojadas en la misma, por lo que el mantenimiento de este software recae en el desarrollador.

Si bien la plataforma ofrece ciertas funcionalidades de administración y la posibilidad de consultar datos para investigadores.

## 6.2. Arquitectura de la plataforma

La arquitectura de la plataforma debe satisfacer los siguientes requisitos:

- **Escalable**, este es un requisito deseable ya que ante un aumento de los usuarios la plataforma debe poder dimensionarse en consecuencia. Para poder escalar solo las funcionalidades que están bajo una mayor presión es necesario un bajo acoplamiento entre los elementos de la plataforma. Como solución a ello la industria ha desarrollado la arquitectura basada en microservicios.

- **Flexible**, en un entorno cambiante y en el que se busca reducir al máximo posible el *time to market* es un requisito que las nuevas funcionalidades sean accesibles lo antes posible, para ello los componentes deben estar versionados y contar con una estrategia de integración continua.

- **Robusta**, ante un posible fallo de la plataforma o uno de sus servicios, es deseable que estos servicios puedan mantener una funcionalidad parcial. Esto se consigue teniendo los servicios que componen la plataforma en alta disponibilidad, esto es, varias réplicas dentro del mismo sistema de los servicios para así en caso de fallo poder redirigir las peticiones entrantes hacia los servicios que todavía están activos. Así como un sistema de alarmado que permita monitorizar la plataforma desde el exterior y detectar la proximidad a un umbral crítico.

- **Trazable**, esta característica no tiene un impacto directo en el servicio percibido por los usuarios, pero si es necesaria para el buen funcionamiento de la plataforma y para la pronta detección de errores. Los servicios que componen la plataforma están dotados de interfaces REST para comunicarse entre ellos, así como *logs* de las operaciones internas realizadas por cada servicio. Por tanto es necesario que una petición pueda trazarse fácilmente por estos logs. Para ello, en la industria está extendido el uso de una cabecera http denominada *correlator*. Esta cabecera puede ser un entero o una cadena, aunque se suele utilizar un UUID, por la mínima probabilidad de colisión. De este modo, realizando una búsqueda en base a dicho valor podemos detectar en qué punto se produjo el error.

### 6.2.1. Arquitectura basada en microservicios

Como respuesta a los grandes monolitos y a la corriente de desarrollo ágil, la industria esta apostando por las arquitecturas basadas en microservicios.

En una visión contrapuesta a los monolitos que se venían realizando, una arquitectura basada en microservicios tiene la particularidad de que permite una mayor escalabilidad de los servicios de manera individual, permitiendo incrementar la capacidad de los sistemas que están recibiendo más carga para garantizar un *throughput* constante y adecuado a las necesidades del servicio. También permite replicar servicios dentro del sistema para obtener alta disponibilidad y hacer el sistema más resistente ante perdidas de servicio. Esta replicación se hace a nivel de nodo físico, destinando las replicas de los microservicios a distintos nodos físicos.

En caso de las arquitecturas de microservicios basadas en *cloud*, esta replicación se puede hacer a nivel incluso de zona de disponibilidad, permitiendo tener réplicas de un microservicio corriendo en distintos *cluster* repartidos por una zona geográfica concreta, garantizando la disponibilidad. Para este tipo de arquitecturas, se configuran balanceadores de carga propios de la *cloud* para redirigir tráfico entre estos microservicios.

La implementación de una arquitectura basada en microservicios también tiene impactos visibles dentro del trabajo diario de los desarrolladores que implementan estos servicios. Los tiempos de compilación se reducen drásticamente, ya que ya no se compila un gran monolito, sino pequeñas partes de él. Permite hacer una planificación de entrega de *releases* que entran a formar parte del sistema con un impacto mínimos sobre el resto de servicios, e incluso permite organizar un equipo en pequeñas unidades dedicadas a cada servicio optimizando así los tiempos

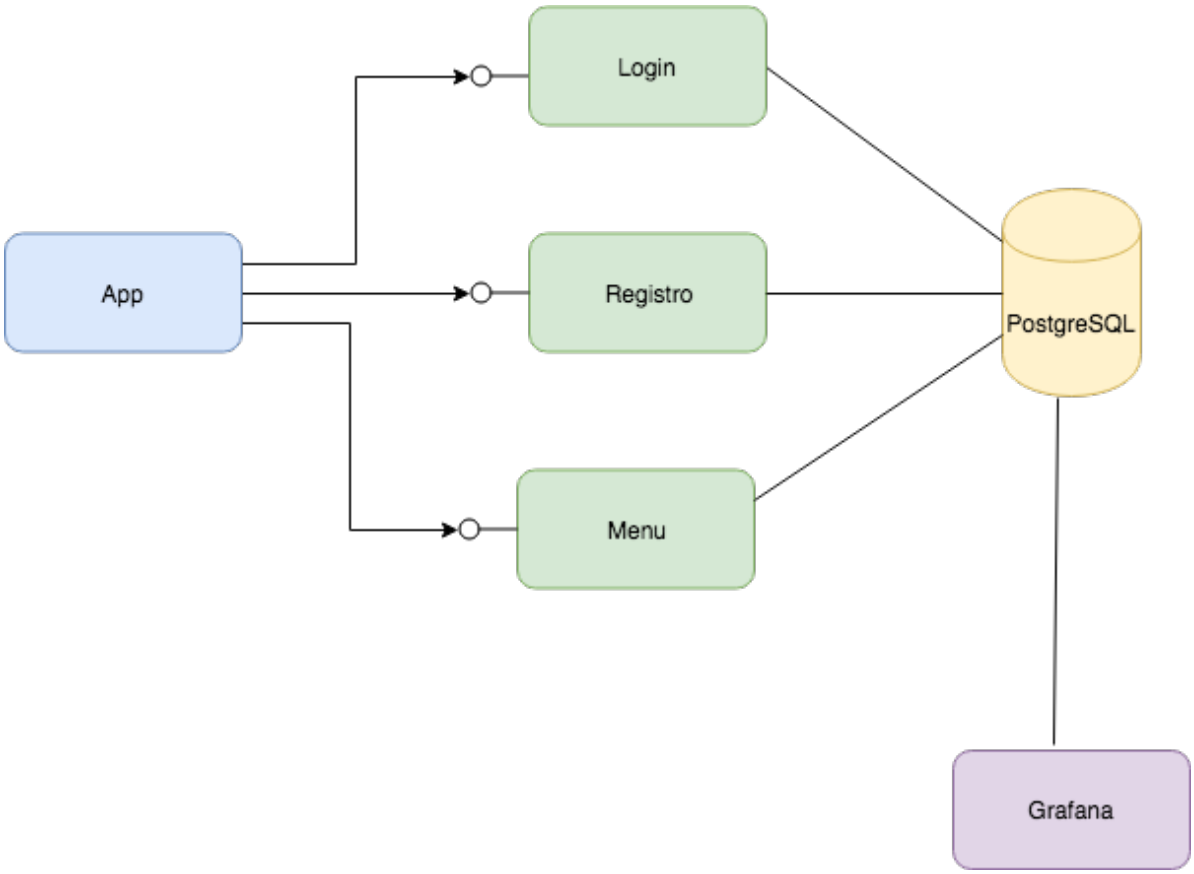


Figura 6.1: Diagrama a alto nivel de la arquitectura de la plataforma

de desarrollo, ya que el conocimiento dentro de ese grupo sobre el componente es muy profundo. Recordando que los miembros de estas unidades pueden y deben realizar pequeñas tareas sobre otros servicios, para evitar generar silos de conocimiento.

Del mismo modo, la industria está pivotando cada vez más hacia tener su infraestructura en la nube, y debido al sistema de tarificación que utilizan estas tecnologías, es más viable a nivel económico desplegar varios servicios en máquinas con recursos más reducidos y en las cuales podemos personalizar más el uso de recursos.

Como ejemplo de una arquitectura basada en micro servicios en la Figura 6.1 se muestra un sencillo diagrama conceptual de la arquitectura que compone la plataforma desarrollada en este proyecto.

Dentro de las arquitecturas basadas en microservicio son elementos importantes la replicación de los servicios y el balanceo de carga entre ellos. Otro elemento importante es la posibilidad de escalar estos servicios y gestionar su estado, detectar cuando un servicio web no esta recibiendo peticiones HTTP o cuando no se están replicando los datos dentro de una réplica de la base de datos para poder realizar acciones sobre el servicio y devolverle a su estado habitual.

En la Figura 6.2 se muestra un diagrama de despliegue de la arquitectura del proyecto sobre distintos nodos físicos. Como se ha comentado previamente, este hecho conjunto a replicación de servicios permiten conseguir alta disponibilidad dentro de la arquitectura.

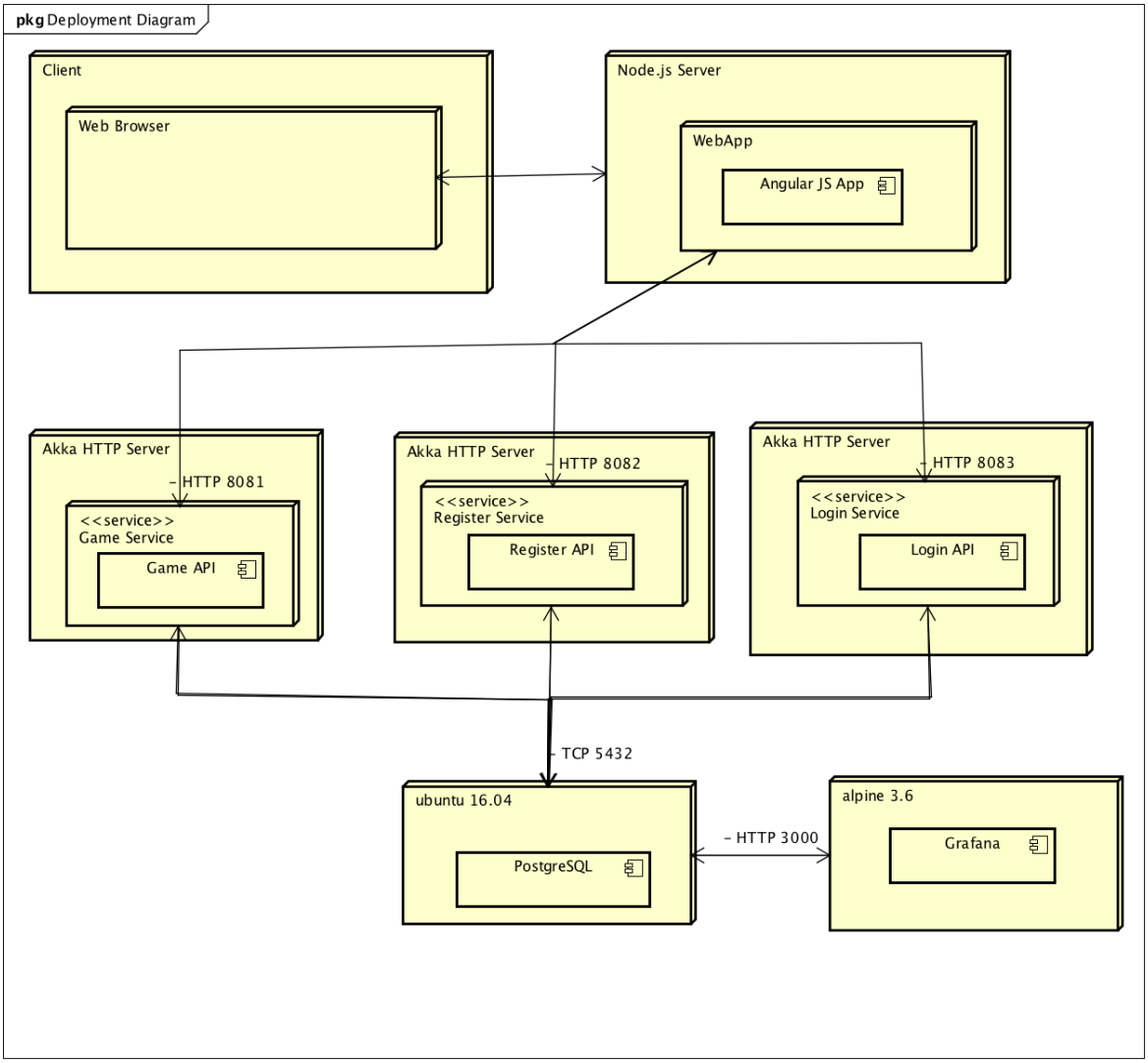


Figura 6.2: Diagrama de despliegue de la plataforma

La comunicación entre los distintos nodos, se realiza a través de los puertos y protocolos especificados por los servicios. Esto implica, que durante la fase de despliegue, es necesario enlazar dichos puertos para permitir esta comunicación.

Para facilitar el manejo y operación de los servicios y permitir que la plataforma pueda escalar, es común delegar en un orquestador. El orquestador es el encargado de balancear carga entre los servicios, de ofrecernos las réplicas que deseamos sobre un servicio, monitorizarlo para confirmar que esta sano. En caso de no estarlo, es el propio orquestador el encargado de aplicas las medidas necesarias para devolverlo al estado deseado.

