
**Universidad de Valladolid**

**ESCUELA DE INGENIERÍA INFORMÁTICA**

**GRADO EN INGENIERÍA INFORMÁTICA**
**Mención en Ingeniería del Software**

---

# Aplicación web como adaptación de un juego educativo sobre la gestión de la deuda técnica

---

**Alumno: Rebeca Hernando Brecht**

**Tutor: Yania Crespo González-Carvajal**

...

# Agradecimientos

A mi familia por todo su apoyo durante todo mi paso por la universidad.

A mis amigos, los de dentro y fuera de la universidad, que siempre me han escuchado, animado y han confiado en mi.

A mis compañeros durante el grado que, tras muchas horas de trabajo y esfuerzo codo con codo, se han convertido en amigos.

A todos los docentes de la universidad que me han trasmitido sus conocimientos y han aportado su granito de arena en mi formación. En especial a mi tutora, Yania, que me ha ayudado en el desarrollo de este trabajo de fin de grado y durante estos cuatro años ha sido un referente de esfuerzo y dedicación.

Gracias.

# Resumen

Hard Choices es un juego educativo definido en el Software Engineering Institute (SEI) para el entrenamiento en la gestión de la deuda técnica. Se trata de un juego de tablero en el que, además, son muy relevantes las discusiones que se suscitan para el enriquecimiento de la experiencia. La gestión de la deuda técnica es un aspecto importante en la formación de los (futuros) Ingenieros de Software. Actualmente, una buena parte de las formaciones/entrenamientos para el desempeño profesional se realizan online y podrían beneficiarse de disponer este tipo de juegos educativos adaptados. El objetivo de este trabajo de fin de grado es diseñar una adaptación de Hard Choices como juego multijugador online y desarrollar una aplicación web que lo implemente.

Se ha desarrollado el diseño de la totalidad del sistema y se ha implementado la línea base del juego en la que se permite jugar simultáneamente entre 2 y 4 jugadores mas un moderador. Para ello se ha utilizado las tecnologías Angular CLI, para el *frontend*, Spring Boot, para le *backend*, y Firebase Realtime como base de datos no relacional.

# Abstract

Hard Choices is an educational game defined by the Software Engineering Institute (SEI) aimed at training in technical debt management. It is a board game in which discussions that arise play an important role on the enrichment of the experience. Technical debt management is important on the training of future software engineers. Currently, a great deal of professional performance trainings are done online, benefiting from these adapted educational games. The main goal of this paper is to design an adaptation of Hard Choices as an online multiplayer game and develop a web application which implements it.
The design of the entire system has been developed and the game's baseline has been implemented in which, 2 to 4 players and a moderator are can be played simultaneously. For this, the Angular CLI technologies have been used for the frontend, Spring Boot for the backend, and Firebase Realtime as a non-relational database.

# Índice general

# Lista de Figuras

# Lista de Tablas

# Capítulo 1

# Introducción

## 1.1. Contexto

La deuda técnica es un concepto nombrado por primera vez por Ward Cunningham en 1992 en *The WyCash Portfolio Management System*, el sistema de gestión de cartera WyCash, donde expone que para entender un programa con alto porcentaje de código inmaduro, se necesitan desarrolladores extremadamente especializados y, además, hace que el código sea muy inflexible. W.Cunningham dice que este tipo de código produce deuda y que un poco de la misma está bien, siempre y cuando se pague con refactorización en un periodo corto de tiempo ya que, si no se hace, se puede generar tal cantidad de deuda técnica que el proyecto puede ser inabarcable [29].

La deuda técnica puede ser voluntaria, deuda incurrida intencionalmente o involuntaria, debido a trabajos de baja calidad. Para solventarla existen 4 posibilidades según el tipo de la misma: ampliar funcionalidad, invertir en arquitectura, invertir en reducir defectos o no hacer nada y asumir la deuda técnica. Esto último siempre tiene consecuencias a largo plazo [13].

En Julio de 2021 se ha publicado un estudio del estado de la deuda técnica que dice que, de media, los desarrolladores emplean una jornada de trabajo para resolver la deuda técnica de los proyectos de su empresa. Esto tiene como consecuencia que los trabajadores estén desmotivados por la cantidad de tiempo que invierten en lidiar con errores provocados por la baja calidad del código. Como consecuencia de este estado de ánimo los desarrolladores son menos eficientes en su trabajo con todo lo que conlleva para el entorno de trabajo y la empresa [23].

Este proyecto se desarrolla en el marco de la asignatura Trabajo de Fin de Grado, concretamente en el grado de Ingeniería Informática en la Mención de Software. Como todo proyecto de desarrollo software ha de seguir unas fases para alcanzar el producto final, empezando por la planificación y el análisis y terminando por el diseño, la implementación y las

pruebas, así como el seguimiento de cada una de ellas [28]. Toda esta información se recoge en esta memoria, la estructura de la misma se explica en el apartado 1.4.

## 1.2. Motivación

El concepto de la deuda técnica explicado en la sección anterior es un gran olvidado en los procesos de desarrollo software, por eso surgió la idea de utilizar un juego para explicárselo a los alumnos que tendrán un futuro en el desarrollo de este tipo de proyectos. El juego escogido para esto es *Hard Choices*, creado por un grupo de ingenieros del SEI, que simula el ciclo de desarrollo de un producto software. Este juego fue creado para comunicar los conceptos de incertidumbre, riesgo y deuda técnica. En el capítulo 2 se da una explicación más exhaustiva sobre el juego y se detallan todos los aspectos importantes del mismo como las reglas de juego y la relación con la deuda técnica.

Al inicio de este proyecto no podía ser un juego en físico, debido a las restricciones socio-sanitarias impuestas por la pandemia y, en mi opinión, muchos de los aspectos que hemos cambiado para adaptarnos a ella están para quedarse como, entre otras cosas, lo relacionado con el trabajo y la educación a distancia o, al menos, muchas de las herramientas que durante este año se han empezado a incluir en todos los centros.

Es por todo esto que se ha decidido desarrollar una web en la que se pueda jugar al juego *Hard Choices*, uno de los juegos más completos para enseñar el concepto de la deuda técnica. Tener esta web permitirá a los profesores un fácil acceso al juego, sin necesidad de preparar material físico, lo que hace muchas veces un gasto ingente de papel y tiempo. Además, los alumnos interesados podrán jugar también fuera de la Escuela.

## 1.3. Objetivos

El objetivo principal de este proyecto es crear una web en la que se pueda jugar al juego educativo *Hard Choices* para que se pueda usar en las aulas, así como en la enseñanza a distancia online, como apoyo en la explicación del concepto de la deuda técnica.

### 1.3.1. Objetivos de la aplicación

- Adaptar las bases del juego de tablero *Hard Choices* en su totalidad para ser un juego como aplicación web multiusuario.
- Desarrollar las variaciones del juego para la web.
- Para potenciar la competitividad entre usuarios, disponer de un ranking.
- Reforzar los conceptos aprendidos de la deuda técnica mediante este juego.

- Diseño de una interfaz atractiva y responsive que se adapte a diferentes tamaños de dispositivos.
- Desarrollar una aplicación que sea compatible con los principales navegadores: Google Chrome, Mozilla Firefox y Safari.

### 1.3.2. Objetivos de formación

- Realizar todas las fases y la documentación de un proyecto software como preparación para el ejercicio de la profesión.

