UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una red social para ajedrecistas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Cuartero Baena, Ramsés

Tutor/a: Llorens Agost, María Luisa

CURSO ACADÉMICO: 2022/2023

Resumen

En este Trabajo Fin de Grado (TFG) se busca unir el mundo de la tecnología y las redes sociales con el mundo del ajedrez tradicional y presencial, ayudando a que los jóvenes (y no tan jóvenes) jugadores amateurs, que aprendieron a jugar por su cuenta de forma online, tengan la oportunidad de disfrutar del ajedrez presencial sin necesidad de apuntarse a un club de forma oficial. Para ello, se ha desarrollado una red social para ajedrecistas llamada Rootlink.

Dicha red social tiene las siguientes funcionalidades principales: publicación de mensajes y noticias a través de la cuenta de cada usuario registrado en la red social, chat privado con los usuarios amigos, creación y organización de eventos presenciales, y consulta en un calendario de los eventos a los que el usuario se ha inscrito, así como de la información de los mismos (descripción, fecha, lugar en un mapa, etc.).

El desarrollo de Rootlink se ha llevado a cabo como una aplicación web responsive, que es accesible tanto desde ordenador como desde dispositivos móviles. Para la parte frontend se ha utilizado TypeScript, Angular y CSS, ya que facilitan el desarrollo responsive. La parte backend se ha realizado con Java y su framework SpringBoot. Dicho backend tiene una arquitectura hexagonal, con el patrón "ports and adapters". Se comunica con el frontend mediante una API REST y con la base de datos relacional MySQL mediante la API de persistencia JPA.

Palabras clave: ajedrez, portal web, red social, torneos, eventos.

Abstract

In this Bachelor's thesis (TFG), the aim is to bridge the worlds of technology and social media with the traditional and in-person chess world, helping young (and not so young) amateur players who learned to play online on their own to enjoy in-person chess without the need to officially join a club. To achieve this, a social network for chess players called Rootlink has been developed.

This social network has the following main features: posting messages and news through the user's account registered on the social network, private chat with friends, creation and organization of in-person events, and access to a calendar displaying the events the user signs up for, along with their information (description, date, location on a map, etc.).

The development of Rootlink is carried out as a responsive web application, accessible both from computers and mobile devices. TypeScript, Angular, and CSS are used for the frontend, as they facilitate responsive development. The backend is implemented using Java and its SpringBoot framework. The backend follows a hexagonal architecture with the "ports and adapters" pattern. It communicates with the frontend through a REST API and with the MySQL relational database using the JPA persistence API.

Keywords: chess, web portal, social network, tournaments, events.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

La tecnología ha avanzado a lo largo de los años, mientras que ciertos aspectos, como el ajedrez, se han mantenido prácticamente inalterados durante siglos. Sin embargo, esto no implica que las nuevas tecnologías no puedan beneficiar al ajedrez. En este proyecto, se explora cómo una red social puede impulsar este deporte, atrayendo a más seguidores y mejorando la experiencia para aquellos que ya están cautivados por él.

1.1. Motivación

Este proyecto ha sido elegido debido a una serie de razones que considero muy importantes. El ajedrez es una de mis mayores pasiones y el hecho de poder desarrollar una red social para personas igualmente apasionadas por este juego siempre representa una motivación adicional para mí.

No solo quiero crear un espacio en línea donde los entusiastas del ajedrez puedan conectarse y compartir su amor por el juego, sino que también deseo brindar ayuda y apoyo a aquellos que aún no han tenido la oportunidad de experimentar la emoción de jugar al ajedrez presencialmente con otras personas. Mi objetivo es enseñarles y permitirles disfrutar de este maravilloso deporte y, si es posible, ¡incluso engancharlos a él!

Además, el hecho de poder aprovechar los conocimientos que he adquirido tanto en mis prácticas como en mi trabajo actual es una fuente adicional de motivación. Me motiva la idea de aplicar estas habilidades y conocimientos de manera personal y explorar la tecnología que utilizo a diario sin las limitaciones que mi empresa impone.

Un motivo más para embarcarme en este proyecto es el desafío que representa el desarrollo del frontend de la red social, que hasta ahora ha sido mi punto débil. Considero que esta área es un campo en el que puedo aprender y mejorar significativamente, y trabajar en un proyecto tan ambicioso me brinda la oportunidad de superar obstáculos y ampliar mis habilidades en este ámbito.

En resumen, esta elección se basa en mi pasión por el ajedrez, mi deseo de ayudar a otros a descubrir y disfrutar del juego, así como en la oportunidad de utilizar mis habilidades y conocimientos de manera personal y explorar áreas en las que deseo crecer. Además, el desafío que representa el desarrollo del frontend me motiva a aprender y mejorar en este campo. En conjunto, estas razones hacen que este proyecto sea especialmente atractivo y significativo para mí.

1.2. Objetivos

El objetivo fundamental de este proyecto consiste en desarrollar una red social, llamada Rootlink, específicamente concebida para satisfacer las necesidades de la comunidad ajedrecista, en particular, para dar la oportunidad de disfrutar del ajedrez presencial sin necesidad de apuntarse a un club de forma oficial. Se trata de una plataforma altamente adaptable, capaz de ofrecer una experiencia óptima tanto en entornos de escritorio como en dispositivos móviles.

Los objetivos secundarios para conseguir el objetivo principal son los que siguen:

- Permitir que los usuarios publiquen mensajes de texto, o links de noticias que ellos quieran.
- Permitir que los usuarios creen eventos con carácter presencial, indicando dirección, fecha y hora en el formulario de creación de eventos.
- Permitir a los usuarios apuntarse a dichos eventos.
- Integrar un calendario independiente para cada usuario donde poder consultar las fechas de los eventos a los cuales se han apuntado.
- Integrar un mapa en la pantalla de información del evento para poder tener, de una manera más gráfica e interactiva, la dirección del evento.
- Permitir a los usuarios cambiar su perfil en cualquier momento, pudiendo introducir una foto de perfil, su elo[1] FIDE (Federación Internacional de Ajedrez), etc.
- Permitir a los usuarios tener un chat privado con el resto de usuarios para poder enlazar amistades de una forma más íntima.

1.3. Estructura de la memoria

En este apartado se presenta un resumen de cada uno de los capítulos que componen este documento.

El capítulo 2 trata del estado del arte, con un análisis de la competencia más destacada para Rootlink en el mercado y un análisis DAFO.

En el capítulo 3 se realiza un análisis del problema, mediante los respectivos diagramas que ayudan a entenderlo de una manera más gráfica.

El capítulo 4 se centra, después de haber hecho el respectivo análisis del problema en el punto anterior, en cómo afrontarlo. Aquí es donde se explica el porqué de la elección de la arquitectura, tecnología utilizada y el diseño.

[1] https://es.wikipedia.org/wiki/Sistema_de_puntuaci%C3%B3n_Elo

En el capítulo 5 se detalla el proceso de desarrollo, las decisiones tomadas al presentarse problemas y la solución a la que se ha llegado.

En el capítulo 6 se describen las pruebas unitarias realizadas a las funcionalidades de la aplicación, así como también las pruebas realizadas directamente a la aplicación ya montada y funcionando.

Las conclusiones del TFG se exponen en el capítulo 7, junto con la relación del trabajo desarrollado con los estudios cursados.

En el capítulo 8 se habla de los trabajos a realizar para las próximas versiones de la aplicación.

Por último, se encuentra la bibliografía y un anexo con los Objetivos de Desarrollo Sostenible (ODS) relacionados con este proyecto.

2. Estado del arte

Actualmente hay bastantes aplicaciones en internet orientadas al ajedrez. Hay portales web para jugar en línea, para formarse, foros, etc. También hay algunos portales que ofrecen una red social para ajedrecistas. Pero, a diferencia del que se desarrolla en este TFG, la principal función y finalidad de estos portales es otra. En este capítulo, se presentan algunas de estas aplicaciones y se realiza el análisis DAFO (1) para Rootlink.

2.1. Aplicaciones similares

En el mercado hay varias plataformas que podrían ser competencia con Rootlink. Por norma general, estas se centran más en poder jugar de forma instantánea al ajedrez con otros oponentes a través de internet y en el aprendizaje. En cambio, Rootlink está pensado como red social para poder contactar con otros ajedrecistas que quieran jugar de forma presencial o, simplemente, participar en la red social para difundir o leer noticias de este deporte.

En las plataformas que se describen a continuación la parte de red social es bastante pobre, dado que no es su principal función. Y, además, no se ha encontrado ninguna aplicación que esté orientada a fomentar el ajedrez presencial. Ahí es donde ocupa su espacio Rootlink.

2.1.1. Chess24

Chess24[2] es una plataforma de ajedrez que ofrece partidas en línea, lecciones, análisis y la posibilidad de seguir y ver torneos en vivo. También cuenta con una comunidad activa de jugadores y ofrece contenido educativo. Aunque no tiene una funcionalidad específica para la organización de eventos presenciales, Chess24 se ha asociado con torneos y eventos de ajedrez presenciales en el pasado. En la Figura 1 se muestra la página principal de Chess24.

[2] https://chess24.com/es

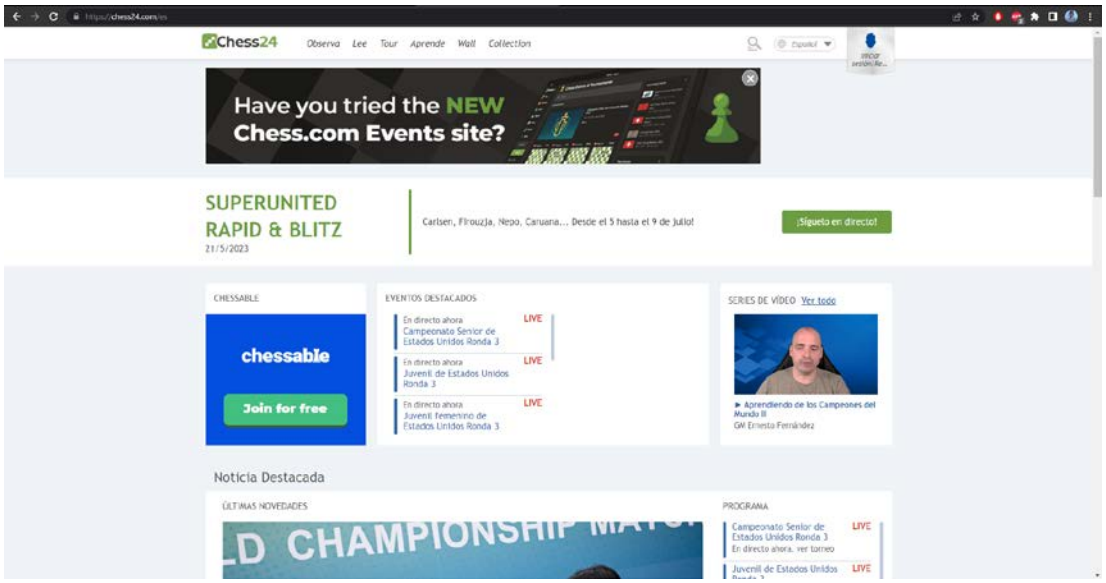

Figura 1. Chess24

2.1.2. Chess.com

Chess.com[3] es una de las plataformas de ajedrez en línea más populares y establecidas. Ofrece una comunidad global de jugadores de ajedrez, permitiendo jugar partidas en línea, participar en torneos, acceder a lecciones y análisis, y unirte a clubes y grupos temáticos. También cuenta con una opción para organizar eventos presenciales y encontrar jugadores cercanos. Chess.com ha construido una sólida reputación y tiene una amplia base de usuarios activos. La web de Chess.com se muestra en la Figura 2.

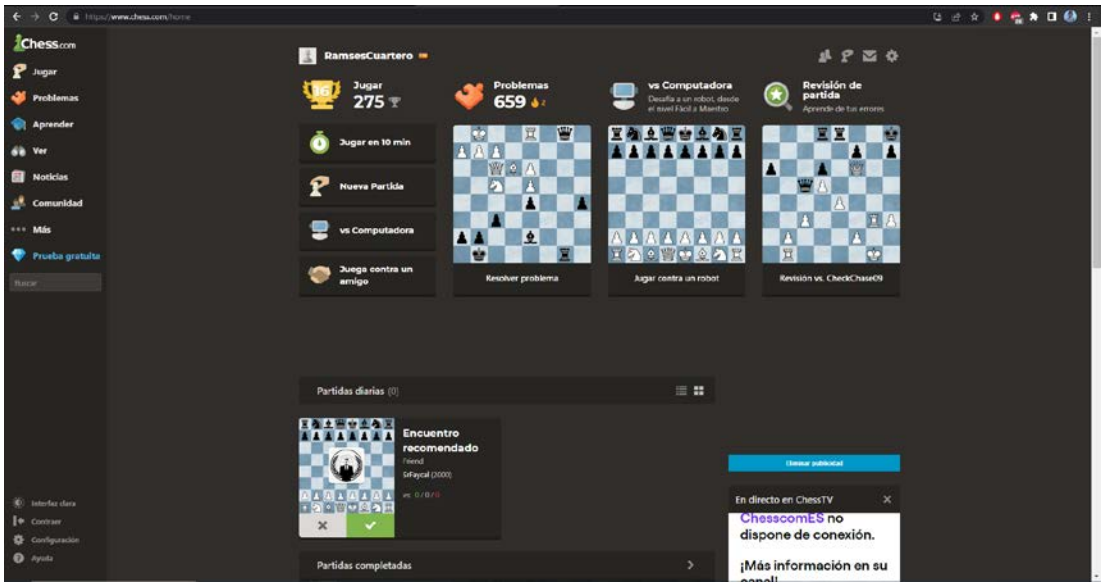

Figura 2. Chess.com

[3] https://www.chess.com/es

2.1.3. Lichess.org

Lichess.org[4] es otra plataforma de ajedrez en línea gratuita y de código abierto. Ofrece funciones similares a Chess.com, incluyendo partidas en línea, torneos, análisis de partidas y acceso a lecciones. Aunque Lichess.org no tiene una funcionalidad específica para la organización de eventos presenciales, es una opción popular para jugar al ajedrez en línea y cuenta con una comunidad activa de jugadores. En la Figura 3 se puede ver una captura de la página principal de Lichess.org.