Dentro del proyecto se ha experimentado con el orquestador Kubernetes. Para un proyecto de estas dimensiones y con las características de producto mínimo viable, puede considerarse sobreingeniería.

### 6.2.2. API REST

Para la interconexión entre los distintos microservicios que componen la plataforma, así como con los clientes que puedan acceder a ella, se utilizan como interfaz varias API REST.

Este interfaz fue definido por Roy Fielding, padre de la especificación HTTP en su publicación *Achitectural Styles and the Design of the Network-based Software Architectures* [6]. Estas interfaces REST(Representational State Transfer) son especialmente sencillas de desarrollar y de entender por lo que han tenido un uso mayoritario frente a otro protocolos como SOAP (Simple Object Access Protocol) [18] [19].

Los elementos que caracterizan un API REST son los siguientes:

- Protocolo cliente/servidor sin estado: Cada petición es autocontenida, es decir, no se requiere de más información por parte de cliente o servidor para su ejecución. Entre petición y petición, el servidor no persiste la sesión.
- Conjunto de operaciones definidas: Cuenta con un conjunto de operaciones delimitadas que se pueden realizar sobre los datos y se identifican mediante cuatro palabras clave. GET (leer y consultar), POST (crear), PUT(editar) y DELETE (borrar datos).
- Uso de URL (Uniform Resource Locator) para identificar de forma única a los recursos. Las URLs son un tipo de URI (Uniform Resource Identifier) y tiene que obedecer ciertas normas como ser únicas, no utilizar verbos, ser independientes del formato, etc.
- Al utilizar HTTP, proporciona códigos de estado y códigos de error para saber cuál es el resultado de las llamadas.
- Uso de cabeceras HTTP para enriquecer la información que responde nuestro sistema.
- Permite cachear información estática, lo que ocasiona un aumento en el rendimiento.

Al carecer de estado, permite una gran escalabilidad al no gastar recursos en persistir información sobre el usuario.

Dentro de las aplicaciones web el desarrollo de API REST está ampliamente extendido y grandes sistemas como Github, Twitter, Amazon o Facebook cuentan con una API REST para exponer sus servicios a otros desarrolladores. Esto da una importancia de este protocolo dentro de la industria.

### 6.2.3. Sistema Asíncrono

Los sistemas asíncronos son los escenarios más cercanos al mundo real, ya que en la mayoría de los casos no se puede garantizar el orden de llegada de los datos, ni el tiempo que tardará una petición en resolverse.

La programación asíncrona permite que algunas operaciones devuelvan el control al método que realizó la llamada antes de que se haya completado, continuando ésta en segundo plano.

Esta funcionalidad hace la ejecución más rápida y beneficia positivamente la escalabilidad, pero dificulta la legibilidad del programa si no se pone especial cuidado y siguen buenas prácticas. Otro punto negativo relacionado con el desarrollo de sistemas asíncronos es que resulta difícil realizar una depuración del código, porque en muchos casos no conocemos el valor retornado por las operaciones intermedias mientras continua la ejecución del método que origino la llamada.

Para conseguir devolver el control de la ejecución al método que realiza la llamada sin tener todavía el resultado, se han desarrollado distintos modelos tratando de asemejar el comportamiento al del código secuencial:

- Modelo de Promesas, se opera con los valores que retorne la operación independientemente de cuando se vayan a obtener estos valores de éxito o fracaso. Se "garantiza"que se obtendrán esos valores y se continúa con la ejecución. Es decir, la ejecución del programa progresa sin el valor específico y cuando se disponga de dicho valor, se aplican todas las operaciones que quedaban pendientes, pero no se bloquea el avance del programa.

- Modelo de eventos, requiere la implementación de una arquitectura basada en eventos que permitan informar con señales de éxito o fracaso del fin de la operación. Este modelo incluye la desventaja de que la correlación de valores entre los métodos tiene que ser síncrona.

- Modelo de generadores, dentro de este modelo se utilizan generadores para devolver temporalmente el control al programa llamante y retornar en un momento posterior a la rutina restaurando el estado en el punto que se abandonó su ejecución.

Dado que la plataforma se basa en interacciones entre la aplicación web y los distintos microservicios a través de un API REST, estas interacciones deben ser asíncronas. El objetivo de este requisito es evitar tiempos de espera acumulados al resolver las distintas peticiones.

El modelo asíncrono se aplica tanto en la aplicación web como en los distintos microservicios. Concretamente Play Framework es internamente completamente asíncrono. Pero requiere de ciertas operaciones por parte del programador para garantizar que las interacciones con el framework seguirán siendo asíncronas, esto es, garantizando que los controladores devolverán *promesas* y por otra parte configurar un *pool* de hilos específico para el acceso a base de datos, ya que PostgreSQL no implementa un acceso asíncrono a estos.

### 6.2.4. Arquitectura basada en capas

Con el fin de separar responsabilidades y siguiendo el patrón arquitectónico basado en capas, propuesto por M. Richars en *"Software architecture patter"* [15] dentro de cada servicio aplicamos una arquitectura basada en capas como se puede ver en la Figura 6.3

Aplicando este planteamiento podemos separar en capas en función de la funcionalidad que se realice dentro de cada una.

En el servidor, dentro del controlador encapsulamos la funcionalidad referente a la recepción de peticiones y retorno de respuesta (interacción con otros componentes). En el repositorio encapsulamos la funcionalidad referente al acceso a persistencia. Finalmente dentro de la capa Modelo agrupamos las clases necesarias para modelar el servicio.

## 6.3. Patrones

### 6.3.1. Patrón Repositorio

Patrón presentado en el libro de Martín Fowler *Patterns of Enterprise Application Architecture* [7] y se considera que los autores de dicho patrón son Edwar Hieatt y Rob Mee.

Para evitar tener que lidiar con el acceso a datos almacenados en persistencia dentro del código de la aplicación, se propone como solución crear una capa dentro de la aplicación a modo de abstracción y delegar en dicha capa el acceso a persistencia.

La creación de dicha capa nos permite eliminar del resto de nuestro código complejas sentencias SQL que de otro modo dificultarían la legibilidad de nuestro código, ya que no parece lógico recuperar objetos mediante sentencias SQL dentro de clases orientadas al objeto.

El uso de este patrón también aporta la ventaja de desacoplar nuestra aplicación del motor de base de datos que estemos utilizando por debajo.

La principal idea del patrón repositorio es que actúe como una colección en memoria del modelo de datos, que nos permita filtrar, borrar y recuperar elementos sin necesidad de escribir código SQL para ello.

Este patrón de diseño esta muy ligado al concepto de modelo de dominio, término muy común cuando se trata Domain Driven Design. Este modelo será el modelo principal de nuestra aplicación y recibirá el nombre de dominio. La principal diferencia con el modelo de base de datos es que permite cambiar la forma de pensar de un punto de vista de tablas a centrarse en la mejor forma de gestionar los objetos dentro del lenguaje de programación.

Este patrón se ha aplicado dentro de todos los servicios en las clases Reposotory, para hacer el acceso a la capa de persistencia, utilizando el ORM proporcionado por Play Framework, Ebean.

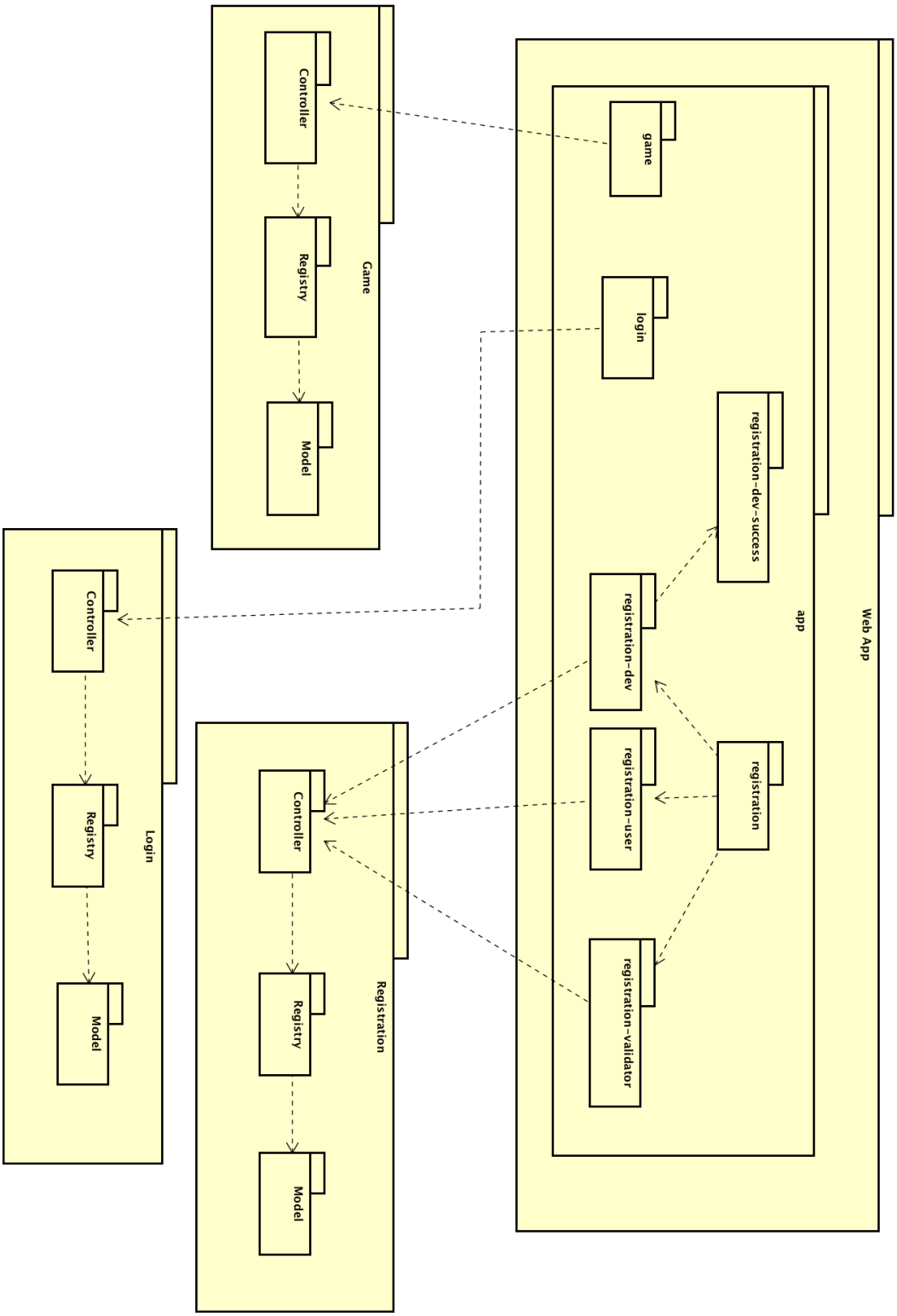


Figura 6.3: Diagrama a alto nivel de la arquitectura de la plataforma

*Capítulo 7*

# Diseño de interfaz gráfica

En este capítulo se analizará el proceso de diseño y los prototipos realizados para el desarrollo de la interfaz web.

A la hora de desarrollar, es necesario tener en cuenta quienes van a ser los usuarios finales de nuestra aplicación. Hay una abundante bibliografía sobre el diseño de aplicaciones y el estudio de la interacción entre el usuario y la aplicación. En este caso, hemos centrado el foco en los aspectos visuales y tratando de visualizar como sería el uso más cómodo de la aplicación para los usuarios.

## 7.1. Interfaz para inicio de sesión

Todos los usuarios de la plataforma, paciente, desarrollador o investigar deben iniciar sesión dentro de la plataforma para poder acceder a sus servicios por lo que esta interfaz debe ser uniforme para todos. El diseño de esta interfaz se muestra en la Figura 7.1

Dentro de esta vista, además del botón para iniciar sesión se añade uno para poder acceder a la vista de registro Figura 7.2, en caso de que no posean usuario.

## 7.2. Interfaz para registro dentro de la plataforma

Para el proceso de registro de usuario se han diseñado las siguientes vistas:

En primer lugar, todos los usuarios que quieran registrarse en la plataforma accederán a la interfaz representada en la Figura 7.2

En esta figura se muestran varios botones para elegir bajo que rol se desea registrar el usuario. En función de la opción seleccionada se accede a un formulario determinado.


Figura 7.1: Diseño de pantalla para inicio de sesión.

### 7.2.1. Registro Paciente

Si se selecciona la opción para registro de paciente dentro de la Figura7.2 se accede a la interfaz representada en la Figura7.3

Dentro de la interfaz representada en la Figura 7.3 se piden los datos necesarios para poder registrar al paciente dentro del sistema y que resultarán de utilidad posteriormente para elaborar las métricas.