## 1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

**Capítulo 1 Introducción:** En este capítulo se explica de manera general en que consiste el proyecto, se enmarcar en un contexto, se explica la motivación que ha llevado a hacerlo, así como los objetivos y la estructura de la memoria.

**Capítulo 2 Hard Choices:** Se explica el juego original que se va a implementar en la web. En concreto se enseña cómo jugar, se detalla qué hacer en las sesiones informativas: una parte importante del juego, y las variaciones que se pueden añadir en cada partida aclarando cuales van a ser implementadas en este proyecto.

**Capítulo 3 Elicitación de requisitos y análisis:** Se plasman las fases de elicitación de requisitos y análisis de la aplicación. Incluye la descripción de los actores y roles, la elicitación de requisitos, la especificación de los casos de uso, el modelado del dominio y el modelo de flujo del sistema.

**Capítulo 4 Plan de proyecto:** Se incluye el documento de planificación del proyecto donde se describen las tareas a realizar con sus fechas en el calendario, la metodología utilizada, el propósito y alcance del proceso de desarrollo, la gestión de riesgos y el presupuesto. Además en esta sección se incluye la tabla con la explicación de los acrónimos utilizados durante la redacción de la memoria.

**Capítulo 5 Tecnologías utilizadas:** Se detallan las tecnologías que se han utilizado en todas las fases del proyecto, es decir en la gestión, análisis y diseño del proyecto así como en implementación de la página web.

**Capítulo 6 Diseño:** Se especifican la decisiones que se han tomado durante el diseño de la aplicación, se muestra los bocetos de las interfaces de usuario, se detallan los patrones arquitectónicos y de diseño utilizados y por último, se muestra la arquitectura del sistema, tanto la lógica como la de despliegue.

**Capítulo 7 Implementación y pruebas:** En este capítulo se describen los detalles de la implementación y los casos de prueba realizados así como los resultados de estos.

**Capítulo 8 Seguimiento del proyecto:** Se especifican los riesgos sucedidos, cómo han afectado a la planificación del proyecto y el tiempo dedicado a cada tarea planificada.

**Capítulo 9 Conclusiones y trabajo futuro:** Se valoran los resultados obtenidos, se detallan las posibles líneas de trabajo futuras y se aporta una conclusión personal.

**Anexo A Manuales:** Incluye el manual de usuario y de despliegue de la aplicación.

**Anexo B Resumen de enlaces adicionales:** Incluye enlaces de interés sobre el proyecto, como el repositorio de código.

# Capítulo 2

# El juego: Hard Choices

*The Hard Choices Game* se podría traducir como *El Juego de las Decisiones Difíciles*, donde se simula el ciclo de vida de un proyecto y se compite por terminarlo y ser el líder del mercado. Los jugadores deberán decidir entre invertir en esfuerzo o tomar atajos, ambas opciones con sus consecuencias positivas y negativas debido a la incertidumbre a la que se exponen todos los participantes.

Los creadores de este juego: Nanette Brown, Philippe Kruchten, Erin Lim, Robert L. Nord e Ipek Ozkaya, afirman que el uso de las metodologías ágiles tiene el peligro de que los desarrolladores nos centremos en la "piel" del sistema y que llegamos a ignorar algunos problemas de la arquitectura del sistema, debido a que uno de los principales pilares del desarrollo ágil es la entrega temprana del software para que los clientes finales vean el progreso [16],

Esto les sirvió de inspiración para crear este juego educativo el cual tiene como objetivo principal brindar a los participantes una mejor compresión de las estrategias que emplean durante el desarrollo software y advertirles de si la deuda técnica de los proyectos crece notablemente estos pueden llegar a colapsar y convertirse en inviables.

A continuación se explica cómo jugar: informando de la configuración del juego, las reglas, cómo obtener puntos y cómo finalizar las rondas. Además, se verá qué es y cómo guiar una sesión informativa y, por último, las posibles variaciones que se pueden introducir en el juego.

## 2.1. Cómo jugar

Pueden jugar entre dos y cuatro personas. La duración aproximada del juego es de 1 hora, al final del mismo se incluye una conversación informativa para que los jugadores pongan

en común sus experiencias y estrategias y discutan pros y contras de las mismas. Gana el jugador con más puntos al final de la partida.

### 2.1.1. ¿Qué representa un jugador?

Un jugador representa al jefe del proyecto software que se ha de lanzar al mercado. Por tanto, es el encargado de tomar todas las decisiones, desde el inicio del proyecto hasta el final, y es el responsable de las consecuencias de las mismas.

Todos los jugadores compiten por ser los primeros en llegar a la misma meta es decir, por ser los primeros en lanzar el mismo producto al mercado. Por ende, cada jugador representa un jefe de proyecto para un mismo producto pero en una empresa diferente.

### 2.1.2. Configuración del juego

El juego está constituido por:

- Un tablero, el cual representa las actividades implicadas en el desarrollo software.
- Casilla de salida de donde deben partir todos los jugadores.
- Casilla de meta a donde deben llegar todos los jugadores.
- Casillas de punto, representadas por una herramienta y equivalentes a la recompensa por invertir en la infraestructura técnica. Cada casilla de punto solo aporta puntos una vez por ronda.
- Casilla de "decisión difícil", marcada con 3 flechas en rojo, en esta casilla el jugador deberá decidir si tomar un atajo o continuar por el camino largo.
- Puentes, representan los atajos que se pueden tomar para no ser los últimos en lanzar el producto al mercado.
- Fichas, que representan a cada jugador y que sirven para ver en que posición del tablero se encuentran.
- Cartas que al final de la partida se traducen en puntos de penalizaciones y de recompensas. Las cartas de puente son las cartas de penalización y las de herramienta cuentan como puntos de recompensa.

### 2.1.3. Reglas del juego

El objetivo del juego es conseguir puntos, esto se logra aterrizando en una casilla de punto y no siendo el último en llegar a la meta. El juego puede consistir en más de una ronda

pero el moderador no ha de avisar de esto. Se recomienda que al menos haya dos rondas. Todos los jugadores empiezan sin ninguna carta en mano, es decir 0 puntos en sus marcadores.

Para determinar quien empieza, cada jugador lanza el dado y el que obtenga el número más alto comienza y se continua en el sentido de las agujas del reloj.

En cada turno el jugador lanza el dado y se mueve tantas casillas como el dado indique:

- Menos el número de cartas puente, penalizaciones, que tenga en su poder.
- Cada jugador puede decidir en que dirección del tablero moverse, incluso hacia atrás. Ejemplo: Si el dado indica un 5 puede moverse 2 casillas hacia atrás y 3 hacia delante.
- Una vez que el primer jugador ha cruzado la casilla de fin el resto de jugadores solo pueden moverse hacia delante.
- Para ingresar en la casilla de fin se ha de tirar el dado y obtener un número igual o superior a las casillas restantes.
- El juego se termina cuando solo queda 1 jugador en el tablero. Este deberá detenerse y no podrá obtener más turnos.

Gana el jugador con más puntos en mano.

### 2.1.4. Obtención de puntos

Se obtienen puntos por llegar a la casilla de fin: el primero jugador obtendrá 7 puntos, el segundo 3 y el tercero 1. También se obtendrán por cada carta de herramienta que el jugador tenga en su poder, es decir por cada vez que haya aterrizado en una casilla de herramienta. El jugador que quede en tablero solo se le contarán los puntos que tenga en mano.