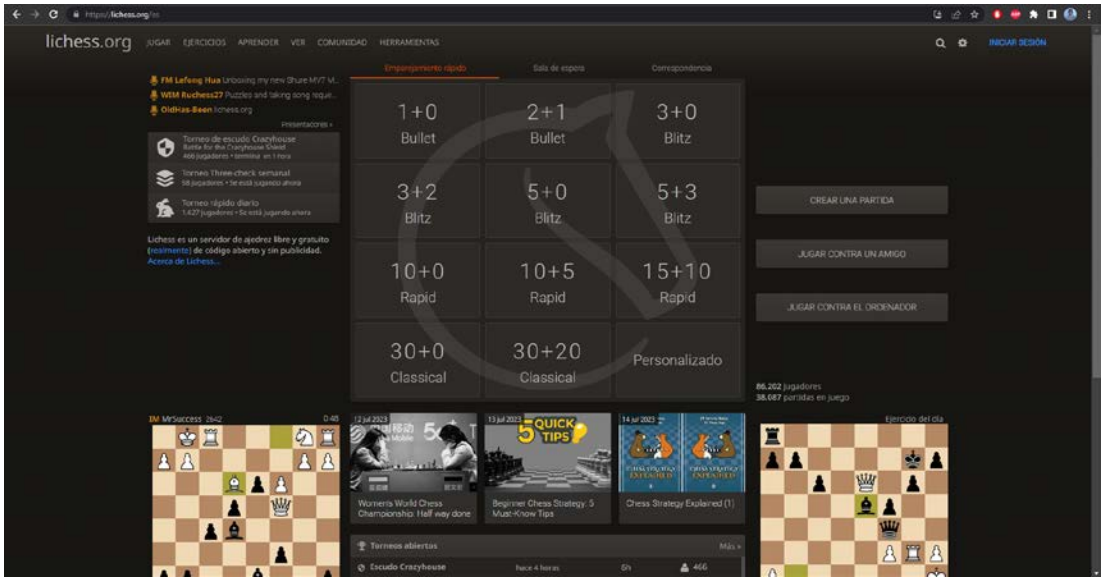

Figura 3. Lichess.org

2.2. Análisis DAFO

El análisis DAFO es una herramienta utilizada para evaluar una situación en diferentes contextos, como negocios, proyectos o incluso en la toma de decisiones personales. El término DAFO proviene de las iniciales de las cuatro áreas clave que se analizan:

1. Debilidades: Aspectos internos que pueden afectar negativamente al negocio.
2. Amenazas: Factores externos que pueden ser un obstáculo.
3. Fortalezas: Aspectos internos positivos y ventajas.
4. Oportunidades: Factores externos que afectan favorablemente al negocio.

El propósito del análisis DAFO es permitir la toma de decisiones más informadas y estratégicas. A partir de la evaluación de estos factores, se pueden desarrollar planes y

[4] https://lichess.org/

estrategias para potenciar las fortalezas, abordar las debilidades, aprovechar las oportunidades y mitigar las amenazas. Esto ayuda a mejorar la planificación y aumentar las posibilidades de alcanzar los objetivos deseados con éxito.

En lo que sigue, se detalla el análisis DAFO para Rootlink, mostrándose en la Tabla 1 un resumen de los cuatro elementos del mismo.

DEBILIDADES	AMENAZAS
• Falta de reconocimiento • Competencia fuerte • Complejidad del desarrollo	• Privacidad y seguridad • Desafíos técnicos • Cambios en las preferencias de los usuarios
FORTALEZAS	OPORTUNIDADES
• Nicho específico • Funcionalidades exclusiva • Interacción social	• Crecimiento de la comunidad de ajedrez • Alianzas estratégicas • Monetización potencial

Tabla 1. Análisis DAFO para Rootlink

DEBILIDADES:

- Falta de reconocimiento: Si el portal web es nuevo, es posible que carezca de reconocimiento y usuarios iniciales. Esto puede dificultar el crecimiento y la atracción de una base de usuarios activa.
- Competencia fuerte: El mercado de las redes sociales y las comunidades en línea es altamente competitivo. Existen otras plataformas sociales y aplicaciones relacionadas con el ajedrez que podrían tener una base de usuarios establecida y leal.
- Complejidad del desarrollo: La implementación de funciones específicas como el calendario, la creación de eventos presenciales y la integración de un mapa puede requerir habilidades técnicas avanzadas y un desarrollo complejo. Esto puede suponer un desafío en términos de tiempo y recursos.

FORTALEZAS:

- Nicho específico: El portal web se centra en un nicho particular, el ajedrez. Esto permite la creación de una comunidad especializada y apasionada de jugadores de ajedrez, lo que podría generar un mayor compromiso y participación.
- Funcionalidades exclusivas: La inclusión de funciones como la creación de eventos presenciales, apuntarse a eventos, un calendario personalizado y un mapa para encontrar los lugares de los eventos, proporciona un valor único a los usuarios. Estas

características pueden diferenciar al portal de otras redes sociales y atraer a los entusiastas del ajedrez que buscan jugar en persona y participar en torneos.

- Interacción social: Al ser una red social, el portal web ofrece la oportunidad de interactuar con otros ajedrecistas, establecer conexiones, compartir experiencias y discutir estrategias. Esto puede fomentar un sentido de comunidad y colaboración entre los jugadores.

OPORTUNIDADES:

- Crecimiento de la comunidad de ajedrez: El ajedrez ha ganado popularidad en los últimos años, especialmente debido a la serie de Netflix "The Queen's Gambit" y otras transmisiones de torneos de alto perfil. Esto brinda una oportunidad para atraer a nuevos jugadores y expandir la comunidad de ajedrecistas en línea.

- Alianzas estratégicas: El portal web puede buscar asociaciones con organizaciones de ajedrez locales, clubes, federaciones o incluso jugadores profesionales. Estas asociaciones pueden ayudar a promover el portal, atraer usuarios y generar credibilidad en la comunidad del ajedrez.

- Monetización potencial: Si el portal web puede establecer una base de usuarios activa y comprometida, existen oportunidades para la monetización a través de anuncios, membresías premium, patrocinios de eventos y promoción de productos relacionados con el ajedrez.

AMENAZAS:

- Privacidad y seguridad: Las redes sociales están sujetas a amenazas relacionadas con la privacidad y la seguridad de los datos. Es fundamental implementar medidas sólidas de seguridad y protección de la información personal de los usuarios para evitar cualquier brecha o acceso no autorizado.

- Desafíos técnicos: La gestión de una plataforma web con múltiples funciones, como el calendario y el mapa, puede ser compleja. Los errores técnicos, las interrupciones del servicio y la falta de escalabilidad pueden afectar negativamente la experiencia del usuario y la retención de usuarios.

- Cambios en las preferencias de los usuarios: Las preferencias y tendencias en línea pueden cambiar rápidamente. Es posible que los usuarios cambien a otras plataformas o aplicaciones si encuentran características más atractivas o si se desarrollan nuevas tendencias en la comunidad del ajedrez en línea.

3. Análisis del problema

3.1. Identificación y análisis de soluciones posibles

En esta sección, se exponen las metodologías más conocidas para el desarrollo de un proyecto software y las diferentes maneras de llevar a cabo dicho desarrollo.

3.1.1. Metodologías de desarrollo de software

Scrum:

Scrum (2) es una metodología ágil de desarrollo de software que se centra en la entrega incremental y la colaboración constante. En lugar de seguir un plan rígido desde el principio, Scrum divide el proyecto en ciclos de trabajo llamados "sprints", que son períodos cortos y fijos de tiempo, generalmente de 1 a 4 semanas.

En Scrum, se utiliza un "backlog" para listar y priorizar las tareas pendientes. Antes de cada sprint, el equipo selecciona un conjunto de tareas del backlog que se compromete a completar durante ese sprint. Durante el sprint, el equipo trabaja en estas tareas y se reúne diariamente en lo que se llama un "Daily Scrum" para compartir el progreso y discutir cualquier obstáculo encontrado.

Al final de cada sprint, el equipo revisa lo que ha desarrollado en una "Sprint Review" con los stakeholders y obtiene retroalimentación. Esto permite ajustar y adaptar el producto según las necesidades cambiantes. Finalmente, en la "Sprint Retrospective", el equipo reflexiona sobre su proceso de trabajo y busca formas de mejorar en el próximo sprint.

Los roles clave en Scrum son:

- Scrum Master: Es el facilitador del equipo. Su función principal es eliminar obstáculos que puedan estar afectando al equipo y asegurarse de que se sigan las prácticas de Scrum.
- Product Owner: Representa los intereses de los stakeholders y define las características y funcionalidades prioritarias para el producto. Es responsable de mantener el backlog actualizado y de asegurarse de que el equipo desarrolle lo más valioso primero.
- Equipo de Desarrollo: Son los profesionales encargados de llevar a cabo el trabajo real de desarrollo durante el sprint. Son autónomos y autoorganizados, y colaboran estrechamente para entregar las tareas comprometidas.

Scrum promueve la flexibilidad al permitir ajustes en el producto y el proceso en función de la retroalimentación constante. Al trabajar en sprints cortos y enfocarse en

entregas incrementales, Scrum ayuda a los equipos a adaptarse más rápidamente a los cambios y a mantener un flujo constante de entregas de valor al cliente.

XP:

La Metodología XP (Extreme Programming) (3) es una metodología ágil para el desarrollo de software que se enfoca en la excelencia técnica, la comunicación efectiva y la retroalimentación constante. Fue creada por Kent Beck a finales de la década de 1990 y ha sido ampliamente adoptada en la comunidad de desarrollo de software.

Las principales prácticas de XP incluyen:

- Planificación y Entrega Incremental: En XP, el desarrollo se realiza en pequeñas iteraciones, también conocidas como "pequeñas entregas" o "releases". Cada iteración representa una versión funcional del producto que se puede entregar al cliente.
- Programación en Pareja (Pair Programming): Los desarrolladores trabajan en parejas, uno escribiendo el código y el otro revisando cada línea de código a medida que se escribe. Esta práctica mejora la calidad del código y el conocimiento compartido en el equipo.
- Pruebas Unitarias (Unit Testing): Se escriben pruebas automatizadas para cada parte del código. Estas pruebas permiten a los desarrolladores detectar errores tempranamente y garantizar que el código funcione correctamente.
- Refactorización Continua: Se mejora constantemente el diseño del código sin cambiar su comportamiento. Esto ayuda a conservar el código limpio y facilita su mantenimiento.
- Integración Continua (Continuous Integration): Los cambios en el código se integran y se prueban automáticamente varias veces al día para evitar problemas de integración tardíos.
- Diseño Simple: Se enfatiza la simplicidad en el diseño del código. Esto evita la creación de soluciones complicadas y facilita el mantenimiento futuro.
- Cliente en el Sitio (On-Site Customer): Un representante del cliente trabaja directamente con el equipo de desarrollo para proporcionar una retroalimentación continua y clarificar los requisitos.
- Metáfora (Metaphor): Se utiliza una metáfora para darle sentido al sistema en desarrollo y mantener una visión compartida del producto entre el equipo y el cliente.
- Juego de la Planificación (Planning Game): Se realiza una planificación iterativa para determinar qué funcionalidades se deben desarrollar en cada iteración y cómo se llevará a cabo el trabajo.
- Estándares de Codificación (Coding Standards): Se establecen pautas para el estilo y la calidad del código, lo que ayuda a mantener la consistencia en el desarrollo.

RUP:

La Metodología RUP (4) es un enfoque para desarrollar software que se basa en dividir el proyecto en diferentes etapas y trabajar de forma iterativa en cada una de ellas. Tiene cuatro etapas principales: Inicio, Elaboración, Construcción y Transición.

En la etapa de Inicio, se definen los objetivos y requisitos del proyecto. En la etapa de Elaboración, se realiza un análisis más detallado y se diseña la arquitectura del software. En la etapa de Construcción, se lleva a cabo el desarrollo del software. Y, finalmente, en la etapa de Transición, se realiza la entrega del producto final y se brinda soporte para su uso.

Cada etapa se divide en pequeñas iteraciones, lo que significa que el desarrollo avanza en pasos cortos y se pueden realizar ajustes en función de los resultados obtenidos. RUP también presta atención a la comunicación efectiva entre el equipo y los interesados, y se enfoca en satisfacer las necesidades del cliente.

En resumen, RUP es un enfoque estructurado y flexible para desarrollar software, que se enfoca en dividir el proyecto en etapas y trabajar de forma iterativa para lograr un producto final satisfactorio.

3.1.2. Proceso del desarrollo del software

TDD:

El Desarrollo Dirigido por Pruebas o Test-Driven Development (TDD) (5) es una metodología de desarrollo de software en la cual se escriben primero pruebas automatizadas para una funcionalidad deseada o cambio en el código (ver Figura 4). Estas pruebas deben fallar inicialmente ya que el código de producción aún no se ha escrito.

Posteriormente, se escribe el código mínimo necesario para que las pruebas pasen con éxito. Una vez hecho esto, se ejecutan todas las pruebas, incluyendo la nueva que fue agregada, para verificar que todo funcione conforme a lo esperado.

Finalmente, si todas las pruebas son correctas, se puede mejorar y optimizar el código sin modificar su comportamiento. Este ciclo se repite para cada nueva funcionalidad o mejora requerida.

El TDD mejora la calidad del código, facilita la detección temprana de errores y proporciona documentación en forma de pruebas automatizadas. Además, promueve un diseño más simple y modular del código. En resumen, el TDD es una práctica que asegura el correcto funcionamiento del código y facilita su mantenimiento a lo largo del tiempo.

TLD:

El Test-Last Development (TLD) (5) es todo lo contrario al TDD. Primero se escribe el código y luego las pruebas (ver Figura 4). Esto sirve para corregir el código en caso de fallo de las pruebas hasta que estas pasen correctamente.