Una vez que el paciente se registra con éxito se le devuelve a la vista de inicio de sesión Figura 7.1 para que se identifique. Una vez identificado correctamente ya puede acceder a la vista principal donde se presentan todos los ejercicios disponibles para el paciente. Esta vista está representada en la Figura 7.4

### 7.2.2. Registro Desarrollador

En caso de que la opción seleccionada en la Figura 7.2 se corresponda con la opción de desarrollador, tras pulsar el botón Desarrollador se accede a la vista para el registro de desarrollador Figura7.5

Una vez registrado con éxito, el sistema muestra una vista con un mensaje de bienvenida, acompañado de un código Figura7.7. Este código servirá para identificar unívocamente al desarrollador frente al sistema y le permitirá publicar nuevos ejercicios y gestionar los ejercicios de los que sea autor y ya estén en la plataforma.

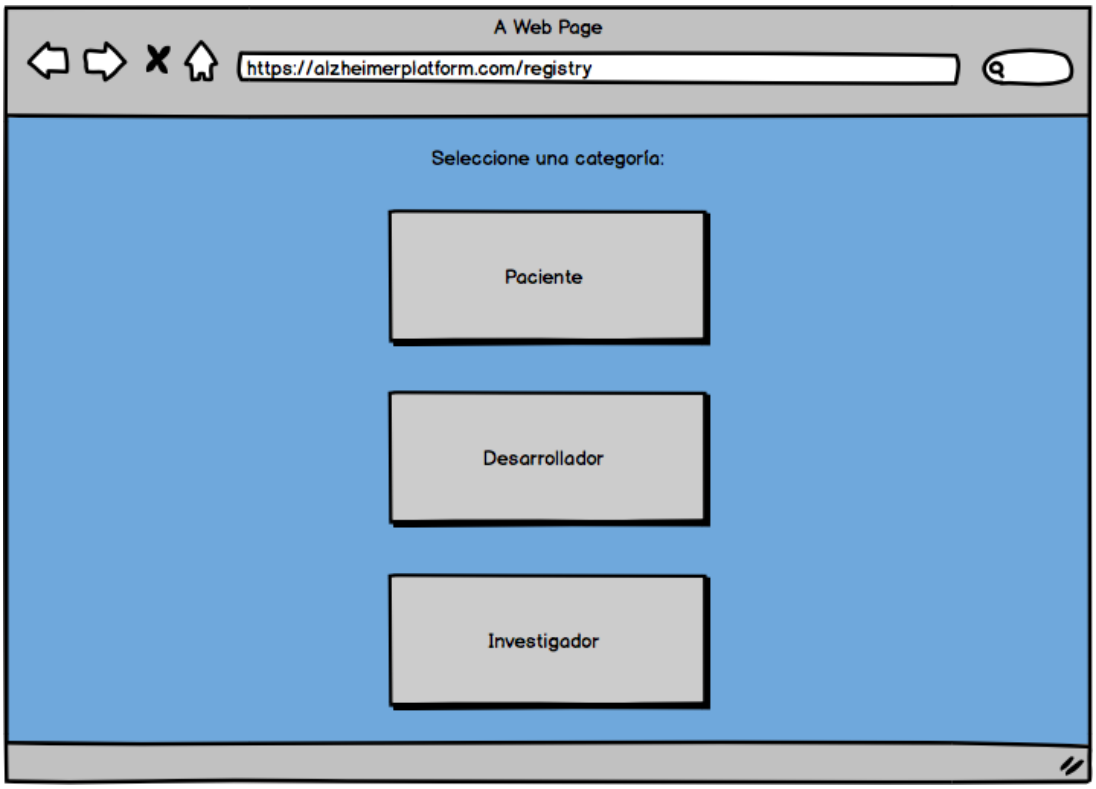


Figura 7.2: Diseño de pantalla para selección de registro.

### 7.2.3. Registro Investigador

Para el registro de investigadores, una vez pulsado el botón Investigador en la vista Figura7.2 el usuario es redirigido a la vista Figura 7.2 en la cual se muestra un formulario.

Una vez registrado con éxito el usuario investigador, ya podría acceder a los servicios de visualización de datos. Dichas interfaces no han sido añadidas en este capítulo porque se trata de vistas propias de la interfaz de Grafana.

## 7.3. Interfaz ejercicios

Para la gestión e interacción con los ejercicios dentro de la plataforma, se han diseñado las siguientes interfaces:

### 7.3.1. Selección de ejercicios

Tras iniciar sesión con éxito Figura 7.1 y visualizar la pantalla principal Figura 7.4, un paciente puede seleccionar un ejercicio para acceder a el. En ese caso, se le muestra al paciente la vista en detalle del ejercicio Figura 7.8. Al hacer pulsar sobre el botón con el texto .Acceder al ejercicio", se le redirigiría a la página donde el desarrollador tuviera alojado el ejercicio.

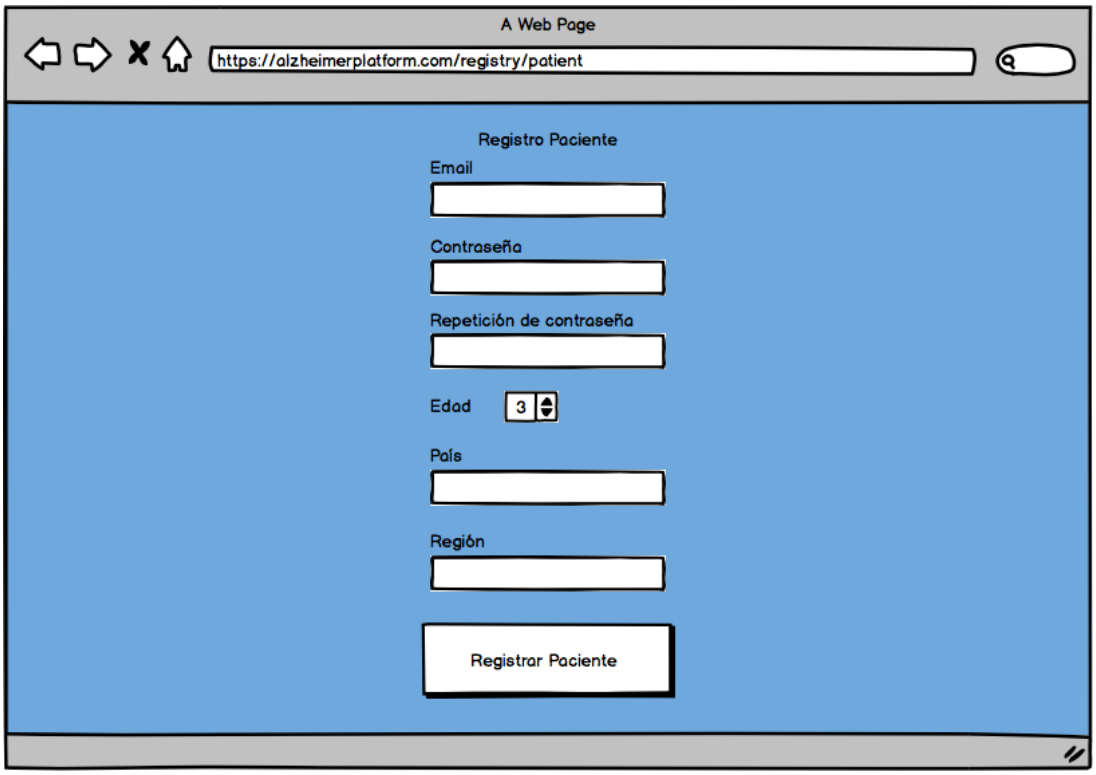


Figura 7.3: Diseño de pantalla para registro de paciente.

### 7.3.2. Publicación de ejercicios

En caso de un desarrollador, una vez iniciada sesión se muestra una vista del conjunto de los ejercicios publicados por el, desde la que puede visualizar y gestionar el contenido Figura7.9.

Al pulsar sobre el botón verde con el texto Publicar, que se encuentra en la esquina superior derecha, el desarrollador accede a una vista en la que se le presenta un formulario para introducir los datos necesarios para la publicación del nuevo ejercicio Figura 7.10.

En caso de que el desarrollador este conforme con los datos introducidos, al pulsar el botón izquierdo verde, ese ejercicio se publicaría dentro de la plataforma y, en esta versión, estaría disponible para todo el conjunto de pacientes. En versiones posteriores, esta publicación podría ser revisada para garantizar que cumple con los estándares de calidad establecidos por la plataforma.

### 7.3.3. Edición y borrado de ejercicios

Tras iniciar sesión como desarrollador en la vista Figura 7.1 y acceder a la vista con los ejercicios publicados en la vista Figura 7.9. Un desarrollador puede hacer pulsar sobre uno de los ejercicios publicados por el y accederá a la vista correspondiente a la Figura 7.11.

Una vez actualizado uno o varios campos, si pulsa sobre el botón verde en la parte inferior, la información referente al ejercicios se actualizará. Para deshacer esta acción solo tiene que volver hacia atrás en la página antes de pulsar sobre publicar.

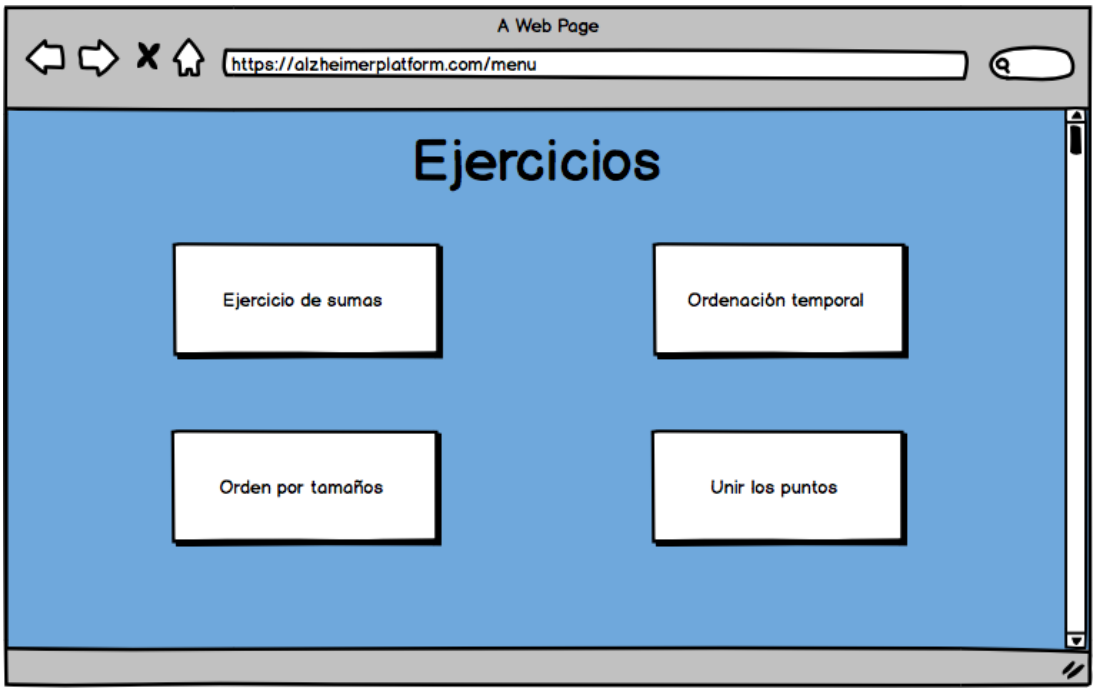


Figura 7.4: Diseño de interfaz para la vista principal de la plataforma.

La vista correspondiente a la Figura 7.11 también puede ser usada por el desarrollador para eliminar un ejercicio publicado por el. Para realizarlo, solo tiene que pulsar sobre el botón rojo de la parte inferior y accederá a una vista Figura 7.12 en la que se pide una confirmación para borrar de manera definitiva de la plataforma ese ejercicio.