### 2.1.5. Fin de la ronda

Cuando todos los jugadores hayan terminado la 1º ronda el moderador debe anunciar la 2º ronda, que cada jugador iniciará con los mismos puntos que se terminó la anterior y así para las siguientes.

Todos los jugadores deben volver a empezar desde la casilla de inicio y las normas son siempre las mismas.

El moderador puede detener las rondas si considera que se ha entendido cómo se acumula la deuda técnica, del mismo modo puede ir añadiendo rondas para que los jugadores experimenten todos los conocimientos subyacentes del juego.

Antes del inicio de cada ronda, a excepción de la primera, habrá un tiempo para que los jugadores ideen la estrategia, además el moderador podrá añadir cualquiera de las siguientes modificaciones:

- **Martillo**: Por cada carta de martillo pueden cruzar puentes sin penalización siempre y cuando retiren una carta martillo de su mano.
- **Sierra**: Por cada carta de sierra que tiene un jugador, suma 1 a cada tirada del dado. No ha de devolver la carta de su mano.
- **Destornillador**: Por cada carta de destornillador que tiene un jugador, reste 1 de cada tirada. No ha de devolver la carta de su mano.
- **Puente**: Por cada carta de destornillador se devuelve una carta puente.
- **Puntos**: Cada carta de herramienta ahora vale un número mayor o menor de puntos; por ejemplo, las cartas de martillo ahora valen 5 puntos cada una, a decisión del moderador.

Las cartas que otorgan puntos: Martillo, Sierra y Destornillador, representan la inversión en esfuerzo por crear código de calidad, por ejemplo: por invertir tiempo en el diseño de la arquitectura del mismo, por una buena elicitación de requisitos, por gastar tiempo en buscar las tecnologías más adecuadas para la implementación del código, etc. Sin embargo, las casillas de puente representan todo lo contrario: por ganar tiempo no se invierte en buenas prácticas y esto hace que aumentemos las penalizaciones, es decir, la deuda técnica.

Por eso, al final del juego se hace un recuento de las cartas de cada jugador que representa que aunque tengamos un poco de deuda técnica, cartas puente, se pude solventar refactorizando, con cartas de herramientas, y así obtener un código de calidad, es decir un marcador que no sea negativo.

## 2.2. Sesión informativa

La sesión informativa se puede realizar al final del juego o al final de cada ronda. Se recomienda esto último para ayudar a los jugadores a pensar sobre su estrategia en la ronda anterior y cómo mejorarla. De este modo se afianzan los conocimientos poco a poco y es más fácil saber si están aprendiendo algo a lo largo de la partida y cuándo podemos parar el juego.

El moderador guiará la discusión preguntando:

- ¿Qué acaba de suceder? Reunir datos.
- ¿Por qué ha podido suceder? Generar conocimiento.
- ¿Y ahora qué? Pensar qué hacer diferente para estrategias futuras.

### 2.2.1. Posibles reflexiones

En esta parte de la partida se ha de ver la relación del juego con la deuda técnica. Por eso no hay que perder de vista que el tablero simula el recorrido que hace un proyecto desde su inicio hasta su lanzamiento al mercado y que cada jugador compite por sacar el mismo producto el primero. Ahora bien, ¿es buena estrategia escoger todos los atajos para lograrlo?, ¿es preferible sacar tu producto en segundo o tercer lugar y haber arreglado todos los fallos de tus prototipos?

Que tu producto sea el primero en el mercado tiene claras ventaja, entre ellas: la novedad y la exclusividad, por eso el juego bonifica con puntos extras a los jugadores que llegan a meta, pero a veces ser el primero implica no sacar un producto de buena calidad y, por tanto, tener un código inmaduro, inflexible, con fallos, etc. Por eso, cada vez que coges un atajo para adelantar a tus competidores el juego te penaliza.

Sin embargo, recoger todas las herramientas del tablero también puede ser mala idea. Si decides ir por el camino largo y recoger todos los puntos puede que no llegues a meta, es decir, que tu producto no llegue a salir al mercado, mientras que el resto de jugadores tendrán una primera versión del producto software. Esto hace referencia a la refactorización del código para reducir la deuda técnica, como se ha dicho en la sección de introducción: 1.1, un poco de deuda técnica no está mal, siempre y cuando en un futuro se aplique refactorización al código.

Para representar la refactorización el juego obliga a que haya mínimo dos rondas por partida y así poder enmendar las malas decisiones. Pero ¿cuántas más ha de haber?, ¿cuánto tiempo se puede dedicar a refactorizar el código de un proyecto para ser el mejor del mercado?

Otra incógnita que el juego intenta que veamos es: ¿se puede tener una aplicación con malas prácticas y dedicar las siguientes versiones a la refactorización? o lo que sería los mismo ¿es buena estrategia llegar a la meta el primero, pasar por todos los puentes y en las siguientes rondas solo recoger herramientas? Para esto hay que tener en cuenta que un juego es competitivo y que el resto de jugadores, hayan llegado a meta o no, seguirán en las siguientes rondas.

Estas son algunas de las preguntas que se pueden exponer en la sesión informativa para que los jugadores vean la similitud del juego con la deuda técnica y reflexionen sobre ella. En cada partida pueden salir otras distintas.

## 2.3. Variaciones del juego

Las variaciones a menudo surgen en la sesión informativa y se aplican para darle más realismo a la analogía del desarrollo software. Las variaciones que se pueden añadir al juego se explican a continuación y su adaptación para el formato web en la sección 3.4.4 de requisitos.

### 2.3.1. Movimiento

No se puede cambiar de dirección una vez escogido el sentido del movimiento en cada tirada. Esto dificulta que los jugadores aterricen en casillas de puntos.

Además, el tablero puede incluir puertas o muros que impidan desplazarse hacia atrás una vez pasados. Esto hace que vivan con las consecuencias de las decisiones tempranas.

### 2.3.2. Penalizaciones

Diseñar estrategias para la devolución de sanciones, por ejemplo solo al inicio de la ronda en lugar de en cualquier momento.

### 2.3.3. Decisiones técnicas

Cambiar las casillas de puntos por casillas de decisiones técnicas. Estas decisiones deben ser más realistas y los jugadores deben leerlas y decidir si tomarla o no. Por ejemplo: “diseñar una base de datos”.

### 2.3.4. Juego en equipo

En lugar de competir individualmente los jugadores de cada tablero podrán competir contra los de otro tablero con el objetivo de ayudar a sus compañeros para que todos lleguen a la casilla de fin antes que el otro equipo.

Esta variación no será incluida en esta versión a desarrollar como parte del trabajo de fin de grado.

# Capítulo 3

# Elicitación de Requisitos y Análisis

A continuación se detallan a quién va dirigida la aplicación los roles en la misma, sus requisitos, tanto funcionales, de información como no funcionales y los casos de uso que se van a desarrollar en este proyecto. Por último se mostrará el modelo de dominio de la aplicación.

## 3.1. Modificaciones respecto al juego original

La aplicación web disponible al finalizar este proyecto pretende ajustarse lo máximo posible al juego original. No obstante, es necesario hacer unos pequeños cambios respecto a lo dicho en el Capítulo 2, donde se explica el juego original.

La primera modificación se verá al inicio del juego ya que, en lugar de pedir a los jugadores que lancen el dado para ver quién empieza, el sistema seleccionará aleatoriamente al jugador que debe iniciar la partida.