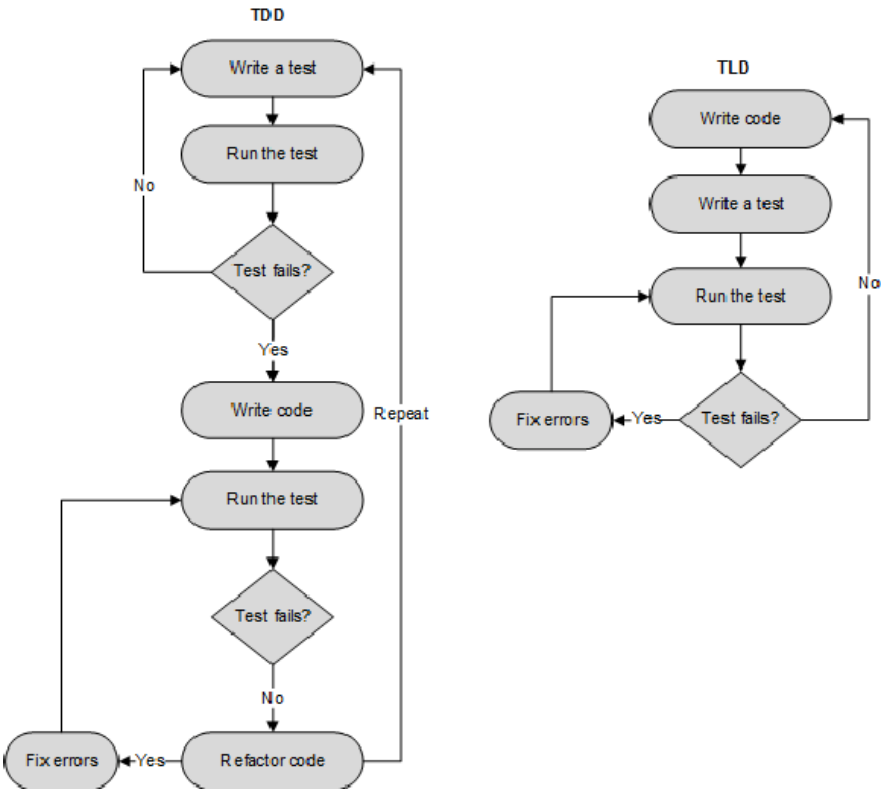

Figura 4. TDD versus TLD (5)

3.2. Solución propuesta

Revisadas las opciones presentadas en la sección anterior, se elige la opción de Scrum. Aunque no se puede llevar a cabo la metodología al completo debido a que el equipo está compuesto por una única persona, sí se llevará a cabo la parte de desarrollo por sprints, mantenimiento de un backlog, etc. En caso de que el equipo se ampliara, sigue estando la posibilidad de aplicar roles.

Respecto al proceso de desarrollo, se ha elegido TLD. Es una buena forma de conseguir un código de calidad y de mejorar la productividad. Además, se mejora la eficiencia a la hora de encontrar errores y saber si se han introducido errores.

3.3. Modelo de dominio

El propósito principal de la red social es fomentar el ajedrez presencial con ayuda de las nuevas tecnologías, permitiendo la creación de mensajes (posts) y de eventos con lugar y fecha para poder apuntarse y asistir. La gestión de dichos eventos tiene mucho potencial para crear una gran aplicación, pero, debido al límite temporal y al número de desarrolladores, no es posible explotarlo y solo se ha desarrollado una primera versión de este portal web. En un futuro, se espera poder ampliar la misma para, esta vez sí, sacar todo el partido que puede tener esta red social en el mercado.

En el modelo de dominio que se muestra en la Figura 5, se distinguen 8 clases. La clase más importante es la clase "usuario" que, como se puede observar, no tiene ninguna categoría especial, dado que, en la aplicación, todos los usuarios pueden acceder a toda la funcionalidad de la misma.

La red social permite la creación de mensajes en el tablón llamados posts, representados en el modelo de dominio por las clases "tablón" y "post". Estos mensajes pueden ser meramente informativos, curiosidades que el usuario quiera compartir, etc.

Como ya se ha comentado, los mensajes también pueden ser tipo evento, donde en el mismo mensaje está indicada la información del evento, el lugar, la fecha y algún dato de interés más que el usuario publicador crea conveniente. Se representan en el modelo por otra clase bastante importante, la clase "evento". Esto pone de manifiesto la funcionalidad principal hacia la que Rootlink está orientada: ayudar a los ajedrecistas a poder disfrutar del ajedrez presencial en eventos creados a partir de esta web.

En la ventana del evento hay un mapa que marca de forma más gráfica e interactiva el lugar del evento, representado por la clase "mapa" en el modelo. A estos eventos, los demás usuarios se pueden apuntar y, automáticamente, se les agenda en un calendario propio para cada usuario registrado en la red social, representado por la clase "calendario".

Además, se crea un sistema de chat privado entre usuarios, lo que puede favorecer la creación de amistades con otros usuarios para poder quedar de forma más privada para jugar al ajedrez. Está representado por las clases "sistema de mensajería" y "mensaje" en el modelo de dominio.

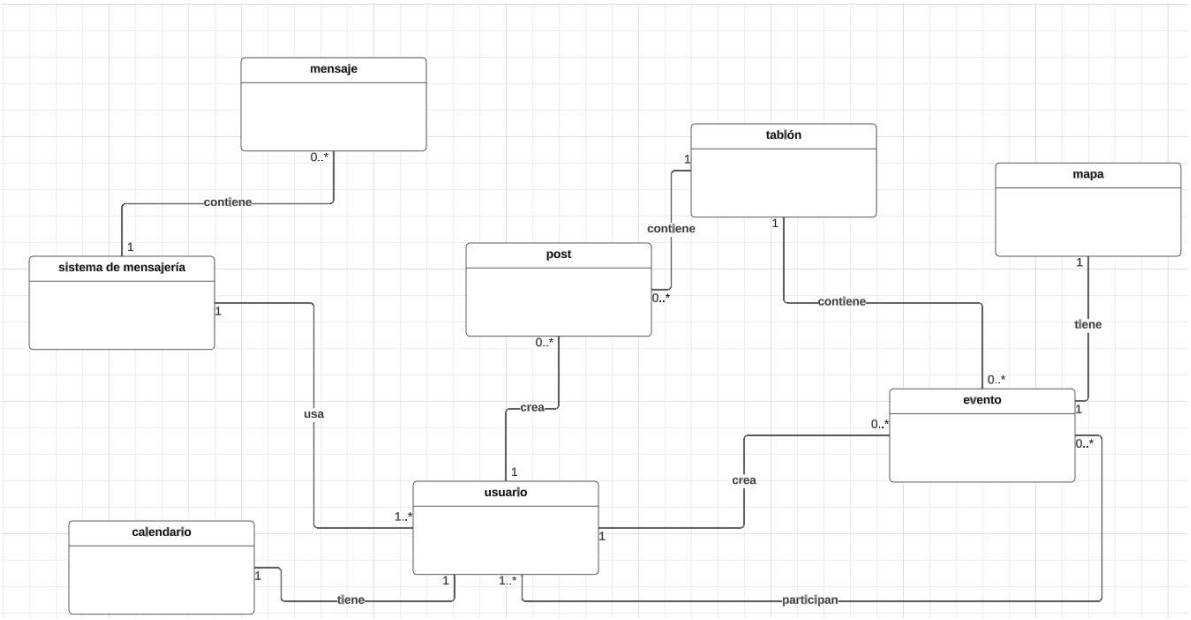

Figura 5. Modelo de dominio

3.4. Modelo de casos de uso

En esta sección, se muestra de manera gráfica la funcionalidad de la aplicación mediante el modelo de casos de uso de UML. Se divide en características de la aplicación para hacerlo más entendible y poder ir a una funcionalidad en concreto en lugar de tener que encontrarla en un modelo enorme.

En la Figura 6 se muestra el diagrama de contexto de la aplicación definiendo los límites del sistema y los respectivos actores que interactúan con la misma.

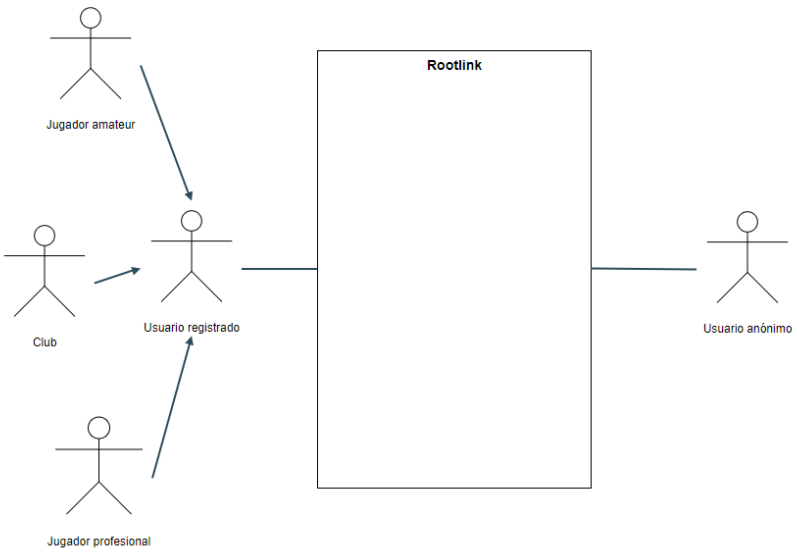

Figura 6. Diagrama de contexto

Se distinguen dos tipos de actores: los usuarios registrados y los usuarios anónimos (en las Tabla 2 y Tabla 3, respectivamente). La red social puede usarse por cualquier usuario registrado. Solo los usuarios registrados pueden iniciar sesión y escribir mensajes en el chat o crear eventos. Un usuario anónimo no puede usar el chat privado que ofrece la aplicación; solo puede registrarse en la red social para poder iniciar sesión y así poder empezar a usarla.

Actor	Usuario registrado	A1
Descripción	Usuarios que ya tienen una cuenta creada en la aplicación.	
Características	Tienen permisos para acceder a todas las funcionalidades de la aplicación.	
Referencias		

Tabla 2. Actor usuario registrado

Actor	Usuario anónimo	A2
Descripción	Usuarios que no han creado una cuenta en la aplicación.	
Características	Solo tienen permisos para crear una nueva cuenta.	
Referencias		

Tabla 3. Actor usuario anónimo

Para que un usuario pueda iniciar sesión en la aplicación, primero debe estar registrado en la misma. En la Figura 7 se muestra el diagrama de casos de uso de un usuario anónimo y su descripción en la Tabla 4.

Figura 7. Usuario anónimo

UC-1	**Registrar usuario**	
Descripción	El usuario anónimo accede por primera vez a la red social	
Actores	Usuario anónimo	
Precondición	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario anónimo accede por primera vez a la web.

	2 3 4	El usuario anónimo accede al apartado de registro de usuario. El usuario anónimo introduce su nombre de usuario y sus datos en la ventana de registro. El sistema comprueba que los datos introducidos son correctos.
Excepciones	Paso	Acción
	3	El usuario ha introducido algún dato de forma incorrecta, o se ha dejado algún campo obligatorio por rellenar. El sistema le informa de su error y vacía todos los campos para que el usuario pueda hacer otro intento para rellenarlos.
Postcondición	Se valida la información y es redirigido a la ventana de inicio de sesión.	

Tabla 4. Caso de uso registrar usuario

Para que el usuario, una vez registrado, pueda acceder a la funcionalidad de la aplicación debe iniciar sesión. En la Figura 8 se muestra el diagrama de casos de uso de un usuario registrado y su descripción en la Tabla 5. Caso de uso iniciar sesión.

Figura 8. Usuario registrado. Iniciar sesión

UC-2	**Iniciar sesión**	
Descripción	El usuario intenta iniciar sesión introduciendo sus credenciales (usuario y contraseña).	
Actores	Usuario registrado	
Precondición	- Que el usuario tenga una cuenta en la aplicación. - Que no tenga una sesión iniciada.	
Secuencia normal	Paso	Acción
	1 2	El usuario introduce su nombre de usuario y su contraseña en la ventana de inicio de sesión. El sistema comprueba las credenciales y le da acceso al sistema.
Excepciones	Paso	Acción
	1	El usuario introduce de manera errónea las credenciales y el sistema la informa del fallo y le

		vuelve a pedir las credenciales vaciando las credenciales introducidas anteriormente.
Postcondición	El usuario accede a la ventana del menú principal.	

Tabla 5. Caso de uso iniciar sesión

Como se ha dicho antes, un usuario registrado tiene toda la funcionalidad de la aplicación a su disposición. En la Figura 9Figura 9 se muestra el diagrama de casos de uso y, desde la Tabla 6 a la Tabla 19, la descripción de los mismos.