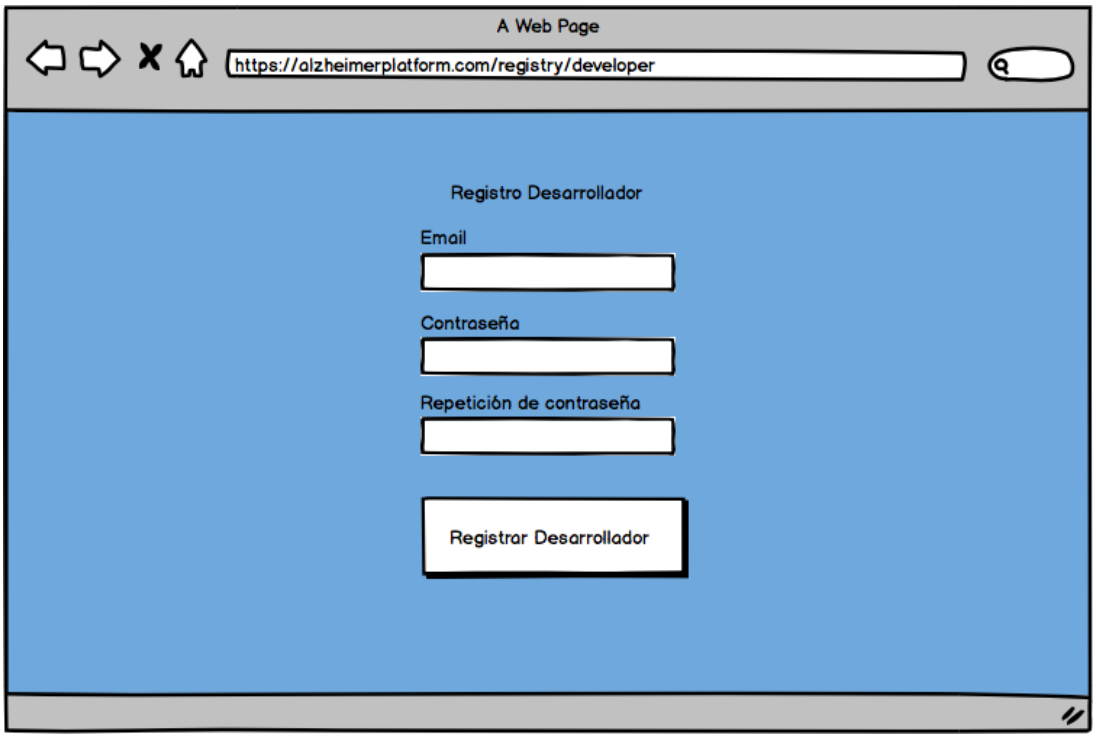


Figura 7.5: Diseño de interfaz para el registro de desarrolladores dentro de la plataforma.

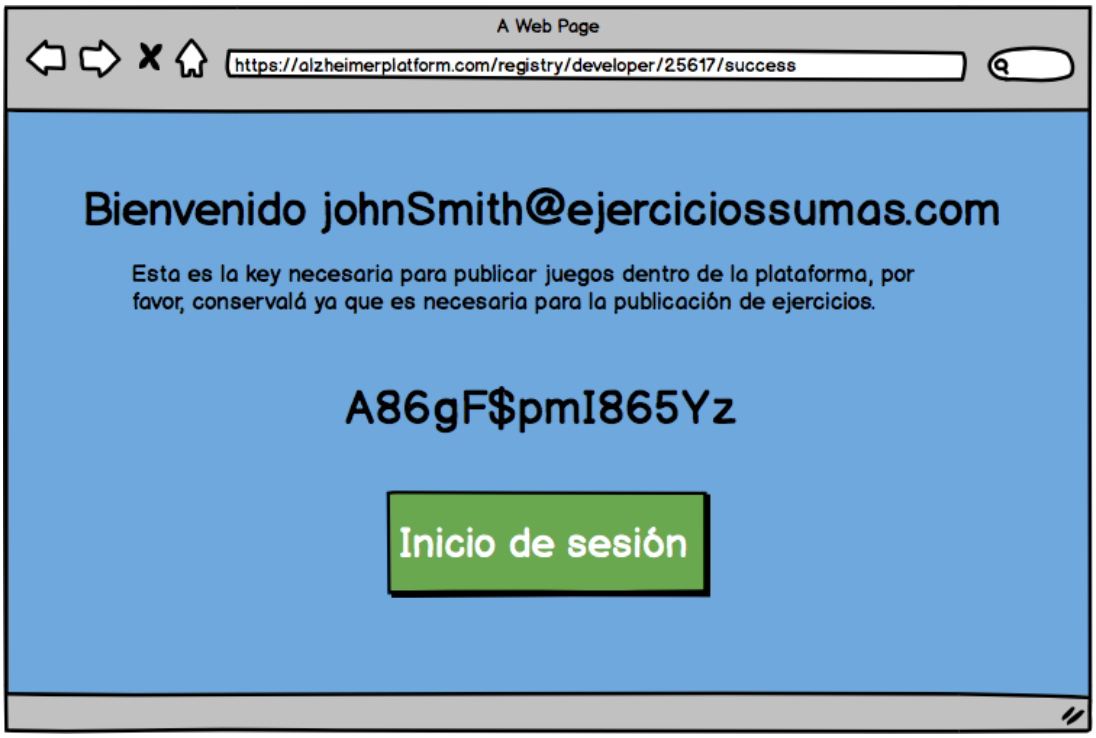


Figura 7.6: Vista con mensaje de bienvenida y clave única para el desarrollador.

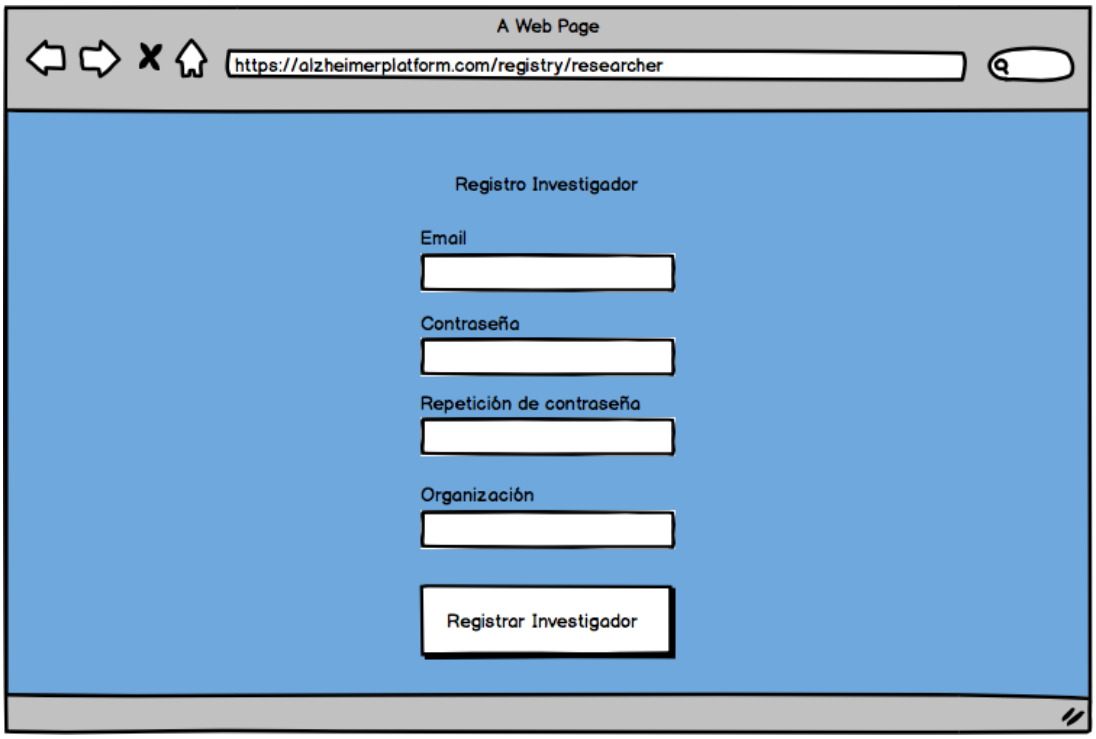


Figura 7.7: Diseño de interfaz para el registro de investigadores dentro de la plataforma.

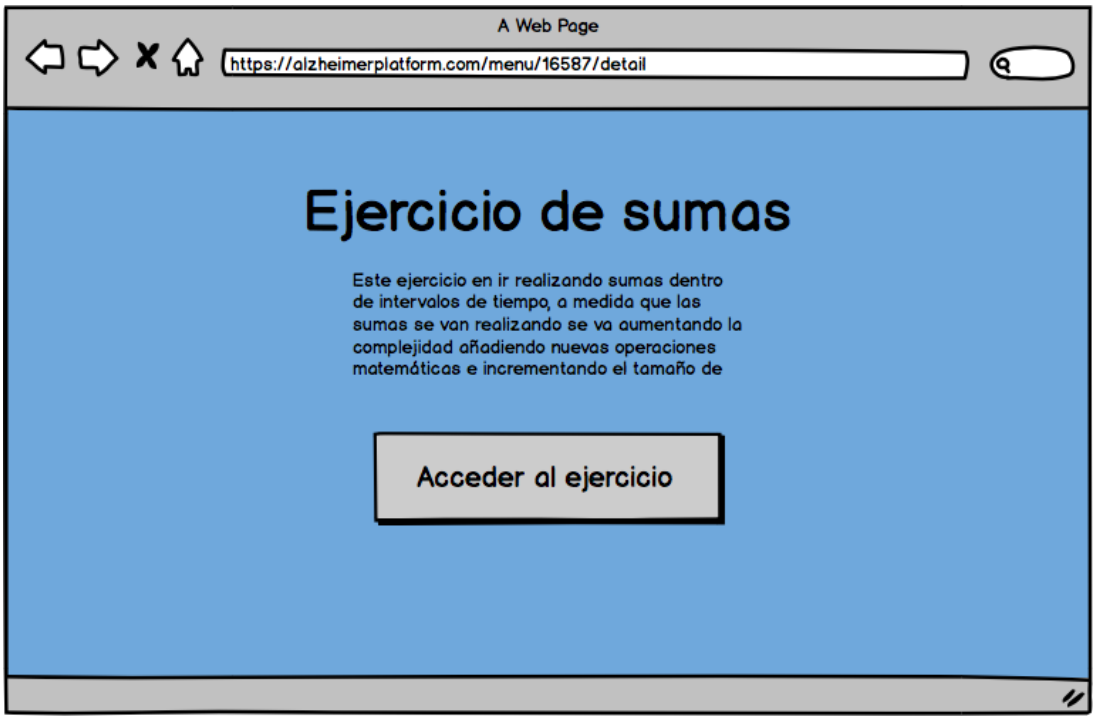


Figura 7.8: Diseño de interfaz para acceder a un ejercicio dentro de la plataforma.

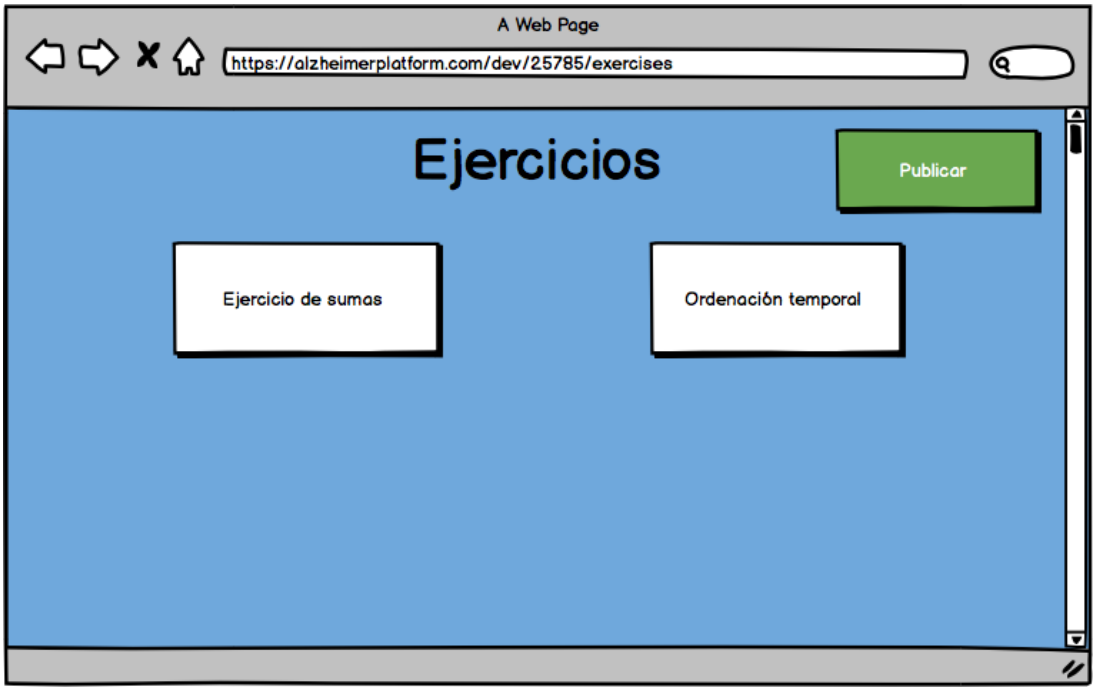


Figura 7.9: Diseño de interfaz para el conjunto de ejercicio publicado por un desarrollador dentro de la plataforma.