Se ha dicho que por cada casilla de herramienta en la que el jugador caiga o por cada puente que cruce ha de coger una carta de punto o de penalización, respectivamente. Estas cartas se sustituirían por contadores de puntos por tanto, cada jugador verá en su pantalla, mínimo, dos marcadores: marcador de puntos y marcador de penalizaciones. Si se añadiese alguna de las variaciones descritas anteriormente, el marcador de puntos se desdoblará para que se visualice un marcador por tipo de herramienta.

En el juego original es el moderador el encargado de monitorizar la partida, como este sistema está pensado para que se use en un entorno educativo donde el profesor actúa como moderador sobre más de una partida simultánea se ha decidido que el caso de uso "monitorizar partida" pueda ser realizado por cualquier usuario del sistema.

## 3.2. Descripción de los actores y roles de la aplicación

Para jugar al juego original *Hard Choices* se necesitan 2 roles, el de jugador y el moderador, los cuales, en este sistema, tienen funcionalidades comunes por lo que heredarán de un mismo rol "Usuario". Además de estos la aplicación incluirá un rol de profesor que actuará como moderador y administrador de su perfil.

Más en detalle los roles son:

- **Moderador**. Es el encargado de crear la partida y de que al finalizar la misma los demás hayan aprendido sobre la deuda técnica. Además podrá parar, reanudar y finalizar la partida cuando considere necesario.
- **Profesor**. Además de las acciones del moderador puede crear salas privadas para agrupar las partidas que va creando. Tiene una cuenta donde podrá ver dichas salas, los rankings que ha creado y su alias. Podrá recuperar su contraseña a través de un correo electrónico en caso de que se olvide de ella.
- **Jugador**. El participante que desarrolla la partida del juego, puede ser un alumno de un centro educativo, y es el que ha de adquirir conocimiento sobre la deuda técnica. Una vez finalizada la partida podrá unirse a uno o varios ranking que él mismo seleccione.
- **Usuario**. Engloba las funcionalidades comunes de los roles jugador y el moderador. En concreto, monitorizar las discusiones finales de cada ronda, añadir las variaciones y rondas y ver los ranking creados en el sistema.

## 3.3. Especificación de requisitos

A continuación se especifican los requisitos funcionales, no funcionales y de información de la aplicación.

### 3.3.1. Requisitos funcionales

A continuación, en la Tabla 3.1 se detallan los requisitos funcionales del sistema.

| ID | Descripción | Prioridad |
|---|---|---|
| RF01 | El sistema permitirá crear partidas de 2 a 4 jugadores. | Alta |
| RF02 | El sistema permitirá crear partidas con 2 o más rondas. | Alta |
| RF03 | El sistema permitirá incluir un enlace a una plataforma de servicio de mensajería instantánea. | Alta |

| | | |
|---|---|---|
| RF04 | El sistema permitirá añadir variaciones al inicio de cada ronda a excepción de la primera. | Alta |
| RF05 | El sistema permitirá al moderador añadir más rondas de las iniciales a la partida. | Alta |
| RF06 | El sistema permitirá al usuario moderador iniciar una partida previamente creada. | Alta |
| RF07 | El sistema permitirá al usuario identificarse con un alias al inicio de cada partida. | Alta |
| RF08 | El sistema permitirá al moderador pausar la partida. | Alta |
| RF09 | El sistema permitirá al moderador reanudar la partida. | Alta |
| RF10 | El sistema permitirá al moderador terminar la partida manualmente sin necesidad de que los jugadores terminen todas las rondas. | Alta |
| RF11 | El sistema permitirá al usuario jugador jugar. | Alta |
| RF12 | El sistema permitirá al usuario jugador salir de la partida. | Alta |
| RF13 | El sistema permitirá al usuario registrarse con el rol de profesor. | Media |
| RF14 | El sistema permitirá al usuario profesor iniciar sesión. | Media |
| RF15 | El sistema permitirá al usuario profesor crear salas en su perfil. | Media |
| RF16 | El sistema permitirá al usuario profesor entrar en una sala previamente creada en su perfil. | Media |
| RF17 | El sistema permitirá al usuario profesor restablecer la contraseña de su perfil. | Media |
| RF18 | El sistema permitirá al usuario profesor cerrar sesión. | Media |
| RF19 | El sistema permitirá a los moderadores crear nuevos ranking. | Media |
| RF20 | El sistema permitirá a los jugadores escoger en qué ranking quieren aparecer además de en el global. | Media |
| RF21 | El sistema permitirá a los usuarios ver los pares alias jugador - puntuación de un ranking seleccionado previamente | Media |
| RF22 | El sistema permitirá al usuario profesor cerrar sesión su sesión | Baja |
| RF23 | El sistema permitirá al usuario profesor cambiar la contraseña de sus sesión | Baja |
| RF24 | El sistema permitirá al usuario profesor eliminar salas que tenga en su perfil. | Baja |
| RF25 | El sistema permitirá al usuario profesor eliminar partidas que haya creado en una sala. | Baja |

**Tabla 3.1. Requisitos funcionales**

Los requisitos RF01 y RF02 se consideran reglas de negocio.

### 3.3.2. Requisitos de información

En la tabla 3.2 se detallan los requisitos de información del sistema.

| ID | Descripción | Prioridad |
|---|---|---|
| RI01 | El sistema deberá almacenar información sobre el progreso de las partidas durante lo que dure la misma | Alta |
| RI02 | El sistema deberá almacenar información sobre el perfil del profesor: correo electrónico, contraseña, alias, salas creadas, partidas creadas, la clave de cada sala creada | Media |
| RI03 | El sistema deberá almacenar información sobre las salas creadas por los profesores: nombre, identificador, las partidas que pertenecen a la sala y, de tenerlo, el enlace al canal de voz | Media |
| RI04 | El sistema deberá almacenar información sobre las partidas creadas por los profesores: identificador y configuración | Media |
| RI05 | El sistema deberá almacenar información sobre el ranking: nombre del ranking, alias y puntuación de los jugadores que lo haya seleccionado ordenados por puntación | Baja |
| RI06 | El sistema deberá almacenar información sobre el perfil del profesor: los rankings creados | Baja |

**Tabla 3.2. Requisitos de información**

### 3.3.3. Requisitos no funcionales

La elicitación de los requisitos no funcionales está basada en el acrónimo [F]URPS+: [15]

- **Functionality** (Funcionalidad): Requisitos que expresan la funcionalidad el sistema, descritos en el apartado 3.3.1 de requisitos funcionales.
- **Usability** (Usabilidad): Requisitos que tienen que ver con las características que ha de tener la interfaz de usuario.
- **Reliability** (Fiabilidad): Requisitos que expresas la disponibilidad del sistema y su recuperación ante fallos.
- **Performance** (Rendimiento): Requisitos que determinan el tiempo de respuesta del sistema.
- **Support** (Soporte): Requisitos sobre el mantenimiento, compatibilidad, instalación, configuración, etc del sistema.

- +: Requisitos adicionales que tiene que ver con privacidad, hardware, implementación, etc.