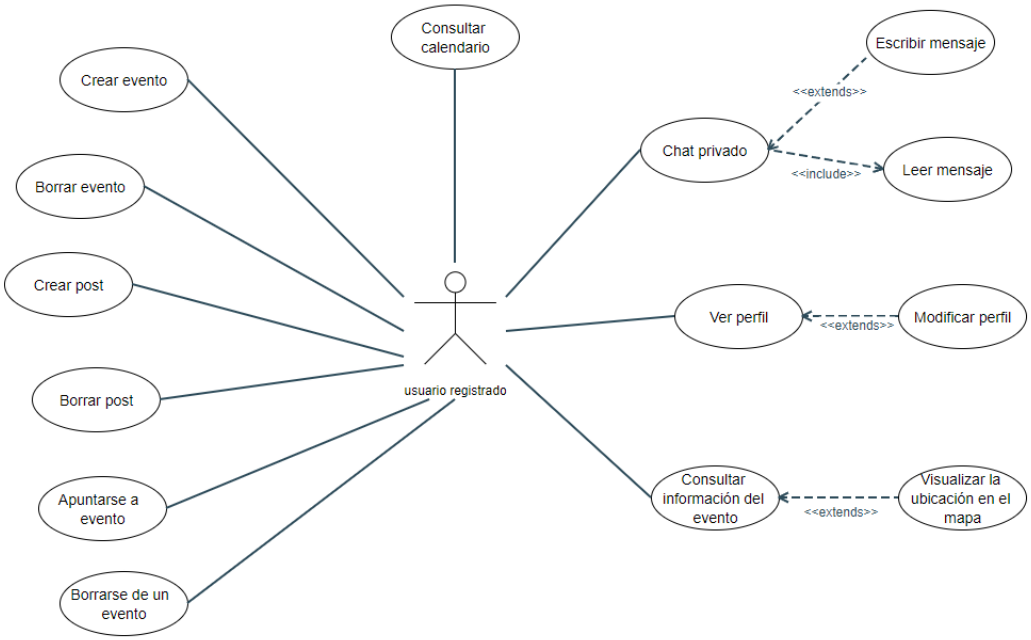

Figura 9. Usuario registrado. Funcionalidad completa

UC-3	**Consultar calendario**	
Descripción	El usuario consulta su calendario con los eventos agendados	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la ventana de calendario.
	2	El sistema rellena el calendario con los eventos agendados del usuario y lo muestra en pantalla.
Excepciones	Paso	Acción
	1	El usuario no tiene ningún evento agendado y el sistema le devuelve un calendario vacío.
Postcondición	Ninguna	

Tabla 6. Caso de uso consultar calendario

UC-4	**Crear evento**	
Descripción	El usuario crea un evento al que otros usuarios puedan apuntarse.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la ventana de creación de evento.
	2	Rellena el formulario con la descripción del evento, fecha, hora, lugar, etc.
	3	El sistema lo publica en el tablón.
Excepciones	Paso	Acción
	1	El usuario no ha rellenado algún campo obligatorio y/o ha rellenado de forma incorrecta algún campo.
Postcondición	Ninguna	

Tabla 7. Caso de uso crear evento

UC-5	**Borrar evento**	
Descripción	El usuario borra un evento publicado	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión y haber publicado al menos un evento.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la pantalla de editar evento.
	2	Pulsa el botón de borrar evento.
	3	El sistema le pregunta si de verdad lo quiere borrar.
	4	El usuario indica que sí quiere borrarlo.
	5	El sistema borra el evento del calendario del usuario y de los calendarios de los usuarios que lo tenían agendado.
Excepciones	Paso	Acción
	4	El usuario indica que no quiere borrarlo y le devuelve a la ventana de editar evento.
Postcondición	El evento se borra del sistema y los calendarios de los usuarios ya no tienen dicho evento agendado.	

Tabla 8. Caso de uso borrar evento

UC-6	**Crear post**	
Descripción	El usuario escribe un mensaje en el tablón de la aplicación.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión.	
Secuencia normal	Paso	Acción

	1 2 3 4	El usuario accede a la ventana de redacción del post. Redacta la publicación en el cuadro de texto para ello. El usuario le da al botón de publicar post. El sistema muestra el mensaje en el tablón de la aplicación.
Excepciones	Paso	Acción
	2 3	El usuario no ha rellenado el cuadro de texto con ningún texto. Entonces el sistema le devuelve un mensaje que indica el error. El usuario le da a cancelar el proceso de publicación de post. El sistema le devuelve al tablón de la aplicación.
Postcondición	El usuario es redirigido al tablón de la aplicación.	

Tabla 9. Caso de uso crear post

UC-7	**Borrar post**	
Descripción	El usuario borra un post que ya ha publicado en el tablón de la aplicación.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión y haber publicado un post.	
Secuencia normal	Paso	Acción
	1 2 3 4 5	El usuario accede al tablón de la aplicación. En la publicación de su post el usuario pulsa el botón de borrar. El sistema le pregunta que si de verdad quiere borrar la publicación. El usuario selecciona que sí quiere borrarlo. El sistema borra el post del tablón.
Excepciones	Paso	Acción
	4	El usuario selecciona que no quiere borrar el post. El sistema le devuelve al tablón.
Postcondición	El usuario es redirigido al tablón de la aplicación.	

Tabla 10. Caso de uso borrar post

UC-8	**Apuntarse a un evento**	
Descripción	El usuario se apunta a un evento creado por un usuario.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión y que algún usuario haya creado al menos un evento.	
Secuencia normal	Paso	Acción
	1	El usuario accede al tablón de la aplicación.

	2	En la publicación de un evento el usuario accede a la ventana de información del evento.
	3	El usuario le da al botón de apuntarse al evento.
	4	El sistema lo suma como participante y le agenda el evento en su calendario.
Excepciones	Paso	Acción
		Ninguna
Postcondición	Ninguna	

Tabla 11. Caso de uso apuntarse a un evento

UC-9	**Borrarse de un evento**	
Descripción	El usuario borra su participación en un evento.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión, que algún usuario haya publicado al menos un evento y haberse apuntado a un evento.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la ventana de su calendario y/o al tablón para seleccionar el evento en cuestión.
	2	El usuario accede a la ventana de información del evento.
	3	El usuario le da al botón de no ir al evento.
	4	El sistema le pregunta si de verdad no quiere ir al evento.
	5	El usuario selecciona que no quiere ir al evento.
	6	El sistema borra la participación del usuario en el evento y le quita de su calendario el evento agendado.
Excepciones	Paso	Acción
	5	El usuario selecciona que sí quiere ir al evento y entonces el sistema le devuelve a la ventana de información del evento.
Postcondición	El usuario es redirigido a la ventana de información del evento.	

Tabla 12. Caso de uso borrarse de un evento

UC-10	**Ver perfil**	
Descripción	El usuario visita un perfil de un usuario o el suyo propio.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión.	
Secuencia normal	Paso	Acción
	1	El usuario busca un nombre de usuario o pulsa en el botón mi perfil del menú principal de la aplicación.

	2	El sistema carga la ventana de perfil del usuario.
Excepciones	Paso	Acción
	4	El usuario busca un usuario que no existe. El sistema le informa que no hay usuarios con ese nombre.
Postcondición	El usuario es redirigido a la ventana de perfil.	

Tabla 13. Caso de uso ver perfil

UC-11	**Modificar perfil**	
Descripción	El usuario modifica su propio perfil.	
Actores	Usuario registrado	
Precondición	Haber iniciado sesión y haber entrado en la ventana de tu propio perfil.	
Secuencia normal	Paso	Acción
	1 2 3 4	El usuario pulsa el botón de editar perfil El sistema le ofrece un formulario a rellenar con los nuevos posibles datos. El usuario los rellena. El sistema los guarda en la base de datos.
Excepciones	Paso	Acción
	3	El usuario no rellena ningún campo o lo rellena de forma errónea. El sistema lanza un mensaje informando del error.
Postcondición	El usuario es redirigido a la ventana de perfil.	

Tabla 14. Caso de uso modificar perfil

UC-12	**Acceder al chat privado**	
Descripción	El usuario accede al apartado de chat privado.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión.	
Secuencia normal	Paso	Acción
	1 2	El usuario pulsa el botón de chats. El sistema carga la ventana de chats privados.
Excepciones	Paso	Acción
		Ninguna
Postcondición	El usuario es redirigido a la ventana de chats.	

Tabla 15. Caso de uso acceder al chat privado

UC-13	**Leer mensaje**	
Descripción	El usuario entra en un chat concreto para leer los mensajes.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión y haber accedido a la ventana de chats.	
Secuencia normal	Paso	Acción

	1	El usuario selecciona un chat privado con un usuario en concreto.
	2	El sistema carga el chat con los nuevos mensajes.
Excepciones	Paso	Acción
	2	No hay ningún mensaje en el chat. El sistema le informa de que no hay mensajes.
Postcondición	Ninguna	

Tabla 16. Caso de uso leer mensaje

UC-14	**Escribir mensaje**	
Descripción	El usuario escribe un mensaje y lo manda al usuario con el que mantiene el chat privado.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión y haber accedido a la ventana de chats.	
Secuencia normal	Paso	Acción
	1	El usuario escribe un mensaje en la caja de texto y le da al botón de enviar.
	2	El sistema envía el mensaje al usuario y se refleja en el chat.
Excepciones	Paso	Acción
		Ninguna
Postcondición	Ninguna	

Tabla 17. Caso de uso escribir mensaje

UC-15	**Consultar información de un evento**	
Descripción	El usuario accede a la ventana de información del evento.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión y que algún usuario haya creado al menos un evento.	
Secuencia normal	Paso	Acción
	1	El usuario le da al botón de información del evento desde la publicación del tablón o desde el calendario del usuario.
	2	El sistema carga la ventana con toda la información del evento.
Excepciones	Paso	Acción
		Ninguna
Postcondición	Ninguna	

Tabla 18. Caso de uso consultar información de un evento

UC-16	**Visualizar ubicación en el mapa**	
Descripción	El usuario visualiza en el mapa la ubicación del evento.	
Actores	Usuario registrado.	
Precondición	Haber iniciado sesión y que algún usuario haya creado al menos un evento.	
Secuencia normal	Paso	Acción

	1 2 3	El usuario le da al botón de información del evento desde la publicación del tablón o desde el calendario del usuario. El sistema carga la ventana con toda la información del evento en el mapa. El usuario interactúa con el mapa en el que hay un punto marcado.
Excepciones	Paso	Acción
	2	No hay ningún punto en el mapa y el mapa está completamente vacío. El usuario puede interactuar con el mapa, pero no hay punto marcado.
Postcondición	Ninguna	

Tabla 19. Caso de uso visualizar ubicación en el mapa

3.5. Prototipos

Un prototipo de las ventanas de una aplicación es una representación visual y estática de cómo se verán las diferentes pantallas de la app antes de su desarrollo completo. Estos diseños preliminares permiten a los diseñadores y stakeholders evaluar la estructura, el diseño y las interacciones básicas de la interfaz de usuario. Además, los prototipos ayudan a visualizar el flujo de la aplicación, la disposición de elementos y la experiencia del usuario. Estos prototipos son útiles para obtener una vista previa de la apariencia y funcionalidad de la app, ahorrando tiempo y recursos al ajustar los diseños antes de pasar a la fase de desarrollo. Una vez aprobado, el prototipo sirve como guía para los desarrolladores, asegurando que la aplicación final se alinee con la visión y especificaciones iniciales del diseño.

En las figuras que siguen se pueden ver los prototipos de las distintas ventanas de la aplicación.

El tablón y menú principal de la aplicación se muestra en la Figura 10. Desde el tablón, para borrar un post, el usuario pulsará encima de un post propio y entonces tendrá la opción de editarlo o borrarlo.

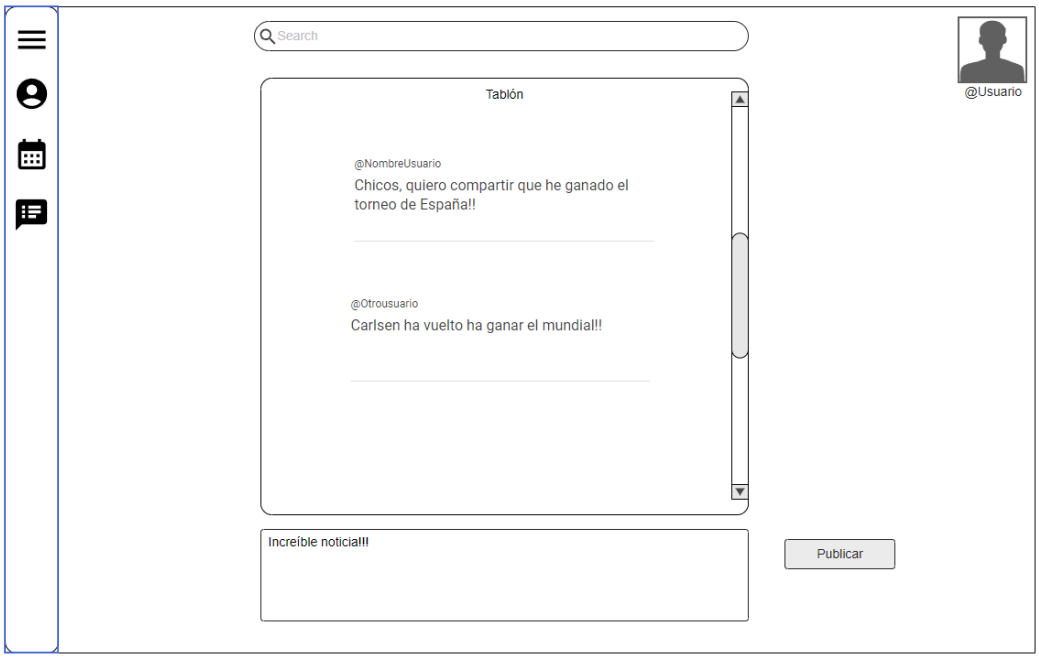

Figura 10. Tablón y menú principal de la aplicación

En la Figura 11 se muestra la ventana de perfil. En el caso de que la ventana de perfil sea la del usuario con sesión iniciada, habrá un botón que de paso a la ventana de editar perfil, que se muestra en la Figura 12.