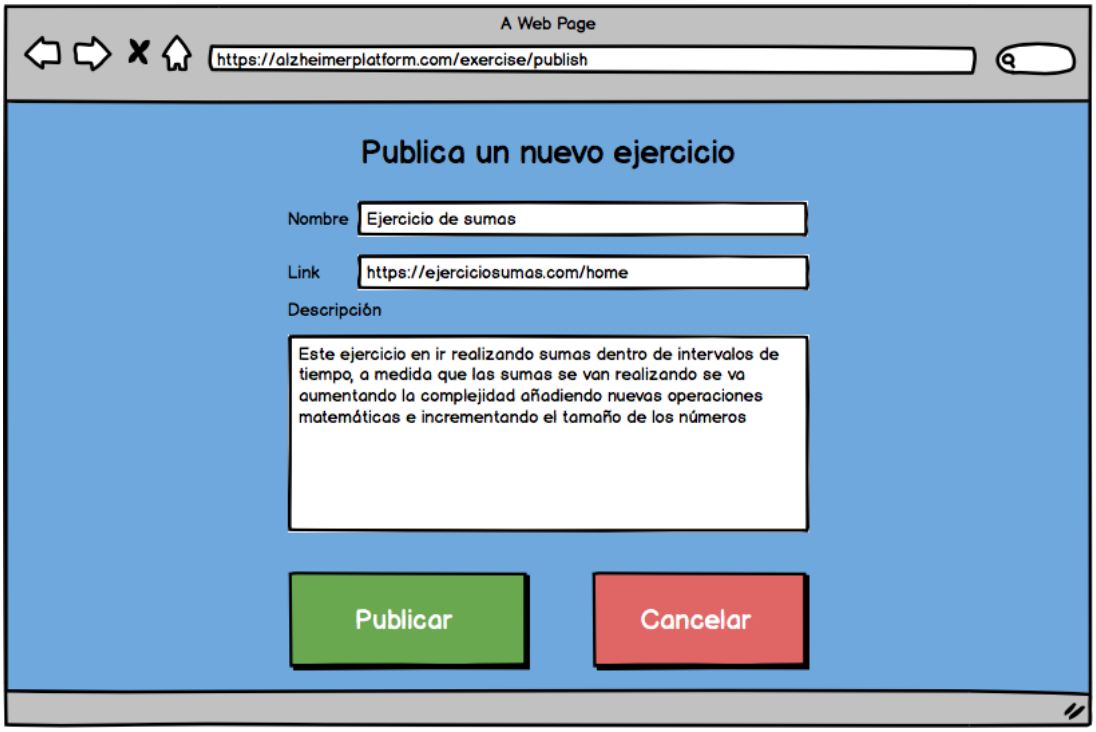


Figura 7.10: Diseño de interfaz para la publicación de ejercicios dentro de la plataforma.

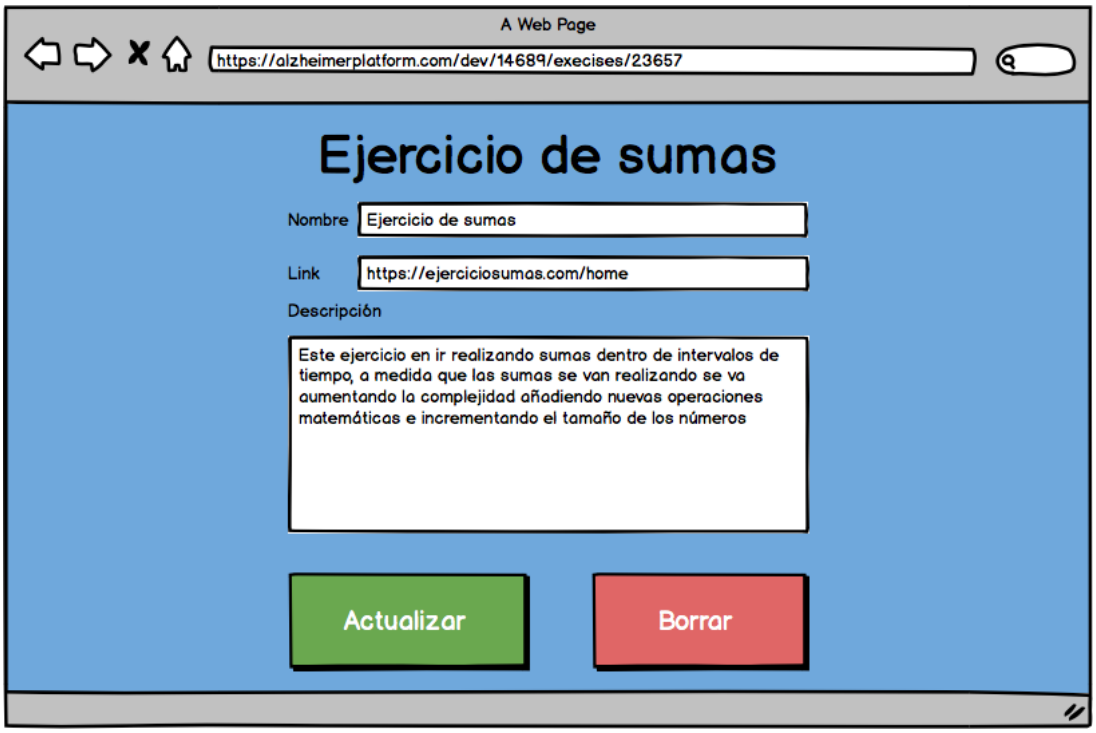


Figura 7.11: Diseño de interfaz para la edición o borrado de ejercicios dentro de la plataforma.

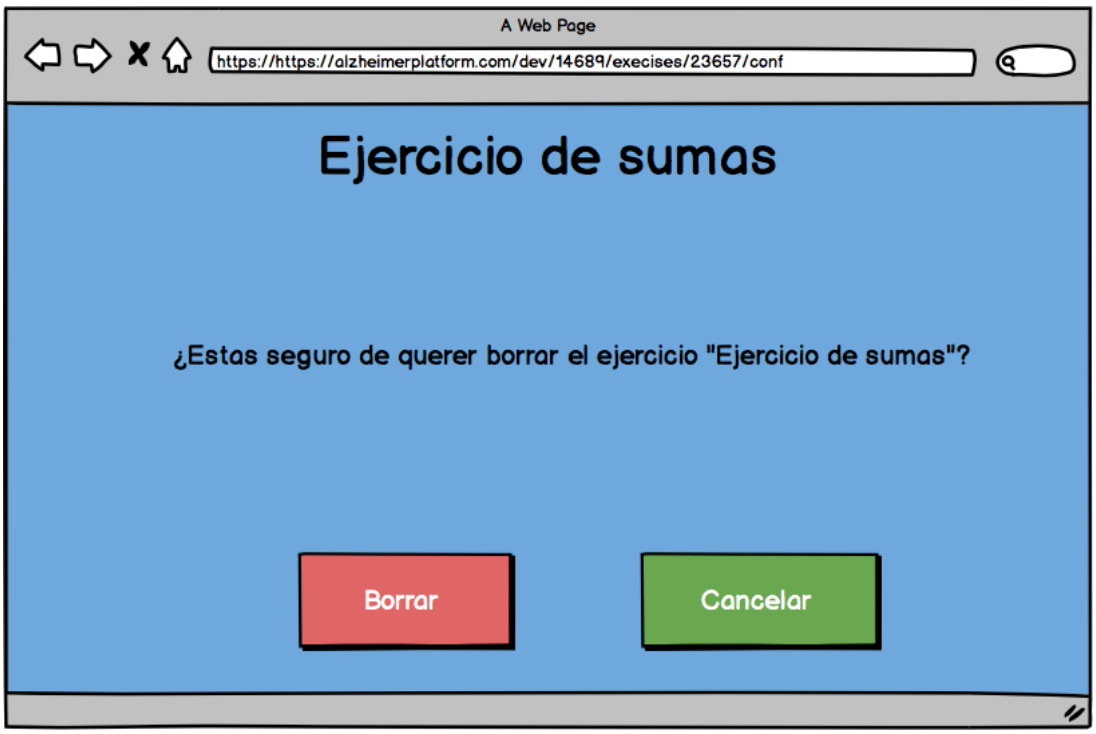


Figura 7.12: Diseño de interfaz para la confirmación de borrado de ejercicios dentro de la plataforma.

*Capítulo 8*

# Desarrollo

Dentro de este capítulo se relatan los puntos que han influido en el desarrollo del proyecto así como decisiones técnicas tomadas durante el proyecto.

Todo el desarrollo del proyecto esta accesible en los repositorios:

- https://gitlab.com/sparta015/alzheimer-platform-frontend
- https://gitlab.com/sparta015/alzheimer-platform-backend

## 8.1. Influencia del Lean Startup

Añadir un punto de vista económico dentro del desarrollo ha sido realmente útil. Pensar en que durante el desarrollo del proyecto los recursos son limitados y debemos enfocarnos en funcionalidades que realmente aporten valor al usuario, o lo que es lo mismo, por las que estuviera dispuesto a pagar el precio de nuestro producto.

Por ese motivo, durante el desarrollo del proyecto se ha tratado de aplicar la metodología Lean Startup [16] y extraer elementos positivos que reforzaran el proyecto.

La metodología Lean Startup fue desarrollada por Eric Ries en Silicon Valley en el año 2008 teniendo en mente las principales empresas tecnológica que estaban desarrollándose y triunfando dentro del valle. Esta metodología busca aplicar el método científico estableciendo una hipótesis, diseñando un experimento, determinando las métricas para validar la veracidad de la hipótesis y aplicando las lecciones aprendidas sobre el producto para volver a iterar.

Lean Startup busca entender a los usuarios, sus motivaciones y sus verdaderas necesidades

También incorpora un elemento importante y es la creación de *minimum viable product (MVP)*. El diseño de un producto basado en la idea del MVP permite otorgar al usuario un producto que cumple su necesidad principal lo antes posible y mediante la realización de los experimentos mencionados anteriormente recibir *feedback* lo antes posible.

Entrar en contacto con el usuario pronto es vital para redirigir nuestro desarrollo en caso de que no se estén cumpliendo las necesidades principales del usuario.

En el caso de este proyecto, los usuarios sobre los que se ha centrado este primer *MVP* son los desarrolladores, investigadores y sobre la figura del cuidador. Es con este conjunto de usuarios con los que se ha interaccionado para ver que podría ofrecer la plataforma que fuera de interés para ellos.

Los contactos con cuidadores e investigadores se dieron a través de entrevistas y visitas a centro para personas de la tercera edad así como centros de día en los que se proporcionan cuidados y se realizan actividades con pacientes. También mediante el contacto con familias que tienen a un paciente de Alzheimer bajo su cuidado.

El paciente, y desarrollar funcionalidades para él, es una pieza angular del proyecto pero debido a que la herramienta de interacción con el paciente son los ejercicios, no se ha podido considerar para esta primera iteración del proyecto.

Se plantearon una serie de hipótesis sobre el conjunto de usuarios al que se orientó el primer MVP:

- Hipótesis Investigador: Obtener datos del avance de la enfermedad en grupos de población sería útil para detectar patrones y entender mejor la patología.
- Hipótesis Desarrollador: Tener un espacio único en el que publicar ejercicios me facilita el desarrollo.
- Hipótesis Desarrollador: Realizar la interacción a través de la interfaz web me es más amigable.
- Hipótesis Cuidador: Disponer de un lugar en el que acceder a un amplio conjunto de ejercicios me facilita el cuidado del paciente.

El experimento diseñado para los tres grupos de interés fue muy similar: Realizar entrevistas tratando de validar dicha hipótesis, tratando de entender la problemática que afrontaba cada conjunto de usuarios y finalmente formulando la hipótesis en forma de pregunta.

La métrica seleccionada para dar la hipótesis por buena es obtener más de un 60 % de respuestas afirmativas a la pregunta realizada dentro de la entrevista sobre la hipótesis.

El ejemplo más extendido para transmitir esta idea es el caso mostrado en la Figura 8.1. En esta imagen se muestra el caso de uso en el que tenemos que satisfacer la necesidad del usuario de poder desplazarse del punto A al punto B.

Una posible solución sería ofrecer al usuario el último modelo de una conocida marca de coches eléctricos y autónomos. En este caso, pasaría mucho tiempo y se gastaría mucho dinero hasta poder entregar el producto final al usuario. Y en el peor de los casos, el usuario podría preferir otro tipo de vehículo y toda la inversión se habría perdido.