En la tabla 3.3 se detallan los requisitos de información del sistema.

| ID | Descripción | Prioridad |
|---|---|---|
| RNF01 | El sistema deberá estar disponible desde un navegador web. | Alta |
| RNF02 | El sistema deberá estar disponible en español. | Alta |
| RNF03 | El sistema deberá tener un tiempo máximo de respuesta de 5 segundos. | Alta |
| RNF04 | El sistema deberá tener una interfaz de usuario adaptable de dispositivos móviles, tablets y ordenadores. | Alta |
| RNF05 | El sistema deberá ser capaz de utilizar el formato UTF-8. | Alta |
| RNF06 | El sistema no requerirá ningún tipo de de instalación a excepción del navegador. | Alta |
| RNF07 | El sistema se compromete a no exportar datos de los usuarios a terceros. | Alta |
| RNF08 | El sistema deberá ser capaz de funcionar en varios navegadores: Google Chorme, Safari y Mozilla Firefox. | Media |
| RNF09 | El sistema deberá asegurar que el 90 % de los usuarios sabrán usar el 90 % de la funcionalidad del sistema pasados 10 minutos. | Media |
| RNF10 | El sistema deberá tener opción de traducción. | Baja |
| RNF11 | El sistema garantiza que el sistema no estará más de un mes y medio al año, de forma discontinua, sin funcionar. | Baja |

**Tabla 3.3. Requisitos no funcionales**

### 3.3.4. Casos de uso

En la tabla 3.4 se detallan los casos de uso del sistema, el diagrama se puede ver en la figura 3.2 en la sección 3.4.1 *Modelo de dominio*.

| Rol | ID | Nombre |
|---|---|---|
| Usuario | CU01 | Monitorizar partida |
| Usuario | CU02 | Añadir ronda |
| Usuario | CU03 | Añadir variación |
| Usuario | CU04 | Ver ranking |
| Usuario | CU05 | Buscar ranking |
| Jugador | CU06 | Realizar jugada |
| Jugador | CU07 | Unirse a la partida |

| | | |
|---|---|---|
| Jugador | CU08 | Salir de la partida |
| Jugador | CU09 | Unirse a un ranking |
| Moderador | CU10 | Pausar partida |
| Moderador | CU11 | Reanudar partida |
| Moderador | CU12 | Iniciar partida |
| Moderador | CU13 | Terminar partida |
| Moderador | CU14 | Crear partida |
| Moderador | CU15 | Crear ranking |
| Profesor | CU16 | Crear sala |
| Profesor | CU17 | Registrarse |
| Profesor | CU18 | Entrar en una sala creada |
| Profesor | CU19 | Buscar sala |
| Profesor | CU20 | Eliminar sala |
| Profesor | CU21 | Iniciar sesión |
| Profesor | CU22 | Eliminar partida |
| Profesor | CU23 | Cerrar sesión |
| Profesor | CU24 | Recuperar contraseña |

**Tabla 3.4. Casos de uso**

### Matriz de correspondencia

Una vez expuestos todos los requisitos y casos de uso del sistema y para asegurar que todos los requisitos funcionales están cubiertos se ha creado la matriz de correspondencia que se muestra a continuación, en la tabla 3.1

### Descripción de los casos de uso

A continuación, de la tabla 3.5 a la tabla 3.28, se especifican los casos de uso del sistema indicados en la tabla 3.4 de la sección 3.3.4 *Casos de uso*.

| **CU01** | **Monitorizar partida** |
|---|---|
| Definición | Finalizada la ronda se para el juego y se muestra una ayuda para que el moderador facilite la sesión informativa.<br>Este caso de uso se refiere a la sección 2.2 |
| Actor | Usuario |
| Precondición | Se ha de haber finalizado una ronda. |
| Postcondición | |

| | |
|---|---|
| Secuencia normal | 1. El actor indica que quiere moderar la discusión.<br>2. El sistema muestra información para ayudar al moderador a facilitar la sesión informativa.<br>3. El actor indica que quiere finalizar la moderación.<br>4. El sistema pregunta si quiere añadir variaciones a la partida.<br>5. El actor acepta.<br>6. Se ejecuta el caso de uso “Añadir variaciones”.<br>7. El sistema pregunta si quiere añadir rondas a la partida.<br>8. El actor acepta.<br>9. Se ejecuta el caso de uso “Añadir ronda”<br>11. Se vuelve al tablero de juego y comienza la siguiente ronda. |
| Excepciones | 5.b. El actor rechaza y el caso uso continúa por el paso 7.<br>8.b. El actor rechaza y el caso uso finaliza.<br>11.b. No quedan rondas y se ejecuta el caso de uso “Terminar partida” |
| Comentarios | |

**Tabla 3.5. Descripción del caso de uso CU01, Monitorizar partida**

| **CU02** | **Añadir ronda** |
|---|---|
| Definición | El usuario puede aumentar el número de rondas máximas de la partida en curso |
| Actor | Usuario |
| Precondición | Se ha ejecutado el caso de uso “Moderar discusión” |
| Postcondición | Se ha añadido una 1 o más rondas a la partida en curso |
| Secuencia normal | 1. El actor selecciona que quiere añadir más rondas.<br>2. El sistema pide al usuario que introduzca el número de rondas que quiere añadir.<br>3. El actor indica el número de rondas que quiere añadir<br>4. El sistema pide confirmación.<br>5. El actor acepta.<br>6. El sistema añade el número de rondas indicadas por el actor a la configuración de la partida. |
| Excepciones | 5.b El actor cancela y se termina el caso de uso sin efecto. |
| Comentarios | |

**Tabla 3.6. Descripción del caso de uso CU02, Añadir ronda**

| **CU03** | **Añadir variación** |
|---|---|
| Definición | El usuario puede añadir variaciones a la partida en curso |
| Actor | Usuario |
| Precondición | Se ha ejecutado el caso de uso “Moderar discusión” |
| Postcondición | Se ha añadido una 1 o más variaciones a la partida en curso |

| | |
|---|---|
| Secuencia normal | 1. El actor selecciona que quiere añadir variaciones (Sección 2.1.4. y Sección 2.3).<br>2. El sistema pide al usuario que indique que variación/es quiere añadir.<br>3. El actor indica que variación/es quiere añadir.<br>4. El sistema pide confirmación.<br>5. El actor pulsa aceptar.<br>6. El sistema añade la/s variación/es indicadas por el actor a la partida. |
| Excepciones | 5.b El actor cancela y se termina el caso de uso sin efecto. |
| Comentarios | |

**Tabla 3.7. Descripción del caso de uso CU03, Añadir variación**

| **CU04** | **Ver ranking** |
|---|---|
| Definición | El actor podrá ver los jugadores y su puntuación que se hayan unido al ranking previamente |
| Actor | Usuario |
| Precondición | |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere ver un ranking.<br>2. Se ejecuta le caso de uso "Buscar ranking".<br>3. El sistema muestra los pares jugador-puntuación que se hayan unido al ranking previamente. |
| Excepciones | |
| Comentarios | |

**Tabla 3.8. Descripción del caso de uso CU04, Ver ranking**

| **CU05** | **Buscar ranking** |
|---|---|
| Definición | |
| Actor | Usuario |
| Precondición | Se ha de haber iniciado el caso de uso "Ver ranking" o el caso de uso "Unirse a un ranking" |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere buscar un ranking.<br>2. El sistema muestra los rankings creados en el sistema.<br>3. El actor selecciona un ranking |
| Excepciones | 3.b El actor cancela y el caso de uso finaliza sin efecto. |
| Comentarios | |