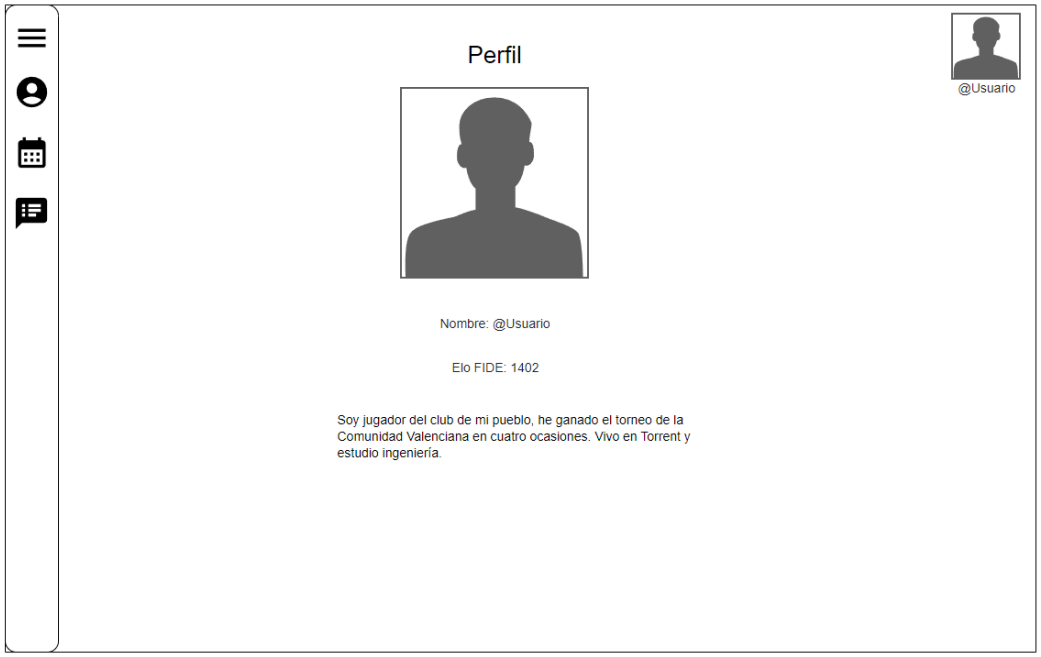

Figura 11. Ventana de perfil

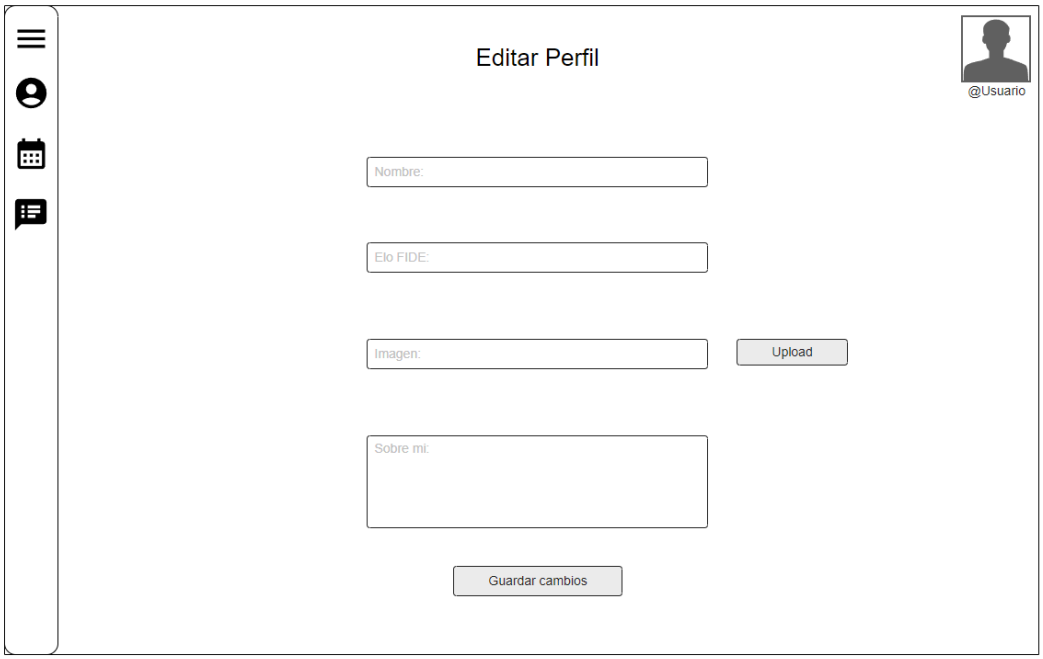

Figura 12. Ventana de editar perfil

En la Figura 13, se muestra la ventana del calendario y, en la Figura 14, la ventana de creación de un evento.

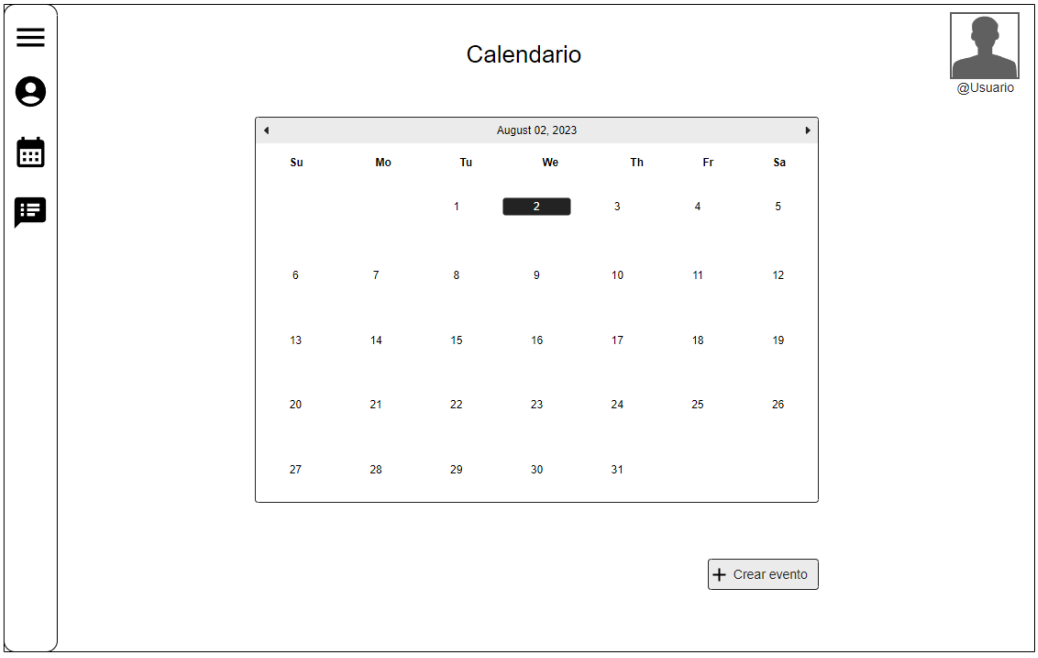

Figura 13. Ventana del calendario

Figura 14. Ventana de creación de un evento

Por último, en la Figura 15, se muestra la ventana de información de un evento. En el caso de ser un evento creado por el usuario, este tiene un botón para poder borrar el evento. En caso de ya estar apuntado al evento, el usuario tiene a disposición un botón para poder borrar su participación en el evento.

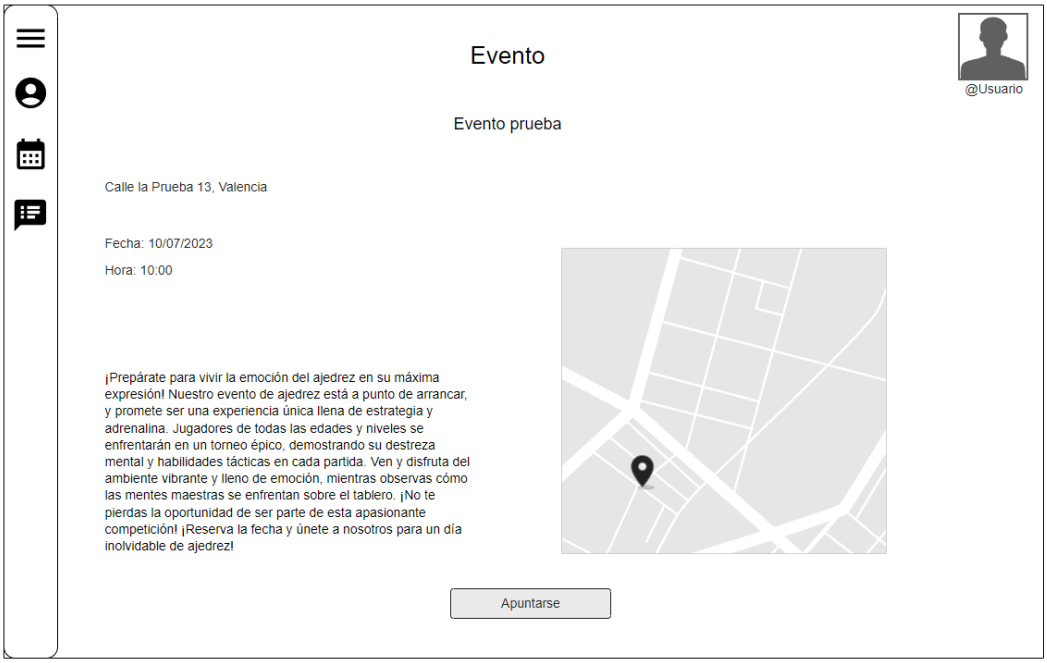

Figura 15. Ventana de información de un evento

3.6. Plan de trabajo

El desarrollo de la aplicación se ha llevado a cabo con una metodología ágil y por sprints. Cada uno de estos sprints tiene una duración de 2 semanas. En el primer sprint se ha llevado a cabo la especificación de requisitos y el modelaje de todos los diagramas. En el resto de sprints se desarrolla parte de la aplicación y al final de los mismos un periodo de pruebas para las funcionalidades desarrolladas.

Para una mejor estimación de la duración del proyecto y una mejor gestión de los tiempos, se ha desarrollado un diagrama de Gantt (ilustrado en la Figura 16).

Figura 16. Diagrama de Gantt

3.7. Presupuesto

Se ha realizado una estimación de lo que costará el proyecto en horas de trabajo (ver Tabla 20). En esta estimación, no se tiene en cuenta el gasto de equipos informáticos. Tampoco se tiene en cuenta el valor de las plataformas usadas, dado que se han usado las que tenían un plan gratuito.

Desarrollo del proyecto		
Tarea	**Horas**	**Coste** (20€/hora)
Especificación de requisitos y modelaje	25	500
Prototipado	10	200
Creación de la infraestructura	10	200
Registro e inicio de sesión	15	300
Publicar posts	20	400
Publicar eventos	18	360
Generar calendario	30	600
Mapa de evento	30	600
Editar y borrar post	15	300
Editar y borrar evento	20	400
Editar perfil	15	300
CSS	10	200
Pruebas	10	200
Total	**228 h**	**4.560€**

Tabla 20. Presupuesto

4. Diseño de la solución

En este capítulo se habla del diseño de la aplicación. En primer lugar, se describe la arquitectura elegida y, a continuación, con más detalle, el diseño.

4.1. Arquitectura del sistema

Para este proyecto se ha decidido usar una arquitectura hexagonal (6). De este modo, se aísla completamente el dominio y se puede cambiar cualquier tecnología, tanto de persistencia como conexión, con el frontend, de manera que no se tenga que hacer ningún cambio en la lógica de negocio.

La arquitectura hexagonal, como puede verse en la Figura 17, consta de los siguientes componentes:

- Dominio (o Core): Esta es la parte central y más importante de la arquitectura hexagonal. Contiene las reglas de negocio y la lógica del dominio de la aplicación. Aquí se definen los objetos de dominio, las entidades, los agregados y los servicios que encapsulan la funcionalidad clave del negocio. Es independiente de las tecnologías y detalles de implementación. Se comunica con el resto de partes del sistema mediante el patrón “ports and adapters” que viene implícito en este tipo de arquitectura.
- Puertos (Ports): Los puertos son interfaces que definen contratos entre el dominio y el mundo exterior. Estos contratos permiten la comunicación bidireccional entre el dominio y las capas externas. Los puertos definen los puntos de entrada y salida del sistema. Pueden ser puertos de entrada, que son utilizados por componentes externos para enviar datos y solicitudes al dominio, o puertos de salida, que permiten que el dominio envíe datos a componentes externos, como bases de datos, servicios externos o interfaces de usuario.
- Adaptadores (Adapters): Los adaptadores son implementaciones concretas de los puertos definidos en el dominio. Actúan como intermediarios entre el dominio y los recursos externos.
- Aplicación (Application): Esta capa es responsable de orquestar la interacción entre los puertos y adaptadores. Aquí se gestionan las transacciones y flujos de control para cumplir con los casos de uso y las solicitudes del usuario. El objetivo principal de esta capa es mantener separada la lógica de negocio y la tecnología usada para conectar con el frontend.
- Infraestructura (Infraestructure): Es la capa encargada de conectarse con la infraestructura diseñada para la aplicación, normalmente la tecnología de persistencia.

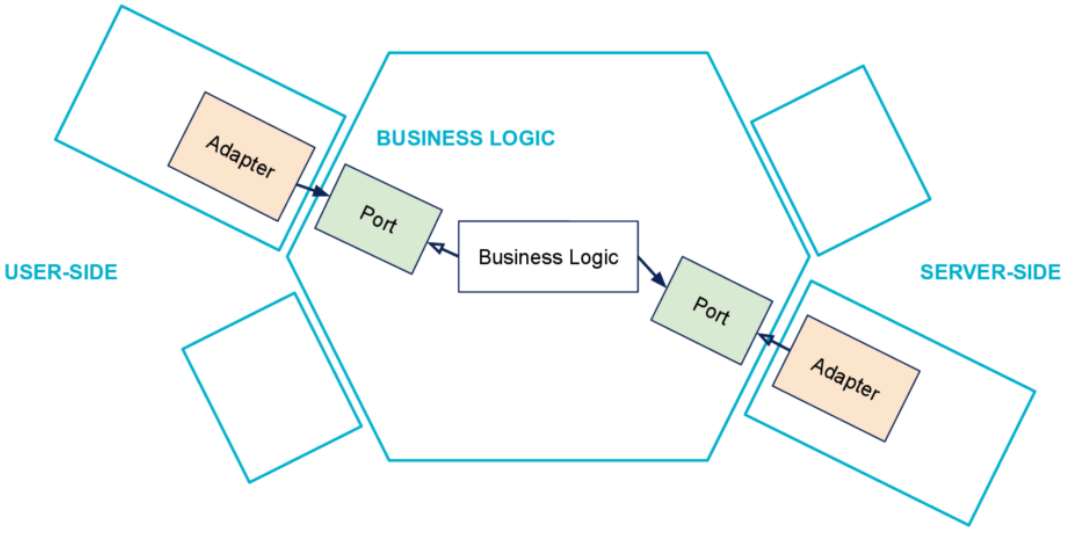

Figura 17. Arquitectura hexagonal (6)

4.2. Diseño detallado

En esta sección se detalla el diseño capa por capa: persistencia, dominio y aplicación.