Si por el contrario, en una primera instancia entregamos un patinete al usuario, en un tiempo muy reducido y con un coste muy acotado, ya habríamos conseguido satisfacer la necesidad de nuestro usuario. Entrar tan pronto en contacto con el usuario, nos permite conocer otras

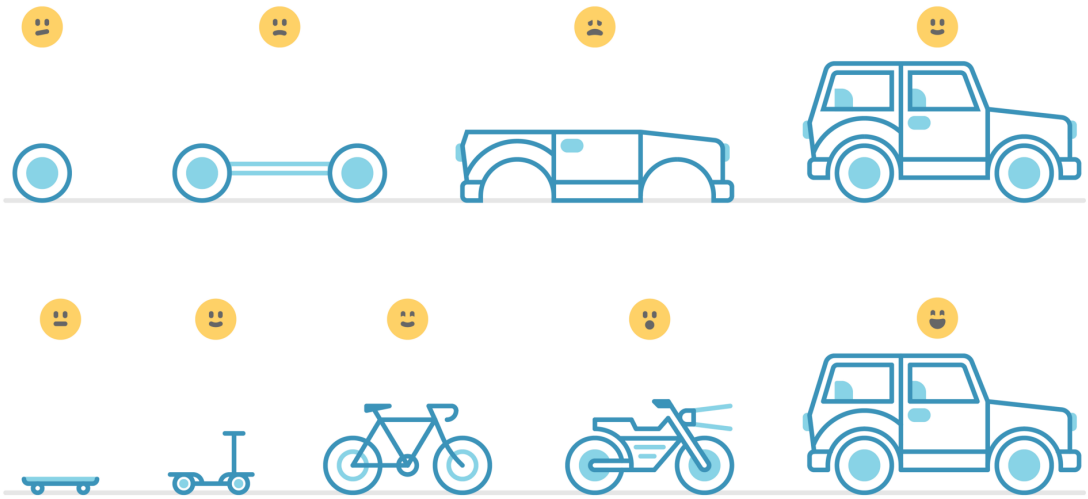

Figura 8.1: Gráfico sobre el desarrollo del producto mínimo viable. [16]

necesidades subsidiarias. En el ejemplo, esta necesidad podría ser que el usuario se cansa mucho al recorrer grandes distancias con el patinete, en ese caso, podríamos evolucionar el producto y entregar una bicicleta.

Esta evolución constante nos permite ahorrar dinero, mejorar la relación con el cliente y reducir nuestros tiempos de desarrollo.

¿Cómo aplica esta metodología al desarrollo del proyecto?

En este caso se han identificado 2 usuarios principales de la plataforma:

- Desarrolladores
- Investigadores o Médicos

Los pacientes de Alzheimer son un grupo importante para el proyecto pero, como se ha indicado antes, su interacción y principal propuesta de valor es la posibilidad de realizar ejercicios dentro de la plataforma, y para esta primera *Release* no se dispone de juegos.

Por lo tanto, es vital orientar el desarrollo hacia un producto mínimo viable que permita, por una parte, a los desarrolladores publicar y gestionar ejercicios dentro de la plataforma y, por otra parte, proporcionar una manera de visualizar los datos obtenidos durante la realización de los ejercicios por parte de los pacientes.

Debido a ello, el desarrollo se ha centrado en:

- Integrar Grafana con Postgres. Dentro de este apartado se han desarrollado las *queries* necesarias para mostrar los datos deseados y creado los *dashboards* dentro de Grafana para poder mostrar dicha información de manera amigable para los investigadores y médicos.

- Proporcionar un API REST sobre Play Framework para la gestión de ejercicios dentro de la plataforma. El primar el desarrollo de un API REST frente al interfaz web, se debe a que, en las entrevistas los desarrolladores mostraron mayor interés en dicha interfaz REST.

Una plataforma con esos dos componentes es el producto mínimo viable. Este producto puede ir evolucionando y proporcionando nueva funcionalidad en la medida que se cumplan estos dos objetivos.

Hay un conjunto de funcionalidades como el inicio de sesión, o el registro de usuarios que proporcionan un valor añadido al proyecto y que serán priorizados para conseguir integrarlos dentro del mismo.

Por lo indicado anteriormente el peso de desarrollo se ha centrado en el desarrollo del microservicio de gestión de ejercicios y la integración de Postgres con Grafana.

## 8.2. Clean Code

Uno de los principios que han regido todo el proceso de desarrollo es la realización de *Clean Code*.

El *Clean Code* es explicado por Robert Martin en su libro Çlean Code: A Handbook of Agile Software Craftmanship" [12]. En este libro el autor explica qué características tiene el código limpio y proporciona pautas para alcanzar este tipo de desarrollo.

El dicho libro [12] de Robert Martin proporciona un amplio número de guías, principios y patrones para alcanzar un desarrollo de código limpio completo. Para el desarrollo del proyecto se han seguido algunas de ellas.

El código limpio se caracteriza por ser elegante, eficaz, legible, mínimo y hacer una cosa bien de un único modo.

Un desarrollador pasa más tiempo leyendo código que escribiendo código, hay métricas que indican que esta proporción es de 10:1. Si el código que leemos está mal escrito, tendremos menos tiempo para desarrollar nuevo código y seremos más propensos a caer en malas prácticas.

Si atendemos al ciclo de vida que experimenta el código, la mayor parte de este tiempo es en la fase de mantenimiento. Tratar de mantener un código ilegible genera una gran perdida de tiempo y de dinero, así como frustración en el desarrollador que tenga que mantener dicho código. Por ello realizar código limpio es vital para poder avanzar en un proyecto.

Dentro del proyecto y como indica el libro de Robert Martin [12], se ha prestado especial atención a elementos como los nombres o los comentarios.

Un nombre tanto de una variable, clase o función, debe ser totalmente intencionado y descriptivo. Es necesario evitar palabras redundantes (info, data, etc) o formados por una única letra como a, e, k, j ya que dificulta realizar búsquedas dentro del código y hace más difícil entender su funcionamiento en una lectura rápida del código. Para el caso de bucles y en un contexto muy acotado si se utilizaron las variables con nombre i, j, k.

Dentro del proyecto, se ha reducido el número de comentarios dentro del código. Como indica Robert Martin, los comentarios solo están justificados cuando no somos capaces de expresarnos con el código. Es necesario evitar realizar comentarios redundantes que expliquen el comportamiento del código.

Sin embargo, dentro del proyecto sí que se han mantenido comentarios referentes a explicaciones de decisiones tomadas y para comentarios de tipo TODO.

Dentro del libro, se hace referencia a distintos patrones y principios, muchos de ellos recogidos en los principios SOLID [10].

A continuación se enumeran ciertos principios tomados del Clean Code [12] que se han seguido durante el desarrollo:

- *DRY Principle (Don't repeat Yourself)*, este principio, que reside también en el core de Play Framework, busca eliminar la duplicidad dentro del código. El autor propone distintas alternativas como la refactorización, el uso de patrones de diseño, la herencia o la composición. Dentro del desarrollo, en los distintos *sprints* se ha dedicado especial atención a la posibilidad de refactorizar el código realizado durante el sprint anterior al añadir nueva funcionalidad o que quedara marcado como deuda técnica.

- *The Principle of Least Surprise*, parece obvio lo que este principio busca y es que las funciones y clases deben hacer lo que se espera de ellas. Este comportamiento debe estar descrito dentro de su nombre, evitando así que sea necesario entrar dentro del código para saber qué función realiza.

- *Single Responsibility Principle*, el más difícil de cumplir y el que ha llevado un mayor grado de refactorización. Este principio lo que indica es üna clase debe tener una y solo una razón para cambiar". Esto puede ser entendido como una única responsabilidad. Las funciones de la clase deben girar en torno a un mismo eje, un mismo propósito. Si una clase atiende a varios propósitos, como por ejemplo obtener qué ejercicios ha realizado el paciente durante los últimos 7 días y la puntuación obtenida dentro de esos, esa clase debe ser dividida y crear una que obtenga el histórico y otra que dado un ejercicio proporcione información sobre el mismo.

- *Boy Scout Principle*, este principio expresa la necesidad de dejar el código mejor de lo que el desarrollador se lo encuentra. Este principio cobra especial importancia cuando se trabaja dentro de un equipo. En el caso de este proyecto, dado que el desarrollo del código se ha alargado en el tiempo, meses después del desarrollo de una funcionalidad tras realizar una nueva funcionalidad dentro de una clase, con la experiencia adquirida, se han detectado anti-patrones que han sido eliminados realizando una refactorización sobre dicha clase.

Es imposible realizar código limpio a la primera, por eso es necesario refactorizar el código cuando funcione. En el caso de este proyecto las tareas de refactorización se han tomado como deuda técnica que era abordada en las iteraciones sucesivas.

## 8.3. Test-Driven Development

Debido a la importancia que tienen los tests en el desarrollo de código limpio, para el proyecto se ha decido seguir *Test-Driven Development* [2].

El *Test-driven development* (TDD) es una práctica de ingeniería del software que involucra dos procesos:

- Escribir las pruebas antes que el código para de este modo, entender como se quiere que se realice la petición y como interactuarán los componentes entre sí.
- Refactorizar

Esta metodología determina que primero es necesario escribir las pruebas, ya sean pruebas unitarias o de integración. Obligar al desarrollador a escribir las pruebas antes que el código, implica es necesaria una fase de análisis previa y determinar el comportamiento que tendrán los componentes y la petición, para así poder desarrollar la prueba.

A continuación, se desarrolla el código correspondiente a dichas pruebas y se comprueba que se satisfacen. Cuando el resultado es satisfactorio, se refactoriza el código escrito para evitar duplicidad y malas prácticas en general.

La metodología TDD proporciona 3 leyes:

- No hay que crear código hasta que se produzca un fallo dentro de un test unitario.
- No hay que crear nunca más de una prueba que falle.
- El código destinado a cubrir la prueba con éxito debe ser mínimo.

El seguimiento de estas tres leyes durante la realización del proyecto ha permitido una evolución paralela del código y los test. Garantizando así la entrega de funcionalidad con la seguridad de que el comportamiento del código es el esperado.

Durante el desarrollo de las pruebas se ha buscado cumplir con el principio F.I.R.S.T [11]:

- **Fast**, los tests deben ser rápidos en su ejecución. De lo contrario, sería posible que al desarrollador le diera pereza ejecutar los test y no los lance con la frecuencia necesaria. De no ejecutar los tests con cada cambio añadido, podría incluirse un *bug* dentro del código y no detectarse hasta pasado un tiempo, lo que dificultaría su resolución.
- **Independent**, cada test debe ser independiente del resto. El resultado de la ejecución de un test no puede condicionar el de los siguientes, este hecho implica además que los test

pueden ejecutarse sin un orden concreto. No seguir esta práctica garantiza que en caso de que se genere un *bug*, no sea trivial detectar en que parte del código se produjo, ya que no fallará un test, sino todos los que tengan dependencias entre ellos.

- **Repetition**, es necesario desacoplar la ejecución de los test del entorno sobre el que se realizan. De otro modo, en caso de fallo podría atribuirse a la ejecución dentro de otro entorno. Dentro de este proyecto, al ejecutar los tests tras la compilación, en la generación de la imagen Docker, garantizamos que siempre se realizan en el mismo entorno aislado por el contenedor.

- **Self-Validating**, la salida de los tests debe ser autoexplicativa, debe ser un valor booleano. Los test deben pasar o no, pero el resultado de un test no debe ser validado posteriormente de manera manual. Este principio sigue alineado con la ejecución efectiva de los test y la velocidad para obtener el estado en el que se encuentra el código.

- **Timely**, este último principio aplica una restricción sobre el modo de desarrollar obligando al desarrollador a realizar primero los tests antes del código. Esto se debe a que si postergásemos la realización de los tests, podrían acumularse varias funcionalidades y sería difícil de encontrar la causa del fallo en caso de que se presentara. Retrasar la realización de los test puede llevar también a la mala práctica de no desarrollarlos en favor de nuevas funcionalidades, esto último sería un anti-patrón total que debe evitarse a toda costa. La realización de los tests antes que el código también tiene un efecto colateral sobre el código ya que obliga al desarrollador a pensar antes de lanzarse a desarrollar y también impacta en el desarrollo haciendo que las funciones sean más atómicas.

## 8.4. Implementación del modelo asíncrono

En el capítulo de arquitectura, se ha hablado sobre la importancia del código asíncrono dentro de un sistema real. En esta sección, se pretende clarificar cómo se ha abordado desde la perspectiva del desarrollo, librerías utilizadas y su uso para la resolución de los problemas presentados.

Java 8 proporciona una API para lidiar con todos los problemas relacionados las llamadas asíncronas, cuando se recibe una llamada a un método que tiene que acceder a persistencia o tiene que conectar con otro servidor. En ambos casos, el código podría bloquear el hilo de ejecución hasta recibir una respuesta o poder acceder a los datos y a continuación retornar la respuesta. La propuesta de Java para evitar este problema es el API de Concurrencia.

La clase que implementa estas funcionalidades es la clase CompletableFuture [13], concreción de la interfaz CompletableStage. Está interfaz define un contrato para realizar un paso asíncrono y que permite ser combinado con sucesivos pasos.

La clase CompletableFuture implementa la interfaz Future incluida en Java 5 y completa esa funcionalidad con más método que hacen más usable la clase.

Dentro de la solución planteada se ha tratado de primar el uso de la interfaz CompletableStage frente a la instanciación de la clase CompletableFuture para mantener el código desacoplado de la implementación de la *thread pool*.

Para procesar el resultado de una llamada asíncrona dentro del desarrollo se ha encapsulado la ejecución dentro de funciones anónimas.

Las funciones anónimas son una función definida sin identificador, se trata de una función anidada y esto le permite acceder a las variables definidas dentro del *scope* de la función en la que se define. El tipo retornado por la función anónima varía en caso de que el cuerpo tenga una única línea, en cuyo caso el tipo será el mismo que el del cuerpo, y en el caso de que tenga más de una línea, el tipo será el retornado por la función y en caso de no posea *return* el tipo será *void*.