**Tabla 3.9. Descripción del caso de uso CU05, Buscar ranking**

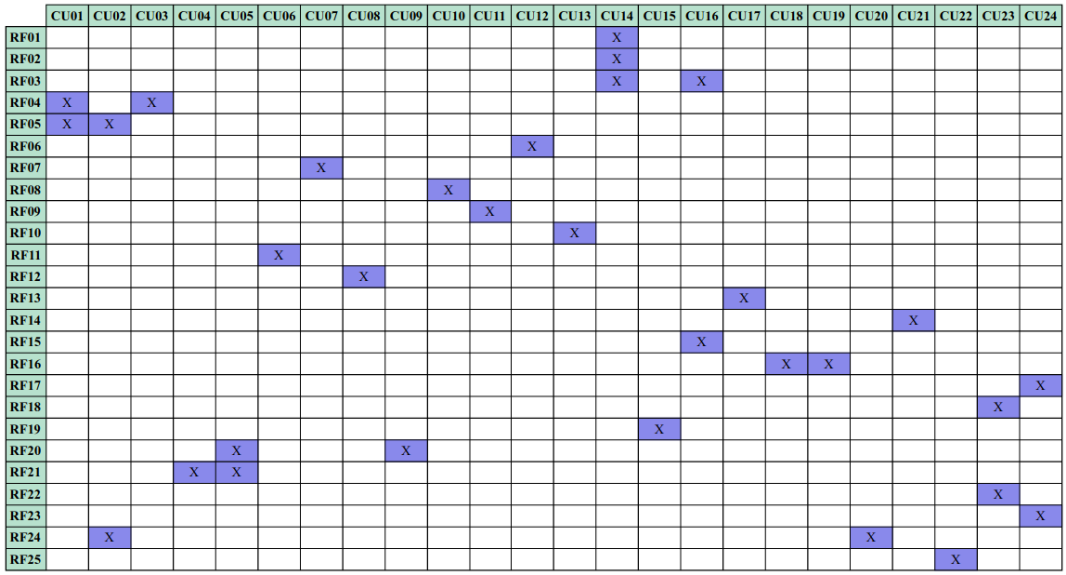


**Figura 3.1. Matriz de correspondencia (RF/CU)**

| **CU06** | **Realizar jugada** |
|---|---|
| Definición | El usuario jugador juega su turno en la ronda correspondiente. |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Iniciar partida” |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere iniciar la jugada.<br>2. El sistema pide que lance el dado.<br>3. El actor lanza el dado.<br>4. El sistema calcula el número de casillas que puede desplazarse. (Sección 2.1.3)<br>5. El sistema pide que indique en que dirección quiere moverse. (Sección 2.1.3)<br>6. El actor indica en que dirección quiere moverse.<br>7. El sistema mueve la ficha del jugador las casillas correspondientes.<br>8.El sistema acumula las penalizaciones de los puentes por los que ha pasado la ficha del actor.<br>9. Si el número de casillas desplazadas en total ha sido menor que el calculado en el paso 4 el caso de uso continua por el paso 5. (Sección 2.1.3)<br>10. El sistema acumula los puntos de la casilla en la que ha finalizado el movimiento.(Sección 2.1.3)<br>11. El sistema genera la señal del próximo jugador. |

| | |
|---|---|
| Excepciones | 4.b. Puede haber modificaciones (sección 2.1.5)<br>9.b. Si la variante movimiento está activada no se genera el bucle.<br>11.b. Si el jugador ha llegado meta y en el tablero solo queda un jugador se genera la señal de fin de ronda y la de próximo jugador |
| Comentarios | |

**Tabla 3.10. Descripción del caso de uso CU06, Realizar jugada**

| **CU07** | **Unirse a la partida** |
|---|---|
| Definición | Los jugadores pueden unirse a las partidas creadas |
| Actor | Jugador |
| Precondición | Se ha de haber realizado el caso de uso "Crear partida" |
| Postcondición | La partida tiene un jugador más asociado. |
| Secuencia normal | 1. El actor indica que quiere unirse a la partida.<br>2. El sistema comprueba los datos de acceso.<br>3. El sistema pide que introduzca un alias público.<br>4. El actor introduce el alias.<br>5. El sistema asocia el nuevo jugador a la partida.<br>6. El sistema muestra la ventana de "espera a que se inicie la partida" |
| Excepciones | |
| Comentarios | En este caso los datos de acceso será la id de la partida que tendrán que introducir en un *input*. |

**Tabla 3.11. Descripción del caso de uso CU07, Unirse a la partida**

| **CU08** | **Salir de la partida** |
|---|---|
| Definición | |
| Actor | Jugador |
| Precondición | Se ha de haber realizado el caso de uso "Terminar partida" |
| Postcondición | Se crea una nueva cuenta con al menos una sala y una partida. |
| Secuencia normal | 1. El sistema calcula la puntuación del jugador.<br>2. El sistema pregunta si quiere unirse a un ranking.<br>3. Se ejecuta el caso de uso "Unirse a un ranking" |
| Excepciones | 3.b El actor indica que no quiere unirse a un ranking, el caso de uso finaliza y se muestra la pantalla de inicio. |
| Comentarios | |

**Tabla 3.12. Descripción del caso de uso CU08, Salir de la partida**

| **CU09** | **Unirse a un ranking** |
|---|---|
| Definición | Una vez finalizada la partida se le propone al jugador unirse a un ranking previamente creado. |
| Actor | Jugador |

| | |
|---|---|
| Precondición | Se ha de haber realizado el caso de uso “Terminar partida” |
| Postcondición | La puntuación y el alias del jugador quedan guardados en el/los ranking/s que seleccione. |
| Secuencia normal | 1. El actor indica que quiere unirse a un ranking.<br>2. Se ejecuta le caso de uso ”Buscar ranking”.<br>3. El sistema pide confirmación.<br>4. El actor confirma.<br>5. El sistema genera la puntuación y la guarda junto el alias del jugador en el ranking seleccionado.<br>6. El sistema pregunta si quiere unirse a otro ranking.<br>7. El actor dice que no.<br>8. El sistema muestra la ventana de inicio. |
| Excepciones | 2.b. El caso de uso ”Buscar ranking” finaliza sin efecto. El caso de uso actual continúa en el paso 8.<br>4.b. El actor cancela. El caso de uso vuelve al paso 2.<br>7.b. El actor indica que si. El caso de uso vuelve al paso 2. |
| Comentarios | |

**Tabla 3.13. Descripción del caso de uso CU09, Unirse a un ranking**

| **CU10** | **Pausar partida** |
|---|---|
| Definición | El actor moderador en cualquier momento de la ronda puede pausar la partida. |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Iniciar partida” y no se ha de estar en la ronda informativa, pues en ese caso el juego ya está parado. |
| Postcondición | El juego se pausa impidiendo que los actores “jugador” efectúen acciones y guardando el estado. |
| Secuencia normal | 1. El actor indica que quiere pausar la partida.<br>2. El sistema muestra un mensaje de pausa , congela las pantallas de los actores que no son moderador y guarda el estado de la partida. |
| Excepciones | |
| Comentarios | |