4.2.1. Nivel de persistencia

A nivel de persistencia se usa la API JPA (Java Persistence API[5]) tanto para crear el diseño de la base de datos como para acceder a ella mediante las consultas JPQL (Java Persistence Query Language). La comunicación con el dominio se realiza mediante el patrón ports and adapters. El diagrama de clases de la aplicación es el que se muestra en la Figura 18.

[5] https://www.oracle.com/java/technologies/persistence-jsp.html

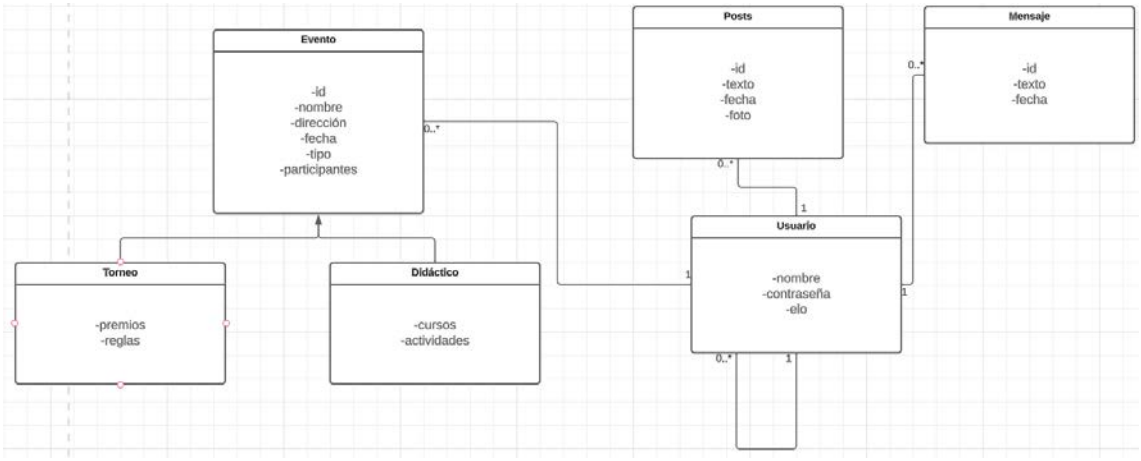

Figura 18. Diagrama de clases

Las 4 clases principales de la aplicación son:

- Post: Es la clase que representa el mensaje publicado en el tablón. Está compuesto de los atributos: id, texto, fecha y foto.
- Usuario: Representa, como su nombre indica, los usuarios que se registran y usan la aplicación. Está compuesto por los atributos: nombre, contraseña y elo.
- Evento: Esta clase representa el evento creado por un usuario. Como vemos en sus atributos indican la fecha y dirección. Esto deja claro que el evento es presencial. Los atributos son: id, nombre, dirección, fecha, tipo y participantes.
- Mensaje: Representa los mensajes que se mandan en el chat privado. Sus atributos son: id, texto y fecha.

Además, hay dos clases más, Torneo y Didáctico, que permiten clasificar el tipo de evento.

4.2.2.Nivel de aplicación

El nivel de aplicación en la arquitectura hexagonal implica implementar la lógica de negocio específica de la aplicación y coordinar la interacción entre los puertos y adaptadores para cumplir con los casos de uso y las solicitudes del usuario. Al aplicar la API REST en este nivel, se definen los recursos, las operaciones CRUD, las rutas y los métodos HTTP para acceder a esos recursos. Los controladores o endpoints se encargan de recibir las solicitudes HTTP, llamar a los servicios de aplicación, y enviar las respuestas al cliente. Los servicios de aplicación coordinan la lógica de negocio y se comunican con el dominio a través de los puertos definidos. Todo esto permite una interfaz clara y estandarizada para que los clientes interactúen con la lógica de negocio de la aplicación.

4.2.3. Nivel de dominio

El nivel de dominio en la arquitectura hexagonal es el núcleo de la aplicación, donde se encuentra la lógica de negocio y las reglas que rigen su funcionamiento. Está compuesto por objetos de dominio, agregados, servicios de dominio y puertos de dominio. Esta capa es independiente de la infraestructura y tecnologías utilizadas, lo que la hace más flexible y fácil de mantener. O lo que es lo mismo, el dominio no puede llamar a nada externo a él, pero todo lo externo puede llamar al dominio mediante sus puertos.

4.3. Tecnología utilizada

4.3.1. Parte backend

La tecnología usada para desarrollar la parte backend ha sido Java en su framework Spring Boot[6], con una arquitectura hexagonal ya explicada anteriormente. En la parte de comunicación con el frontend se usa una API REST para que se realice mediante peticiones HTTP.

Para la parte de la infraestructura se utiliza la API JPA (Java Persistence API) para diseñar, mediante clases entidad, la base de datos y realizar las consultas con JPQL (Java Persistence Query Language).

En la parte de dominio se usa Java con su librería Lombok[7] para ayudar a generar los constructores y las llamadas a los atributos de las clases del dominio.

Las pruebas unitarias se realizan con JUnit[8] y Mockito[9]. Mockito permite aislar completamente las funcionalidades de cualquier llamada a la base de datos y realizar una prueba unitaria real.

[6] https://spring.io/projects/spring-boot

[7] https://projectlombok.org/

[8] https://junit.org/junit5/

[9] https://site.mockito.org/

4.3.2. Parte frontend (presentación)

Para esta parte se utiliza el lenguaje TypeScript[10], que es una forma ampliada de JavaScript, con su framework Angular[11].

Angular es un potente y completo framework de desarrollo de aplicaciones web. Su arquitectura basada en componentes facilita el desarrollo modular, reutilizable y mantenible de aplicaciones. Una ventaja destacada es el binding bidireccional, que permite una sincronización automática entre los datos y la interfaz de usuario. Además, al estar desarrollado en Typescript, se obtiene el beneficio de un tipado estático y una mayor productividad en el desarrollo. Angular también ofrece soporte para enrutamiento, lo que facilita la creación de aplicaciones de una sola página y una navegación eficiente. La inyección de dependencias permite una gestión más sencilla y reutilizable de componentes y servicios. Con el Angular CLI, se puede crear, probar y construir proyectos fácilmente, ahorrando tiempo en configuraciones. La comunidad activa y la abundante documentación son una gran ventaja para aprender y obtener ayuda en línea. Finalmente, Angular está optimizado para el rendimiento, proporcionando herramientas como la detección de cambios y la compilación AOT para mejorar el rendimiento de las aplicaciones.

4.3.3. Persistencia

Para la parte de persistencia se ha usado una base de datos relacional MySQL en el host gratuito de Railway[12]. La base de datos relacional nos ayuda a mantener la información bien estructurada y también ayuda a filtrar de forma avanzada en las consultas, lo que ayuda a quitar algo de carga a la capa de dominio.

4.3.4. Creación de modelos

Para la creación del modelo de dominio y el modelo de base de datos se ha usado la aplicación online Lucidchart[13].

Para la creación de los casos de uso y los prototipos se ha usado la aplicación online Moqups[14]. Permite hacer casi todos los modelos que hay en el desarrollo de software.

---

[10] https://www.typescriptlang.org/

[11] https://angular.io/

[12] https://railway.app/

[13] https://www.lucidchart.com/

[14] https://moqups.com/es/

4.3.5. Entorno de trabajo

Para desarrollar el código del backend se ha usado IntelliJ, una herramienta para desarrollo en Java del paquete JetBrains[15], que permite una variedad infinita de plugins que ayudan a desarrollar de una forma más ágil.

Para la parte frontend se ha usado WebStorm, del mismo paquete JetBrains, con las mismas ventajas que IntelliJ pero para el desarrollo orientado a web.

Para la gestión de la base de datos se ha usado la propia herramienta online que ofrece el host Railway.

4.3.6. Administración del desarrollo

Se ha elegido la herramienta usada en las asignaturas PSW y PIN del Grado de Ingeniería Informática, Worki[16]. Permite administrar un backlog y los sprints, así como los tiempos estimados para cada tarea y el coste real de realización de la misma. Tiene gráficas que ayudan a saber cómo está yendo el sprint o el proyecto en general.

[15] https://www.jetbrains.com/

[16] http://tuneupprocess.com/

5. Desarrollo

En este capítulo se detalla cómo se ha desarrollado el proyecto, comentando por encima las tareas realizadas en cada sprint, las dificultades y las soluciones encontradas. También se habla de las herramientas usadas en el desarrollo y, en el último apartado de este capítulo, se presenta un ejemplo de funcionamiento de la aplicación.

5.1. Sprint 1

El Sprint 1 se enfocó en varios aspectos cruciales para sentar las bases del proyecto. Primero, se trabajó en la especificación de requisitos para delinear claramente las necesidades y expectativas del sistema. Paralelamente, se crearon los repositorios necesarios en plataformas de control de versiones para facilitar el seguimiento del código. Una tarea importante fue la selección de hosts adecuados para alojar la aplicación, considerando factores como coste (en este caso gratis), disponibilidad y fiabilidad. Por último, se diseñaron los prototipos iniciales de la aplicación, ofreciendo una vista previa del diseño y la funcionalidad que se espera en la versión final. Este primer sprint fue esencial para establecer un fuerte cimiento sobre el cual construir el resto del proyecto.

5.2. Sprint 2

En el segundo sprint se comenzó con el desarrollo de la aplicación. Se quiso empezar con algo básico de una aplicación como la funcionalidad de inicio de sesión y registro de usuarios. En esta funcionalidad surgieron los primeros problemas típicos de cuando se intenta conectar el frontend con el backend por primera vez en el proyecto. Estaba claro que iba a ser mediante una API REST, pero, no estaba clara la librería en el front. Al final se eligió la librería HttpClient[17], una librería muy utilizada y, gracias a ello, con mucho soporte en internet. También en este sprint se desarrolló la funcionalidad de publicar posts. En esta funcionalidad hubo problemas en las relaciones en la representación de la base de datos mediante entidades en el backend. Una vez arregladas no hubo muchos problemas porque la lógica de negocio sí que estaba bien implementada. Luego de implementar estas funcionalidades se hicieron pruebas de las mismas, tanto unitarias como de integración. Se detalla en el capítulo 6 como se han implementado las pruebas en el proyecto.

[17] https://angular.io/api/common/http/HttpClient

5.3. Sprint 3

En el tercer sprint, se comenzó el desarrollo de la publicación de eventos y toda la funcionalidad de apuntarse a los mismos. La gran dificultad de esto ha sido generar el calendario; no se conseguía que el calendario cargara bien los días. Al final, acabó funcionando, buscando en foros como StackOverflow[18] y refinando la solución encontrada con ChatGPT[19]. Pero luego vino el problema de guardar las fechas. En TypeScript, las fechas se manejan con el tipo "Date", mientras que en la base de datos y el backend se manejan con "LocalDate". Esto provocó que se tuviesen que hacer formateos de fecha en el frontend para poder pasarlas al backend ya con un formato procesable. Respecto al mapa, no hubo demasiado problema: es un mapa que ya se utilizó con anterioridad en alguna asignatura del Grado, simple y que funciona muy bien para la funcionalidad que tiene en la aplicación. Y, como en el sprint anterior, se termina con pruebas unitarias y de integración de la funcionalidad implementada en este sprint, junto con la ya implementada en el sprint anterior.

5.4. Sprint 4

En el cuarto sprint, se empezó el desarrollo de la funcionalidad de editar y borrar posts. Es algo sencillo; simplemente es un formulario que recoge el texto modificado y lo sustituye por el que estaba. Con esto no hubo ninguna dificultad. Otra funcionalidad que se desarrolló es la de editar eventos. Esta sí tuvo algo de dificultad, sobre todo en la parte de cambiar la fecha del evento. Aquí, el problema se ha dado en que no se cargaba bien la fecha en el calendario. Tras investigar un poco, se descubrió que era otra vez por el formato. En este caso, tenía que formatear a "Date" y luego volver a formatear a "LocalDate". Después de esto, no tenía más complejidad que la funcionalidad de editar posts, solo que esta tenía más campos que recoger. También se desarrolló la funcionalidad de editar perfil sin problemas, después de haber resuelto los potenciales problemas que podían dar los formularios en las funcionalidades anteriores. Para terminar el proyecto, se ha dedicado una tarea exclusiva a modificar el CSS para dejarlo atractivo, minimalista (gusto personal) y adaptado a dispositivos móviles. Los únicos problemas que se han tenido son la adaptación a los dispositivos móviles. A pesar de que Angular da bastantes facilidades para esto, debido a la inexperiencia en esta tecnología, surgieron problemas que se solucionaron conforme aprendía cómo se comportaba la misma. Por último, como en todos los sprints anteriores, se realizaron pruebas de la funcionalidad implementada en este sprint y en los anteriores.

[18] https://stackoverflow.com/
[19] https://chat.openai.com/

5.5. Herramientas usadas

Para el desarrollo del backend, hecho en Java, se ha usado el entorno de desarrollo IntelliJ. Una herramienta mundialmente usada para el desarrollo en este lenguaje y con la que ya se tenía experiencia. Tiene bastantes complementos, es altamente customizable y hace muy intuitiva la programación.

Para el frontend se ha usado WebStorm. No es el mejor para programar en Angular (se recomienda usar VSCode para esta tecnología), pero funciona también muy bien y además, es muy parecido a IntelliJ por ser de la misma empresa. Funciona muy bien para JavaScript, por lo tanto, para TypeScript también, y por todo esto ha sido el elegido.

Para la persistencia se ha elegido el host de Railway.app. Host con un plan de prueba gratuito, con un panel de control intuitivo, con representación gráfica de las tablas y altamente disponible. Te deja elegir la tecnología con la que quieres la base de datos, estando las más usadas en el mundo. Para este caso se ha elegido MySQL. Fácil de usar y con mucha comunidad para poder solventar las dudas en la web. La única desventaja que se puede poner a este host es que tiene algo de retraso en las peticiones y eso hace que la aplicación tarde en cargar algunas funcionalidades.