Dentro del desarrollo se han utilizado principalmente 2 funciones proporcionadas por la interfaz CompletableStage:

- supplyAsync(), esta función retorna un nuevo CompletableFuture que es es completado asíncronamente por la tarea que se está ejecutando dentro de la *pool* de hilos con el valor obtenido de ejecutar la función lambda definida en su interior.

- thenApplyAsync(), esta función retorna un CompletionStage que en caso de terminar correctamente permite tomar el valor previo a esta llamada como argumento para ejecutarse.

En las llamadas a estas funciones es necesario proporcionar un contexto para tenga los atributos necesarios a la hora de ejecutarse dentro del nuevo hilo.

La implementación realizada se puede consultar en cualquiera de las clases dentro de https://gitlab.com/sparta015/alzheimer-platform-backend/tree/master/backend/app/repository

## 8.5. Integración con Grafana

Para poder mostrar datos a través de Grafana es necesario indicar al servicio a qué base de datos se puede conectar. En el caso de la plataforma la conexión será sobre la base de datos Postgres.

Para poder leer los datos almacenados en Postgres, es necesario que el usuario destinado para ese propósito solo tenga permisos de lectura, para evitar que los datos puedan ser alterados. Para conseguir esto dentro de la creación de la tabla de Investigadores, se han modificado sus permisos dentro de Postgres.

Para la visualización de datos son necesarios dos pasos:

- Creación de dashboard, es necesario crear los dashboards donde se insertarán los gráficos y las tablas. Estos dashboard pueden ser creados de manera manual o mediante la carga de un

fichero JSON. Para hacer la plataforma escalable y no tener que estar siempre manteniendo este punto, se ha determinado extraer el fichero JSON del dashboard generado y cargarle dentro de un volumen de datos del contenedor Docker. Para que, de este modo, cada vez que se reinicie el servicio, estén disponibles los dashboards.

- Realizar las *queries* sobre la base de datos Postgres para mostrar estos valores. Estas *queries* se realizan del mismo modo que sobre una base de datos tradicional, con la mejora añadida de que Grafana motrará en formato gráfico los datos obtenidos.

Durante el desarrollo de la solución se ha detectado que a pesar de que Grafana aporta un gran número de funcionalidades sobre la visualización de datos y que a permitido avanzar muy rápido sobre la historia de usuario de investigar, no es la mejor solución para la visualización de datos si se tiene una amplia base de usuarios. Los principales problemas que plantea son la concurrencia sobre las búsquedas y los permisos sobre los usuarios.

## 8.6. Implementación del API REST

Para el desarrollo de la plataforma se han desarrollado 3 microservicios que expone cada uno de ellos un API REST con las operaciones disponibles.

La idea tras este planteamiento es desacoplar la funcionalidad básica proporcionada por la plataforma de la presentación de datos. Esto se debe a que para el registro o inicio de sesión puede ser útil disponer de una interfaz web para visualizar estos datos. Pero en contra punto, disponer de las operaciones necesarias para publicar, actualizar o borrar un ejercicio mediante operaciones CRUD que pueden ser realizadas desde una terminal con curl puede ser una interfaz más cómoda para el desarrollador.

Para la implementación de cada una de las API REST correspondientes a los servicios de login, registro de usuarios o administración de ejercicios se ha utilizado Play Framework, ya que, como se ha comentado previamente, permite hacer un desarrollo ágil e incorpora la programación asíncrona con Java 8 de manera natural.

Cada microservicio en Play Framework cuenta con un fichero de rutas en el que se definen los distintos endpoint que estarán disponibles para recibir peticiones. En cada una de las entradas del fichero correspondientes a las url, se especifica la operación que se realizará sobre ellos (uno de los verbos correspondientes a las operaciones dentro de las API REST).

Dentro de la implementación de este API REST, se crean unas clases Java las cuales hacen la función de controlador y puerta de entrada para las peticiones que recibe el microservicio. Los métodos disponibles dentro de estos controladores, coinciden con los endpoints definidos dentro del fichero de rutas.

Siguiendo el patrón Repository, definido dentro del capítulo de arquitectura, para el acceso a persistencia se implementa una clase Repository por servicio. En el caso concreto del servicio de

Registro se cuenta con tres clases Repository, se tomo esta decisión para conseguir mantener desacoplados los tres conjuntos de usuarios.

Dentro de los métodos dentro de las clases Repository se utilizan los métodos asíncronos proporcionados por la interfaz CompletableStage de Java 8.

En los métodos definidos dentro de los controladores, una vez recibida una petición se genera un objeto en correspondiente a una de las clases del Modelo en base a unos datos de entrada en formato JSON.

## 8.7. Arquitectura de datos

La jerarquía de herencia dentro del modelo de datos de la aplicación se ha resuelto dividiendo los atributos en varias tablas (una para cada tipo de datos). Las tablas que hacen referencia a las clases "hijas" tienen una clave foránea referenciando a la tabla de la clase "padre".

Para recuperar los datos correspondientes a las clases hijas tendremos que realizar un *join* de dichas tablas.

La estructura del modelo de datos sería la siguiente la que se muestra en la Figura 8.2

Otra posible solución habría sido utilizar una única tabla para las clases afectadas por la herencia. De esta manera, la tabla correspondiente a la clase Usuario tendría todos los atributos y en caso de que quisiéramos rescatar datos correspondientes a un paciente, tendríamos a *null* las columnas propias de un desarrollador.

Esta solución a pesar de ser más más óptima a nivel de operaciones sobre la base de datos y más sencilla de implementar, resulta menos clara y dificulta el mantenimiento de la aplicación a largo plazo. Así como la presencia de atributos nulos dentro de la aplicación que podrían provocar *NullPointerException* dentro del código.

## 8.8. Despliegue

Para el despliegue de toda la plataforma Figura 6.2 se han aislado los distintos servicios dentro de contenedores Docker.

Esta decisión se debe a que permite emular el despliegue que se haría dentro de un *cluster*, en que cada servicio corre en un nodo computacional distinto y sus replicas están repartidas entre el resto de los nodos.

Otra de las ventajas de Docker y que a propiciado su elección es que aporta la ventaja de aislar al servicio del sistema operativo sobre el que corre el contenedor. Esto ha sido de utilidad al permitir desarrollar los servicios sobre MacOS y poder luego desplegarlos sobre las máquinas del laboratorio.

También nos permite no tener que compilar todo el código del repositorio cada vez que queremos desplegar un servicio, sino que podemos descargar la imagen Docker del registro que proporciona Docker y agilizar los despliegues.

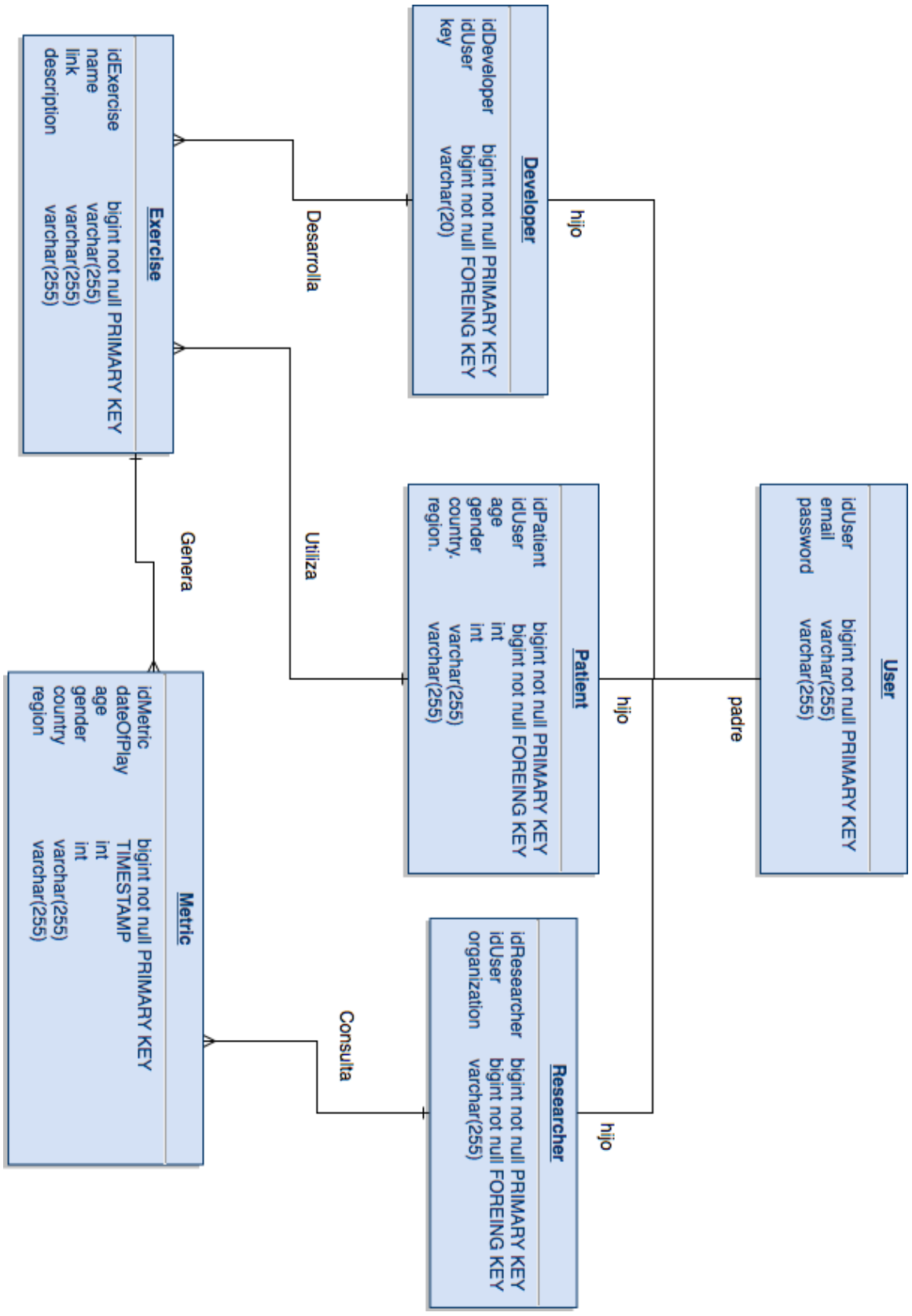


Figura 8.2: Diagrama Entidad Relación.

Durante el despliegue se toma ventaja de las imágenes intermedias generadas durante el creación de las imágenes Docker de los servicios de Login, Registro y Administración de servicio a partir de sus DockerFiles.Lo cual reduce el tiempo necesario para la construcción de estas imágenes.

*Capítulo 9*

# Testing

La calidad es un elemento fundamental en el desarrollo del software que determina el éxito que éste tendrá cuando llegue al usuario final. Dentro de los equipos es habitual contar con personas especializadas para garantizar la calidad del software.

En la industria era una práctica habitual destinar una fase del proyecto a las pruebas sobre el código una vez desarrollado. En la actualizad esto está cambiando, pivotando hacia un modelo mixto en el que las pruebas están integradas dentro de la fase de desarrollo, esta segunda corriente es la que se ha seguido en el desarrollo de este proyecto.

Normalmente los tests los realiza una persona distinta a la que va a realizar el código, para así, garantizar encontrar errores que de otra manera no habrían sido detectados por el propio desarrollador.

Actualmente, dentro de la industria, se está tendiendo a incrementar la responsabilidad del desarrollador sobre el código que realiza y las pruebas sobre el mismo o lo que es lo mismo, la calidad del software. De este modo, dado que el desarrollador escribe los tests que pasará su código en primera instancia, es más consciente de la calidad y mantiene alto el nivel de la misma.

Esto no significa la desaparición de los equipos de QA (*Quality assurance*), sino que, éstos son responsables de desarrollar baterías de tests paralelas que se pasarán como parte del proceso de integración continua del código. Así mismo, tienen el papel de guiar al equipo de desarrollo en la realización de las pruebas, especialmente con las pruebas de integración o las pruebas exploratorias.

Es una práctica habitual el desarrollo de pruebas de caja negra y caja blanca. En las pruebas de caja negra, el desarrollador que escribe el test no ha visto aún el código desarrollado, por lo que se hacen dichas pruebas en función de los requisitos. Por otro lado, las pruebas de caja blanca, se hacen en conjunto con el desarrollador que ha escrito el código, buscando errores más relacionados con los casos frontera.

En este caso, dado que solo hay un desarrollador, la manera que se ha encontrado de conseguir estas pruebas de caja negra y caja blanca, es dividiendo la fase de desarrollo de pruebas en dos: las pruebas de caja negra serán escritas antes que el código y las pruebas de caja blanca una vez

se haya escrito el código. No es una solución al 100 % ya que es el mismo desarrollador el que escribe todas las pruebas, pero es una buena aproximación.