**Tabla 3.14. Descripción del caso de uso CU10, Pausar partida**

| **CU11** | **Reanudar partida** |
|---|---|
| Definición | |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Pausar partida” |
| Postcondición | Se reanuda la partida en el estado en el que se había parado. |
| Secuencia normal | 1. El actor indica que quiere reanudar una partida parada.<br>2. El sistema descongela las pantallas manteniendo el estado de en el que se había pausado la partida. |

| | |
|---|---|
| Excepciones | |
| Comentarios | |

**Tabla 3.15. Descripción del caso de uso CU11, Reanudar partida**

| **CU12** | **Iniciar partida** |
|---|---|
| Definición | El moderado inicia la partida cuando los jugadores se hayan unido a la partida |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Crear partida”.<br>Entre 2 y 4 actores “jugador” han de haber realizado el caso de uso “Unirse a la partida” |
| Postcondición | La partida queda iniciada. |
| Secuencia normal | 1. El actor indica que quiere iniciar la partida.<br>2. El sistema comprueba que haya el mínimo de jugadores y si es la primera ronda, si lo es, determina el primer jugador y le indica que puede empezar. |
| Excepciones | 2.b No hay el mínimo de jugadores unidos a la partida. EL sistema muestra un mensaje de error y regresa al paso 1. |
| Comentarios | |

**Tabla 3.16. Descripción del caso de uso CU12, Iniciar partida**

| **CU13** | **Terminar partida** |
|---|---|
| Definición | Se puede terminar una partida porque ya no queden más rondas, porque se haya pasado el tiempo o porque el actor moderador así lo quiere.<br>En todos los casos se guarda la puntación hasta el momento. |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Iniciar partida” |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere finalizar la partida.<br>2. El sistema inhabilita las pantallas de los actores jugadores y calcula la puntuación obtenida hasta el momento.<br>3. El sistema pregunta al moderador si quiere crear un ranking.<br>4. El moderador indica que si.<br>5. Se realiza el caso de uso “Crear ranking”<br>6. Se ejecuta el caso de uso “Salir de la partida”<br>7. El sistema pregunta si quieren jugar otra partida.<br>8. El actor indica que no.<br>9. Se muestra la pantalla de inicio. |
| Excepciones | 4.b. El moderador indica que no. El caso de uso continua por el paso 6.<br>7.b El actor indican que si y se ejecuta el caso de uso ”Iniciar Partida” |

| | |
|---|---|
| Comentarios | |

**Tabla 3.17. Descripción del caso de uso CU13, Terminar partida**

| **CU14** | **Crear partida** |
|---|---|
| Definición | El actor podrá crear la partida añadiendo la configuración que desee. |
| Actor | Moderador |
| Precondición | Si el actor es profesor ha de haber realizado el caso de uso "Crear sala" o "Entrar en una sala creada". |
| Postcondición | Se ha creado una nueva partida con la configuración indicada por el actor. |
| Secuencia normal | 1. El actor selecciona que quiere crear una nueva partida.<br>2. El sistema muestra las opciones de configuración de la partida.<br>3. El actor introduce los datos de la configuración de la partida.<br>4. El sistema pide confirmación.<br>5. El actor acepta.<br>6. El sistema comprueba que los datos introducidos son correctos.<br>7. El sistema genera una partida con la configuración indicada, un enlace de unión a la misma y muestra la pantalla de inicio de partida. |
| Excepciones | 5.b. El actor cancela y el caso de uso queda sin efecto.<br>6.b. Los datos introducidos no cumplen con las restricciones del sistema. Se muestra un mensaje de error señalando los errores y se vuelve al paso 4. (Sección 2.1) |
| Comentarios | |

**Tabla 3.18. Descripción del caso de uso CU14, Crear partida**

| **CU15** | **Crear ranking** |
|---|---|
| Definición | El moderador podrá crear un ranking nuevo una vez se haya finalizado una partida |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso "Terminar partida" |
| Postcondición | Se crea una ranking público con el nombre dado por el actor. |
| Secuencia normal | 1. El actor indica que quiere crear un ranking.<br>2. El sistema pide que se introduzca un nombre.<br>3. El actor introduce un nombre.<br>4. El sistema pide confirmación.<br>5. El actor acepta.<br>6. El sistema comprueba que el nombre no exista.<br>7. El sistema crea y publica un nuevo ranking con el nombre dado por el actor. |
| Excepciones | 5.b. El actor cancela y el caso de uso queda sin efecto.<br>6.b El nombre introducido ya existe, se muestra un mensaje de error y se vuelve al paso 2. |
| Comentarios | |

**Tabla 3.19. Descripción del caso de uso CU15, Crear ranking**

| **CU16** | **Crear sala** |
|---|---|
| Definición | El actor crea una nueva sala en su perfil. |
| Actor | Profesor |
| Precondición | Se ha de haber realizado el caso de uso “Iniciar sesión” o “Registrarse” |
| Postcondición | Se crea una sala con al menos una partida en ella. |
| Secuencia normal | 1. El actor indica que desea crear una sala.<br>2. El sistema indica que introduzca un nombre para la sala.<br>3. El actor introduce un nombre.<br>4. El sistema comprueba que el nombre no exista en el perfil del actor.<br>5. Se realiza el caso de uso “Crear partida”<br>6. El sistema pregunta si quiere duplicar la partida anterior.<br>7. El actor indica que quiere duplicar la partida anterior.<br>8. El sistema pide que indique cuantas veces.<br>9. El actor introduce las veces que desea duplicar la partida.<br>10. El sistema comprueba que la entrada es correcta.<br>11. El actor acepta.<br>12. El sistema pregunta si quiere crear un ranking asociado a la sala.<br>13. Se realiza el caso de uso “Crear ranking”<br>14. El sistema crea la sala con el nombre dado, tantas salas como el actor haya indicado (al menos la del paso 5), un enlace de acceso a la sala y el ranking asociado. |
| Excepciones | 4.b. El nombre de la sala ya existe. Se muestra un mensaje de error y se vuelve al paso 2.<br>5.b. El caso “Crear partida” termina sin efecto. Se muestra un mensaje de error y el caso de uso termina sin efecto.<br>7.b. El actor indica que no quiere duplicar la sala creada. El caso de usa continua en el paso 11.<br>11.b El actor cancela y el caso de uso continua en el paso 6.<br>13.b. El caso de uso “Crear ranking” finaliza sin efecto. El caso de uso continua por el paso 15 sin asociar un ranking a la sala. |
| Comentarios | |

**Tabla 3.20. Descripción del caso de uso CU16, Crear sala**

| **CU17** | **Registrarse** |
|---|---|
| Definición | Un usuario crea una cuenta donde podrá gestionar salas y partidas. |
| Actor | Profesor |
| Precondición | |
| Postcondición | Se crea una nueva cuenta con al menos una sala y una partida. |

| | |
|---|---|
| Secuencia normal | 1. El actor indica que quiere registrarse.<br>2. El sistema pide que introduzca los datos.<br>3. El actor introduce los datos.<br>4. El sistema comprueba que los datos son correctos.<br>5. Se realiza el caso de uso "Crear Sala".<br>6. El sistema guarda los datos de la nueva cuenta.<br>7. El sistema muestra la pantalla de la cuenta creada. |
| Excepciones | 3.b. El actor cancela y el caso de uso finaliza sin ningún efecto.<br>4.b. Los datos introducidos son erróneos. Muestra un mensaje de error y que datos son correctos. Se vuelve al paso 2.<br>5.b. El caso de uso "Crear Sala" finaliza sin efecto. Se muestra un mensaje de error y el caso de uso finaliza sin efecto.<br>6.b. Se produce un error en mientras se guardan los datos. Se muestra un mensaje de error y el caso de uso finaliza sin efecto. |
| Comentarios | |