Para probar las peticiones se ha usado la conocida app de Postman. Rápida, intuitiva y ofrece muchas posibilidades para probar las peticiones más complejas que puedas implementar.

Para el desarrollo de los prototipos se ha usado Lucidchart. Una app sencilla de usar para cosas básicas, pero para utilizarla de forma profesional se tiene que acceder a su versión premium. Y esta es una de sus desventajas, porque en su versión gratuita solo se tiene espacio para tres proyectos. Si se quieren más proyectos de forma simultánea, se tendría que adquirir la versión de pago.

5.6. Ejemplo de funcionamiento

En esta sección se muestra un ejemplo de funcionamiento de una de las funcionalidades de la aplicación. Se ha elegido la funcionalidad de crear un evento.

En primer lugar iniciamos sesión (ver Figura 19) y la aplicación muestra la pantalla del Dashboard donde se encuentra el tablón con los posts (ver Figura 20).

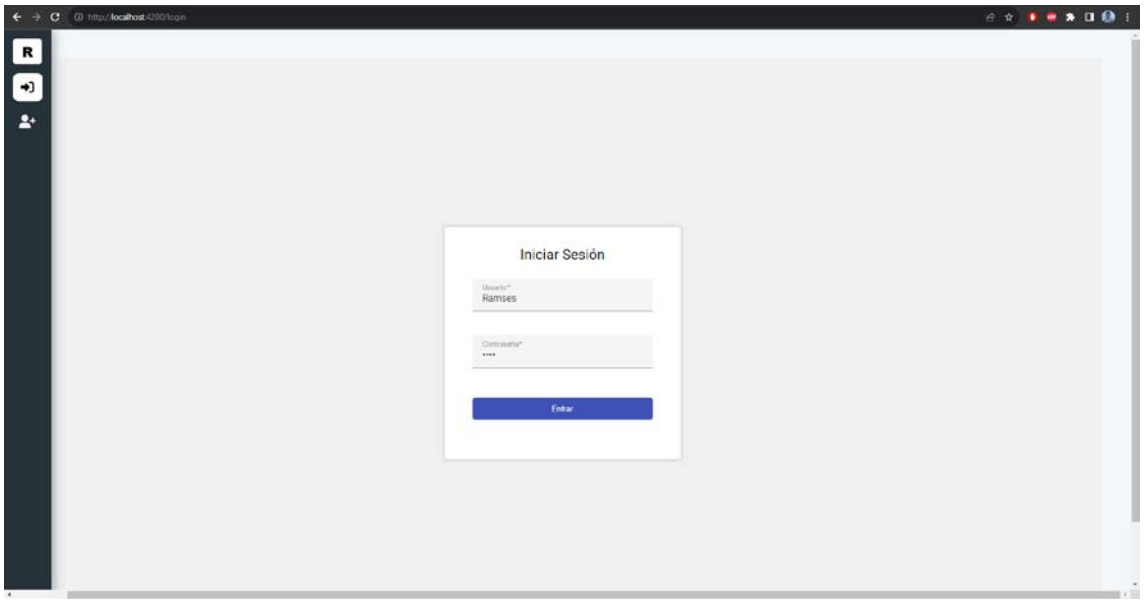

Figura 19. Inicio de sesión

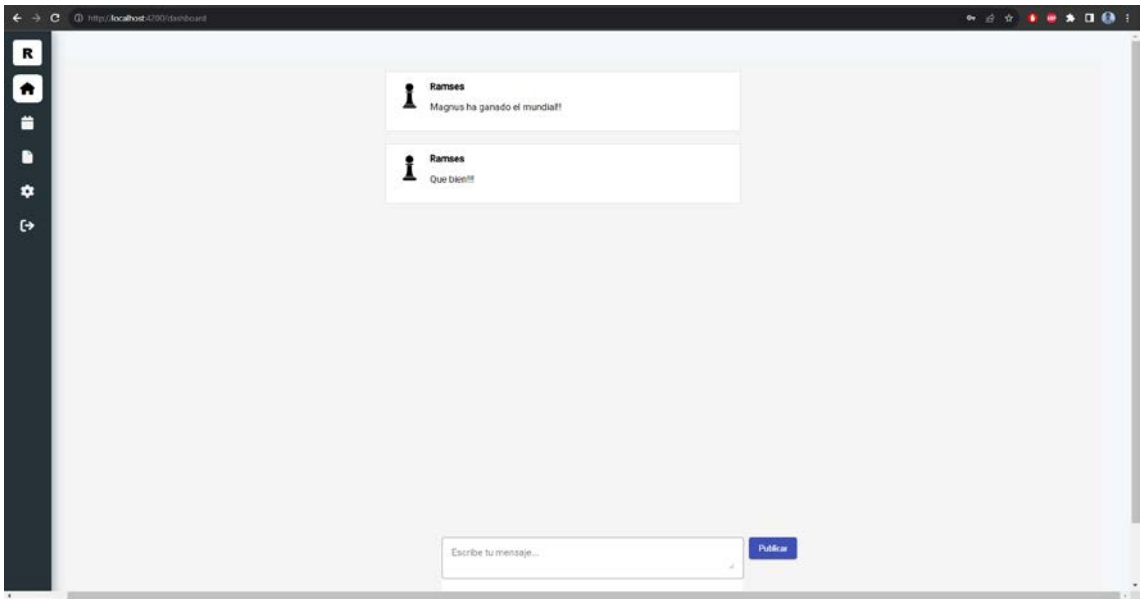

Figura 20. Dashboard y Tablón

A continuación se selecciona el calendario que se encuentra en la barra de navegación de la izquierda (ver Figura 21). Esto nos va a llevar a la pantalla del calendario donde están agendados todos los eventos a los que un usuario concreto está apuntado.

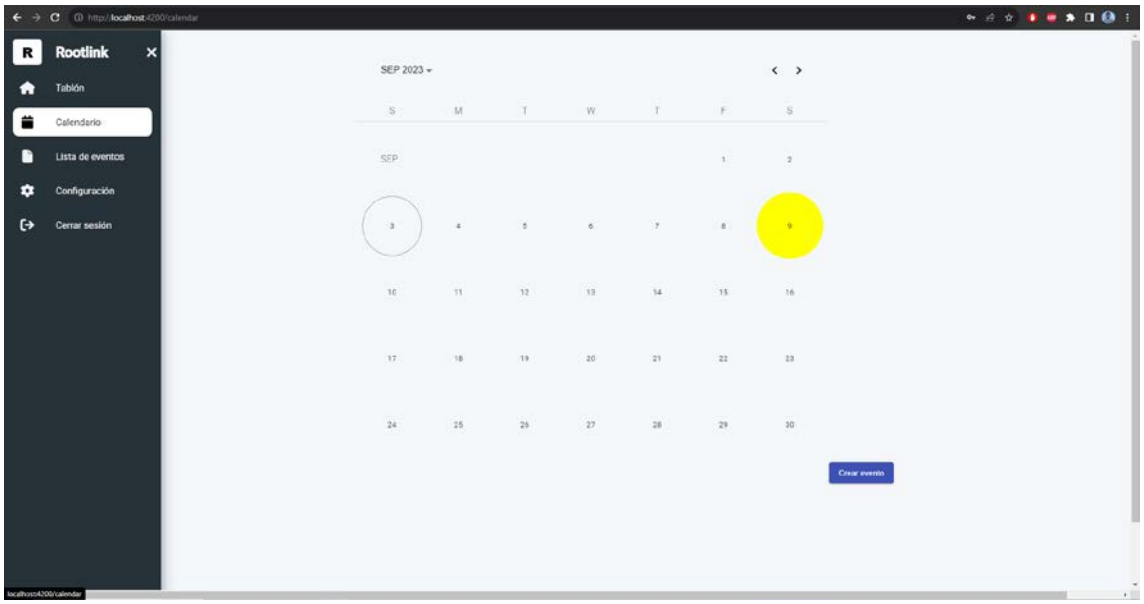

Figura 21. Calendario

En esta pantalla se puede ver que ya tenemos un evento agendado el día 9. Se quiere crear un nuevo evento, para ello se pulsa en el botón de “Crear evento” y se abre la ventana de crear evento (ver Figura 22).

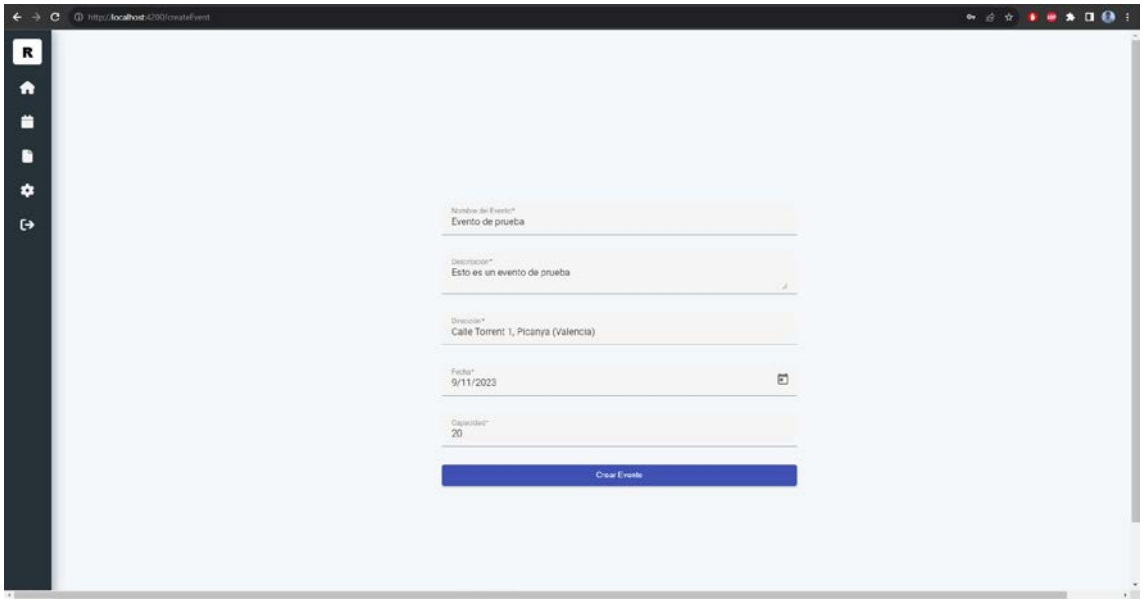

Figura 22. Crear evento

Cuando se pulsa en crear evento, el evento se crea y redirige a la ventana del calendario, pero esta vez, con el evento marcado como muestra la Figura 23. Esto ocurre porque la aplicación está configurada para que el usuario creador del evento esté apuntado por defecto al evento que crea.

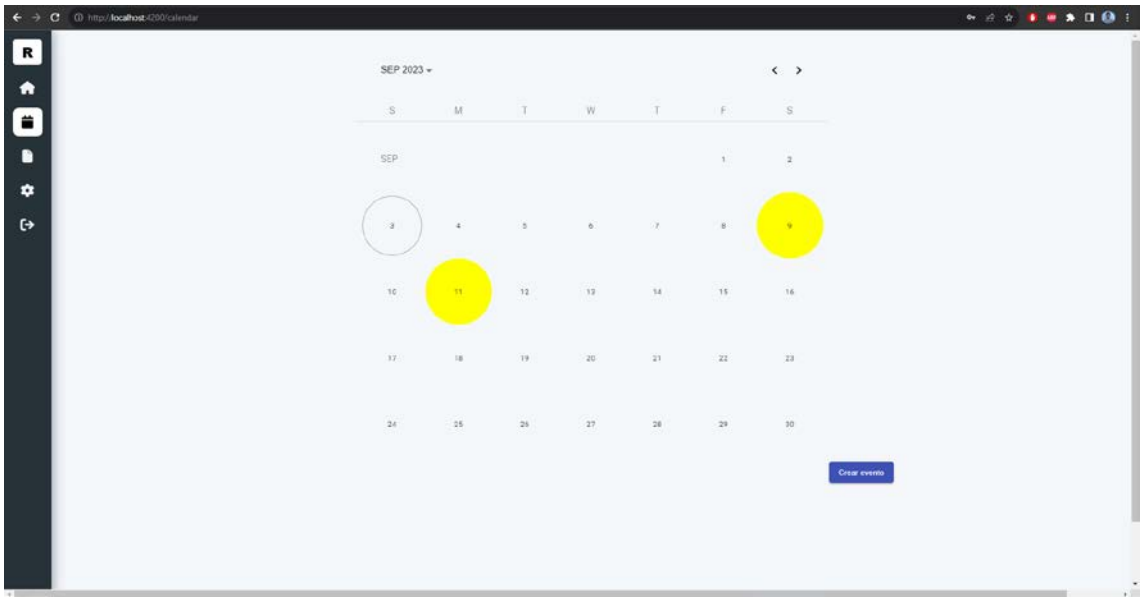

Figura 23. Calendario (con evento creado)

Si se selecciona el día marcado en el calendario, la aplicación redirigirá a la ventana de información del evento. Donde hay un mapa con la ubicación del evento como se muestra en la Figura 24.