Dado que se ha priorizado el desarrollo del API REST que da funcionalidad a la plataforma, se han centrado las pruebas sobre este componente.

## 9.1. Servidor

Para la parte del servidor hemos seguido la metodología de *Test-driven development* (TDD) como se ha descrito en el capítulo de Desarrollo.

En una primera instancia, dado que se pretendía desarrollar primero la interacción con la base de datos y la correcta escritura y lectura de datos en la base de datos, se realizaron un primera instancia pruebas unitarias en las que se comprobaba que el código procedente del Repository era capaz de acceder a la base de datos y devolver objetos equivalentes a los generados con el constructor de la clase en el modelo.

Para lanzar los test, desde la consola de sbt es necesario ejecutar:

- **clean**, para eliminar datos de ejecuciones anteriores
- **compile**, para compilar el código y asegurar que todo es correcto
- **test**, para pasar las baterías de test que se han fijado sobre cada servicio.

### 9.1.1. Pruebas unitarias

Las pruebas unitarias nos permiten probar cada método de manera aislada. Para su correcta realización, debemos escribir una prueba que valide el funcionamiento correcto, lo que comúnmente se conoce como *happy path*. A continuación, debemos escribir el resto de pruebas sobre la funcionalidad que comprueben un caso de error como puede ser un campo que no debe ser vacío o que no debe superar un número determinado de caracteres.

Un error habitual al realizar este tipo de test es englobar en un test más de un caso de error, esto puede llevar a ocultar errores dentro de la aplicación, por lo que es recomendable ser lo más atómico posible a la hora de escribir estas pruebas.

Al probar operaciones sobre la base de datos, nos centraremos en las cuatro operaciones que puede realizar un API REST, esto es, escritura, lectura, actualización y borrado, que se corresponden con las operaciones CRUD. Buscando para estos casos una situación de éxito y explorando los posibles errores que podrían darse en estas operaciones para poder controlarlos después en el código.

Los requisitos impuestos por el modelo de datos, como los campos obligatorios o las longitudes de algunos campos, han sido foco de atención en la búsqueda de casos frontera.

El proceso de realización y mejora de estas pruebas es iterativo, es decir, como el carácter de las pruebas es destructivo, en una primera ejecución de la batería de tests se espera encontrar un

gran número de errores. Posteriormente, estos errores son corregidos en el código para después, volver a pasar la batería de pruebas sobre el conjunto de código. En esta nueva comprobación se busca verificar que se ha solucionado el error y que no se han generado otros nuevos con esta solución.

Para ilustrar este proceso, se ha seleccionado el código del servicio para el API de administración de ejercicios.

En un primer lugar se quería abordar la funcionalidad correspondiente a obtener todos los ejercicios, por lo que se comenzó escribiendo un test para acceder a la capa de persistencia y recuperar todos los valores en la base de datos.

Como paso previo al test, se realiza una inserción en la base de datos. Como parte del test, se obtienen todas las entidades dentro de la tabla ejercicios. Finalmente se hace una búsqueda en el *array* por el identificador del valor usado en la inserción.

En una primera ejecución este test devolverá un valor de fallo, esto recordemos que es positivo, porque el carácter de las pruebas es destructivo. Este fallo se debe a que aún no está implementada la lógica dentro de la llamada en la clase RepositoryDeveloper.java.

Una vez realizada la implementación, se vuelve a lanzar la batería de test para determinar si se ha resuelto el problema. En este caso, con el código desarrollado la respuesta del test fue positiva.

Durante la realización de estos test se detectó que no se estaba validando la inserción de valores nulos dentro de la base de datos. Esto se debe a que en el primer diseño los datos venían validados desde la aplicación cliente y solo se generaban a través de la interfaz web por lo que fue necesario desarrollar unos test para validar que se estaba produciendo esa vulnerabilidad y solucionarla.

En los test, se trataba de insertar dentro de la base de datos un juego con diversos campos nulos, tras hacer la inserción, se recuperaba el valor y se comprobaba si alguno de sus campos era nulo. La condición de éxito del test era el lanzamiento de una excepción. Durante la primera ejecución del test, no se lanzo ninguna excepción de tipo INVALID ARGUMENT por lo que el test falló.

Se modificó el código para añadir una validación previa a la inserción que comprobará que no hubiera campos nulos dentro del objeto y de ser así, dispara una excepción de tipo INVALID ARGUMENT. Tras lanzar de nuevo los test sobre el servicio, el resultado fue positivo para esta prueba.

*Capítulo 10*

# Conclusiones y trabajo futuro

## 10.1. Análisis de la consecución de los objetivos

Dentro de esta sección se analizará el grado de complitud que se a alcanzado sobre los objetivos que se marcaban al inicio del proyecto.

### 10.1.1. Plataforma accesible para pacientes y/o cuidadores

La idea detrás de este objetivo era conseguir desarrollar una plataforma amigable para pacientes y/o cuidadores que les permitiera acceder a los ejercicios que hasta ahora en su mayor parte, solo estaban disponibles en papel.

En esta primera fase del proyecto, ese objetivo no ha podido completarse al 100 %. En un primer momento, si que se trabajo para alcanzar este objetivo, se añadieron taras relacionadas con el desarrollo de funcionalidad orientada hacia los pacientes y cuidadores y se realizó un proceso de diseño para poder realizar la implementación de esas interfaces.

La realidad es que ese código esta avanzado y disponible, pero no se le ha podido dedicar todo el tiempo que sería necesario para dar esta etapa de desarrollo como concluida. Ya que se decidió primar el desarrollo del API REST, con el objetivo de dotar de funcionalidad a la plataforma y permitir que los desarrolladores puedan integrarse con ella en un corto periodo de tiempo.

La falta de experiencia por parte del alumno con las tecnologías web fue un elemento que impactó mucho a la planificación ya que no se hizo una estimación realista del tiempo de desarrollo.

### 10.1.2. Publicación y gestión por parte de desarrolladores de ejercicios dirigidos a pacientes con diagnostico de enfermedad de Alzheimer

Este objetivo si se ha cumplido, dentro del desarrollo de la primera versión se ha conseguido tener un API REST funcional que permite a los usuarios integrarse con la plataforma, publicar y administrar juegos.

### 10.1.3. Proveer de datos y métricas sobre el Alzheimer a investigadores a través de datos obtenidos de los ejercicios de los pacientes

A falta de una colaboración más estrecha con investigadores, el desarrollo necesario por parte de la plataforma para acceder y visualizar los datos está resulto, por lo que podríamos considerar este objetivo como cumplido.

## 10.2. Lineas de trabajo futuras

Dado que aún quedan muchos puntos para mejorar y continuar a lo largo del desarrollo, este proyecto pasará a estar a disposición de la universidad para poder ser continuado por otros alumnos.

Un aspecto de mejora del proyecto y que queda fuera por la magnitud que implica, sería el desarrollo de un sistema para realizar la validación de los juegos subidos a la plataforma de manera automática.

### 10.2.1. Estandarización de contenido

Para el desarrollo de esta funcionalidad sería necesario establecer una buenas prácticas que deben ser cumplidas por los desarrolladores de cara a la publicación y así como el establecimiento de unas métricas para validar que se cumplen requisitos no funcionales como el tiempo de respuesta.

Como base para comenzar este desarrollo se podrían tomar como ejemplo los test automáticos realizados en otros proyectos sobre las interfaces web.

### 10.2.2. Figura del asistente

En caso de la plataforma tuviera un gran uso en centros de día, donde hay un mayor número de usuarios, se presenta la necesidad de poder mantener a varios usuarios con las credenciales de sesión almacenadas en la aplicación y así poder cambiar entre estos usuarios de una manera más dinámica.

Esto nos lleva a una nueva figura, “el cuidador o asistente", este nuevo objeto tendría bajo su supervisión a varios pacientes y podría vincular estos perfiles al suyo, teniendo solo que iniciar sesión una vez.

### 10.2.3. Notificaciones

Otra funcionalidad que fomentaría el uso continuado de los ejercicios dentro de la plataforma sería añadir el envió de notificaciones.

Esta funcionalidad podría desarrollarse si se utiliza el navegador web Chrome siguiendo las guías proporcionadas por Google. Este navegador es el instalado por defecto en los sistemas Android. Para los usuarios que utilicen un dispositivo IOS, podrán seguir utilizando la aplicación pero no podrán recibir estas notificaciones. Lo mismo sucederá con los usuarios que utilicen un navegador web como Firefox o Opera.

### 10.2.4. Mejora en la experiencia de usuario

Centrar una nueva *release* del proyecto en aspectos puramente funcionales del producto como ofrecer la posibilidad al paciente de tener una vista personalizada con los ejercicios más habituales y poder gestionar ese grupo de ejercicios permitiendo añadir y eliminar elementos de ese conjuntos.

Otra mejora para la experiencia de usuario sería añadir la posibilidad de buscar ejercicios dentro de la vista principal para facilitar al paciente encontrar un ejercicios concreto.

Dentro de esta *release* tendría cabida también un estudio más detallado del uso que hacen los pacientes de la plataforma así como sus principales errores para tratar de resolverlos.

### 10.2.5. Mejoras en arquitectura

De cara a aumentar el rendimiento de la plataforma, hay cambios que se pueden realizar sobre la arquitectura:

#### Publicadores y Subscriptores

Un posible cambio sería desacoplar el proceso de publicación de ejercicios y su validación. Adaptando la arquitectura de la aplicación a un conjunto de elementos productores y consumidores.

En este tipo de arquitecturas se establecen un conjunto de productores (servicios encargados de generar contenido) que escriben en un sistema de colas, para que posteriormente los subscriptores (servicios que leen datos de dicho sistema de colas) recuperen esa información de forma asíncrona para completar sus servicios.

La posición de lectura desde la que deben consumir los subscriptores información dentro de la cola se establece a través de un *offset*. Para permitir un comportamiento asíncrono y concurrencia de varios consumidores sobre un único publicador, se establece un *topic*.

Este topic, es el elemento sobre el que hay que

Para satisfacer este sistema, sería necesario añadir un nuevo componente sobre la arquitectura que permitiera soportar esta arquitectura basada en colas como puede ser RabbitMQ, Redis o Kafka.

Evolucionar el sistema hacía una arquitectura basada en colas permitiría redefinir las métricas actualmente almacenadas en el sistema como series temporales.

# Bibliografía


[1] A. Association, "Datos y cifras sobre la enfermedad del alzheimer en 2018," https://www.alz.org/alzheimers-dementia/facts-figures.

[2] K. Beck, "Test-driven development by example," 2003.

[3] S. M. body of knowledge, "Estimación de poker," 17 Octubre 2016, https://www.scrummanager.net/bok/index.php?title=Estimación_de_póquer.

[4] M. Cohn, "Estimating with tee shit size," 2 de Abril 2013, https://www.mountaingoatsoftware.com/blog/estimating-with-tee-shirt-sizes.

[5] A. disease international, "Risk factors," https://www.alz.co.uk/info/risk-factors.

[6] R. T. Fielding, "Achitectural styles and the design of the network-based software architectures," 2000.

[7] M. Fowler, "Patterns of Enterprise Application Architecture," 5 de noviembre de 2002.

[8] P. Framework, "Understanding play thread pools," https://www.playframework.com/documentation/2.6.x/ThreadPoolsUnderstanding-Play-thread-pools.

[9] ——, "Akka HTTP Implementation," 2018, https://www.playframework.com/documentation/2.6.x/AkkaHttpServer#Akka-HTTP-Implementation.

[10] R. C. Martin, "Getting a solid start," 02 de Diciembre de 2009, https://sites.google.com/site/unclebobconsultingllc/getting-a-solid-start.

[11] ——, "Clean code: A handbook of agile software craftsmanship," 17 de Julio de 2008.

[12] ——, "Clean code: A handbook of agile software craftsmanship," 17/07/2008.

[13] Oracle, "Completablefuture," 2018, https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Comp

[14] S. overflow, “Most popular technologies,” https://insights.stackoverflow.com/survey/2017most-popular-technologies.

[15] M. Richards, “Software architecture pattern.”

[16] E. Ries, “El método lean startup,” 2011.

[17] Scrum.org, “What is a scrum development team?” 2018, https://www.scrum.org/resources/what-is-a-scrum-development-team.

[18] Stackify, “Soap vs. rest: The differences and benefits between the two widely-used web service communication protocols,” 14/03/2017, https://stackify.com/soap-vs-rest/.

[19] C. Wodehose, “Soap vs. rest: A look at two different api styles,” https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/.

*Apéndice A*

# Contenido del CD

- Memoria en versión PDF.
- Diagrama Astah Diagrama de modelo inicial
- Diagrama de Arquitectura
- Diagrama de Despliegue

*Apéndice B*

# Guía de Despliegue

## B.1. Pre requisitos

Tener instalado Docker CE 17.06

## B.2. Instalación

Ejecutar el *script* de despliegue alojado en la carpeta delivery.

Dentro del repositorio, dicha carpeta se puede consultar bajo la ruta /backend/repository dentro de la rama master