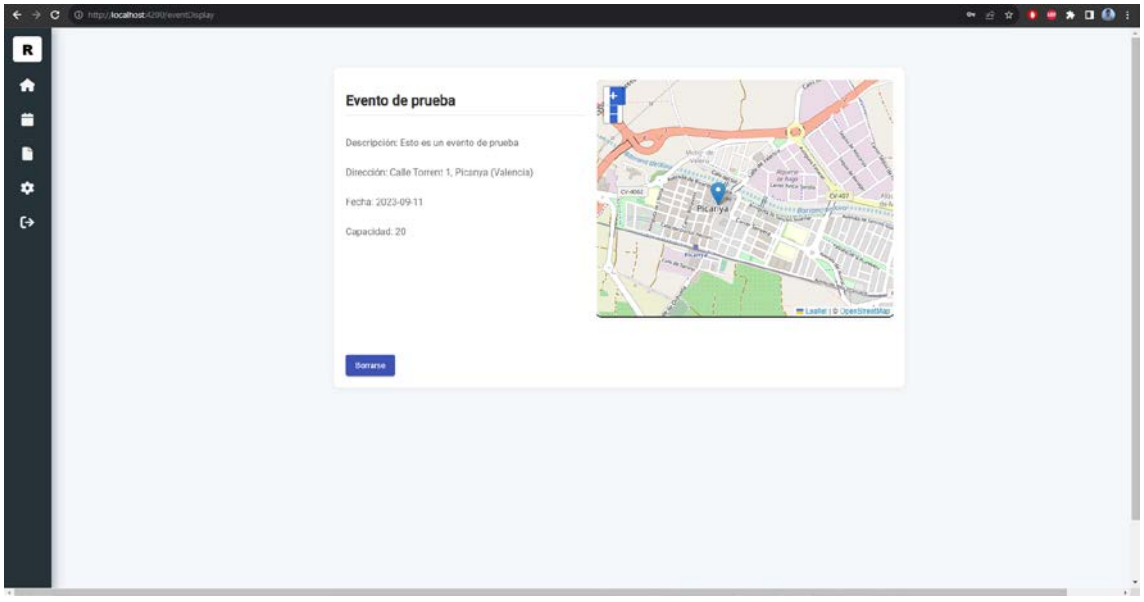

Figura 24. Información del evento

Se puede observar que hay un botón para poder borrarse del evento. En caso de que el usuario no esté apuntado, en el botón pondrá “Apuntarse”, y pulsando este botón apuntará al usuario al evento.

6. Pruebas

Durante el desarrollo del proyecto, más concretamente en el final de los sprints, se ha dedicado una tarea solo para probar la funcionalidad desarrollada. Una forma técnica de probar la funcionalidad desarrollada es con las pruebas unitarias (7). En este capítulo se describen las pruebas unitarias realizadas a las funcionalidades de la aplicación, así como también las pruebas realizadas directamente a la aplicación ya montada y funcionando.

6.1. Pruebas unitarias

En la Figura 25 tenemos un ejemplo de las pruebas unitarias que comprueban (en este caso) la lógica de negocio del inicio de sesión de la aplicación. En estas pruebas se comprueban todos los casos con un final “no deseado” y al final una con un final de “éxito”.

```
new *
@Test
void loginWithPasswordWrong() {
    String name = "testing";
    String password = "passwordWrong";

    assertThrows(RuntimeException.class,
            () -> userUseCase.login(name, password));
    verify(userDbPort).login(any(), any());
}

new *
@Test
void loginWithNameWrong() {
    String name = "testingg";
    String password = "password";

    assertThrows(RuntimeException.class,
            () -> userUseCase.login(name, password));
    verify(userDbPort).login(any(), any());
}

new *
@Test
void loginWithAllCorrect() {
    String name = "testing";
    String password = "password";
    User user = User.builder().name(name).password(password).build();
    when(userDbPort.login(name, password)).thenReturn(user);

    User userLogged = userUseCase.login(name, password);

    verify(userDbPort).login(any(), any());
    assertEquals(name, userLogged.getName());
    assertEquals(password, userLogged.getPassword());
}
```

Figura 25. Test de inicio de sesión

Una vez lanzadas todas las pruebas, tendríamos un resultado como el de la Figura 26. En caso de que algo en las pruebas no haya salido como se espera, en esta representación saldría con una cruz en rojo y una descripción del error. Esto es muy útil para facilitar un futuro mantenimiento de la aplicación.

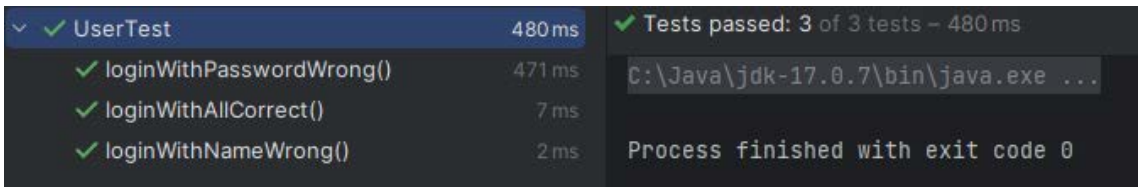

Figura 26. Resultados de los test

6.2. Pruebas de la app

También se llevan a cabo las pruebas en la propia aplicación. Este caso es simple, se toma el papel de QA se prueba la funcionalidad de la aplicación, como si de un usuario normal se tratase, en busca de posibles errores. Para volver al ejemplo del apartado anterior, se accede a la ventana de inicio de sesión y se prueba el comportamiento de la aplicación en los casos de fallo y uno en caso correcto. En la Figura 27 se muestra uno de los casos en los que el usuario es incorrecto y el mensaje que avisa al usuario de su error.

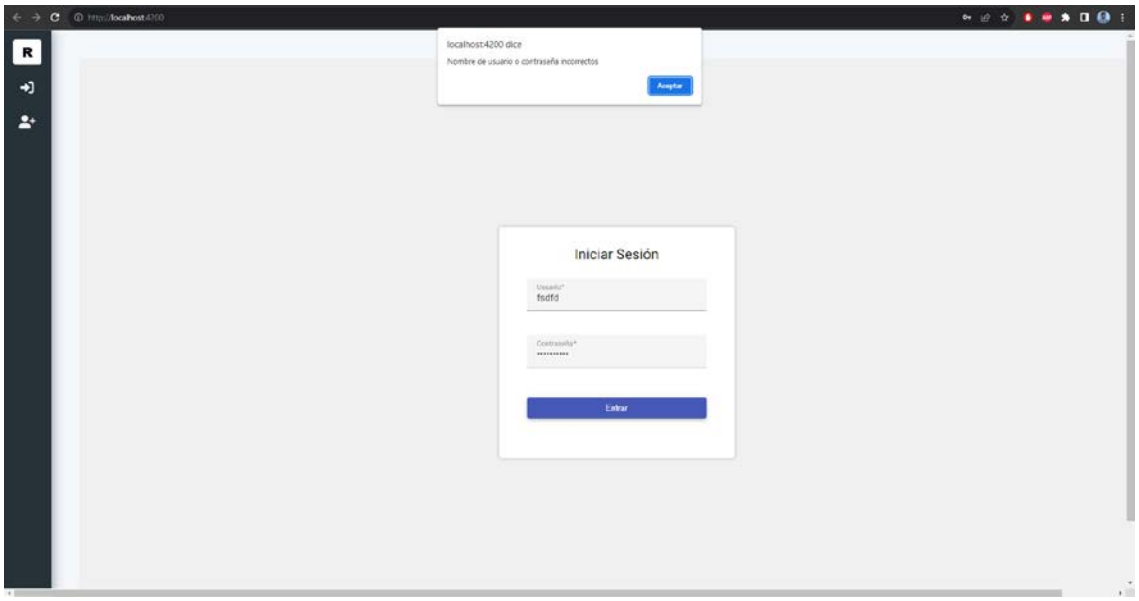

Figura 27. Pruebas en la app

7. Conclusiones

Al comienzo de este proyecto se planteó como objetivo principal el desarrollo de una red social que ayudase a la comunidad ajedrecística a reunirse para jugar ajedrez presencial y que los usuarios pudieran enterarse de las noticias sobre el ajedrez. Finalmente, se ha conseguido dicho objetivo. En particular, los usuarios pueden editar su perfil, publicar mensajes, crear eventos presenciales y apuntarse a dichos eventos. Además, se ha integrado un calendario para cada usuario dónde consultar los eventos a los que se ha apuntado, así como un mapa con la ubicación del evento.

Hacerse un hueco en este sector no es fácil, la comunidad del ajedrez es pequeña y, para la mayoría de la gente, poco atractiva. Pero se ha encontrado una oportunidad en el mercado. La mayoría de portales se centran más en la enseñanza online o en jugar online. Y, aunque tienen su parte de red social, no es su funcionalidad principal y, por ello, no tienen tan desarrollada esa parte. Ahí es donde entra Rootlink que, además de todo lo anterior, se enfoca al ajedrez mas tradicional, un ajedrez presencial.

7.1. Relación del desarrollo con los estudios cursados

Con relación a mis estudios cursados, he usado las metodologías ágiles de las asignaturas PIN y PSW (Scrum y Kanban) para organizar los tiempos del proyecto. Las pruebas con Junit de la asignatura MES, aunque complementadas con Mockito para que sean realmente unitarias. En la asignatura CSW aprendí como hacer una aplicación con calidad y esto me ha llevado a hacer una aplicación lo más profesional posible aplicando, por ejemplo, las heurísticas de Nielsen (8). Y para el análisis de requisitos, obviamente se ha aplicado todo lo aprendido en la asignatura AER, todo el primer sprint se basa en esta asignatura. Tampoco olvidarme de las asignaturas que te enseñan los fundamentos básicos para aprender a programar (PRG, LTP y CSD). Es verdad que en mi caso ya tenía unos conocimientos básicos, pero estas asignaturas me ayudaron a entender qué es realmente programar y sus problemas más comunes (como la eficiencia o los problemas que causan las aplicaciones multi hilo y asíncronas). Sin ellas no podría haber abordado este proyecto.

7.2. Nivel personal

El proyecto me ha enriquecido en conocimientos, sobre todo en la parte de frontend. En la empresa en la que trabajo soy backend developer (aunque también desarrollo algunas partes de frontend) y tener el reto de hacer el frontend de toda una aplicación me ha dado

la oportunidad de indagar en la tecnología Angular, universalmente usada. También poner en práctica por mí mismo las metodologías ágiles y darme cuenta que tienen su utilidad, aunque es verdad que están más pensadas para un equipo que para una sola persona.

No solo es a nivel técnico lo ganado en este TFG. El tener que aprender a administrarme los tiempos para compaginar mi trabajo con el desarrollo de este proyecto ha sido todo un reto. Y me ha hecho adquirir un sentido de la responsabilidad aún mayor que el que tenía, si cabe.

8. Trabajos futuros

No todo ha salido como se quería inicialmente. En este capítulo se describen las funcionalidades no implementadas por falta de tiempo y que se pretende implementar en próximas versiones de la aplicación. Dichas funcionalidades son las siguientes:

- Disponer de un chat privado. Es una funcionalidad muy importante que se ha quedado fuera y que será una prioridad máxima en una futura versión del proyecto.
- Subir imágenes a un bucket. En la mayoría de los casos pedían que ingresara mi tarjeta de crédito, algo que no estaba dispuesto a hacer. Y, en las páginas que no me la pedían, no daban los requisitos que yo necesitaba para este proyecto.
- Distinguir entre evento didáctico y evento de torneo. Aunque es importante, estando el apartado de descripción, era algo de lo que se podía prescindir y será añadido en futuras entregas del proyecto.

Bibliografía

1. Análisis DAFO. [En línea] [Citado el: 9 de 8 de 2023.] https://es.wikipedia.org/wiki/An%C3%A1lisis_FODA.

2. Drumond, Claire. Qué es Scrum y cómo empezar. *Atlassian.* [En línea] [Citado el: 9 de 8 de 2023.] https://www.atlassian.com/es/agile/scrum.

3. NimbleWork. ¿Qué es la programación extrema (XP) y sus valores, principios y prácticas? *Nimble.* [En línea] [Citado el: 9 de 8 de 2023.] https://www.nimblework.com/es/agile/programacion-extrema-xp/.

4. Ortega, Lara. Metodología RUP: ¿Qué es, cúal es su objetivo y cómo se utiliza? *Lean Management Blog.* [En línea] [Citado el: 9 de 8 de 2023.] https://lean-management.site/rup/.

5. Munir, Hussan, y otros. An experimental evaluation of test driven development vs. test-last development with industry professionals. *EASE '14: 18th International Conference on Evaluation and Assessment in Software Engineering.* s.l. : Association for Computing Machinery, 2014. Vol. 1, 50, págs. 1-10.

6. Alliaume, Erwan y Roccaserra, Sébastien. Hexagonal Architecture: three principles and an implementation example. *Le Blog des Octos.* [En línea] 15 de 10 de 2018. [Citado el: 9 de 8 de 2023.] https://blog.octo.com/hexagonal-architecture-three-principles-and-an-implementation-example/.

7. Team, KeepCoding. ¿Qué son las pruebas unitarias de software? [En línea] [Citado el: 20 de 8 de 2023.] https://keepcoding.io/blog/que-son-las-pruebas-unitarias-de-software/.

8. Serafinelli, Stefano. Los 10 principios heurísticos de Nielsen explicados con ejemplos. [En línea] [Citado el: 18 de 8 de 2023.] https://www.teacuplab.com/es/blog/los-10-principios-heuristicos-de-nielsen-explicados-con-ejemplos/.

ANEXO. OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

ODS 3: Salud y Bienestar

El tablón de la red social podría dedicar una sección a compartir artículos científicos y noticias que resalten los beneficios del ajedrez para la salud mental, como la mejora de la concentración y la reducción del estrés. En cuanto a los eventos presenciales, estos encuentros de "ajedrez a la vieja usanza" pueden fomentar el bienestar mental al ofrecer un espacio para la interacción social cara a cara, algo que a menudo falta en nuestra vida digital. El simple acto de sentarse frente a un oponente y jugar una partida puede contribuir significativamente a la reducción del aislamiento social, mejorando el bienestar emocional de los participantes.

ODS 4: Educación de Calidad

El tablón también podría ser una fuente de material educativo de alta calidad. Podrían compartirse guías y estrategias, análisis de partidas famosas y otras formas de contenido didáctico que fomenten un aprendizaje autodirigido y de calidad. Los eventos presenciales, en particular los torneos, se convierten en escenarios ideales para el aprendizaje práctico. Estas competencias no solo permiten a los jugadores aplicar lo que han aprendido en línea, sino que también ofrecen la oportunidad de recibir retroalimentación en tiempo real de otros jugadores más experimentados. En conjunto, estos elementos contribuirían a una educación ajedrecística de alta calidad, accesible para todos los miembros de la comunidad.