**Tabla 3.21. Descripción del caso de uso CU17, Registrarse**

| **CU18** | **Entrar en una sala creada** |
|---|---|
| Definición | El actor puede entrar en una sala previamente creada de su perfil. |
| Actor | Profesor |
| Precondición | Se ha de haber realizado el caso de uso "Iniciar sesión" |
| Postcondición | El actor entra en la sala indicada de su perfil |
| Secuencia normal | 1. El actor indica que quiere entrar en una sala.<br>2. El sistema pregunta si quiere buscar una sala.<br>4. Se ejecuta el caso de uso "Buscar sala".<br>5. El sistema muestra la pantalla donde se ve la información de la sala indicada. |
| Excepciones | 4.b El caso de uso "Buscar sala" finaliza sin efecto. El caso de uso actual finaliza sin efecto. |
| Comentarios | |

**Tabla 3.22. Descripción del caso de uso CU18, Entrar en una sala creada**

| **CU19** | **Buscar sala** |
|---|---|
| Definición | |
| Actor | Profesor |
| Precondición | Se ha de haber iniciado el caso de uso "Entrar en sala creada" o el caso de uso "Eliminar sala" |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere buscar una sala<br>2. El sistema muestra las salas que existen en el perfil del usuario.<br>3. El actor indica una sala.<br>4. El sistema selecciona la sala. |

| | |
|---|---|
| Excepciones | |
| Comentarios | |

**Tabla 3.23. Descripción del caso de uso CU19, Buscar sala**

| **CU20** | **Eliminar sala** |
|---|---|
| Definición | El actor puede eliminar una sla creada previamente de su perfil. |
| Actor | Profesor |
| Precondición | Se ha de haber realizado el caso de uso "Iniciar sesión" |
| Postcondición | Se elimina la sala indicada por el actor y todas las partidas creadas en ella. |
| Secuencia normal | 1. El actor indica que quiere eliminar una sala de su perfil.<br>2. El sistema pregunta si quiere buscar una sala.<br>3. Se ejecuta el caso de uso "Buscar sala".<br>4. El sistema comprueba que si elimina la sala queda al menos una sala en el perfil.<br>5. El sistema pide confirmación.<br>6. El actor acepta.<br>7. El sistema comprueba que no hay jugadores en ninguna partida de la sala.<br>8. El sistema elimina todas las partidas de la sala.<br>9. El sistema elimina la sala indicada por el actor. |
| Excepciones | 4.b.Sala indicada por el actor es la única en su perfil. El sistema muestra un mensaje de error y el caso de uso queda sin efecto.<br>6.b. El actor cancela y el caso de uso queda sin efecto.<br>7.b. El caso de uso "Eliminar partida" termina sin efecto. Se muestra mensaje de error y el caso de uso queda sin efecto. |
| Comentarios | |

**Tabla 3.24. Descripción del caso de uso CU20, Eliminar sala**

| **CU21** | **Iniciar sesión** |
|---|---|
| Definición | El actor puede iniciar sesión siempre y cuando se haya registrado antes. |
| Actor | Profesor |
| Precondición | |
| Postcondición | El actor entra en su cuenta. |
| Secuencia normal | 1. El actor indica que quiere iniciar sesión.<br>2. El sistema pide que introduzca la identificación.<br>3. El actor introduce la identificación.<br>4. El sistema comprueba que la identificación sea correcta.<br>5. El sistema muestra la pantalla de la cuenta del actor. |
| Excepciones | 3.b. El actor cancela y el caso de uso finaliza sin efecto.<br>4.b. La identificación no es correcta. Se muestra un mensaje de error y se continua en el paso 2. |

| | |
|---|---|
| Comentarios | |

**Tabla 3.25. Descripción del caso de uso CU21, Iniciar sesión**

| **CU22** | **Eliminar partida** |
|---|---|
| Definición | El actor puede eliminar una partida creada de una sala creada. |
| Actor | Profesor |
| Precondición | Se ha de haber realizado el caso de uso "Entrar en una sala creada". |
| Postcondición | Se elimina la partida indicada por el actor. |
| Secuencia normal | 1. El actor indica que desea eliminar una partida.<br>2. El sistema pregunta que partida de las de la sala quiere eliminar.<br>3. El actor selecciona una partida.<br>4. El sistema pide confirmación.<br>5. El actor acepta.<br>6. El sistema comprueba que no haya jugadores en la partida.<br>7. El sistema elimina la partida seleccionada de la sala. |
| Excepciones | 5.b. El actor cancela y el caso de uso termina sin efecto.<br>6.b. Hay jugadores en la partida, se muestra mensaje de erro y el caso de uso termina sin efecto. |
| Comentarios | |

**Tabla 3.26. Descripción del caso de uso CU22, Eliminar partida**

| **CU23** | **Cerrar sesión** |
|---|---|
| Definición | El usuario cierra la sesión abierta |
| Actor | Profesor |
| Precondición | Se ha ejecutado el caso de uso "Registrarse" o se ha ejecutado el caso de uso "Iniciar sesión" |
| Postcondición | Se ha cerrado sesión del usuario guardando las salas y partidas creadas |
| Secuencia normal | 1. El usuario selecciona que quiere cerrar sesión.<br>2. El sistema cierra sesión y vuelve a la página principal. |
| Excepciones | |
| Comentarios | |

**Tabla 3.27. Descripción del caso de uso CU23, Cerrar sesión**

| **CU24** | **Recuperar contraseña** |
|---|---|
| Definición | Un usuario identificado podrá crear una nueva contraseña para su perfil |
| Actor | Profesor |
| Precondición | Se ha de haber realizado con éxito el caso de uso "Registrarse" |
| Postcondición | El sistema elimina la antigua contraseña y guarda la introducida por el actor |

| | |
|---|---|
| Secuencia normal | 1. El actor indica que quiere recuperar la contraseña.<br>2. El sistema pide que introduzca su correo electrónico.<br>3. El actor introduce el correo electrónico.<br>4. El sistema comprueba que el correo introducido figura en el perfil de un usuario ya registrado y pide que confirmación al usuario.<br>5. El actor confirma.<br>6. El sistema envía un correo al usuario con un nuevo enlace.<br>7. El actor abre el enlace.<br>8. El sistema indica que introduzca una contraseña nueva.<br>9. El actor introduce la contraseña.<br>10. El sistema pide confirmación.<br>11. El actor confirma.<br>12. El sistema elimina del almacenamiento la contraseña y guarda la introducida por el usuario. |
| Excepciones | 3.b. El actor cancela y el caso de uso finaliza sin efecto.<br>4.b. El sistema comprueba que el correo electrónico introducido no figura en el almacenamiento. El sistema muestra un mensaje de error y vuelve al paso 3.<br>5.b. El actor cancela y el caso de uso finaliza sin efecto.<br>9.b. El actor cancela y el caso de uso finaliza sin efecto.<br>11.b. El actor cancela y el caso de uso finaliza sin efecto. |
| Comentarios | |

**Tabla 3.28. Descripción del caso de uso CU24, Recuperar contraseña**

| **CU25** | **Terminar ronda** |
|---|---|
| Definición | Se puede terminar una ronda porque un jugador ha llegado a meta o porque el actor moderador así lo quiere.<br>En todos los casos se guarda la puntuación hasta el momento. |
| Actor | Moderador |
| Precondición | Se ha de haber realizado el caso de uso “Iniciar partida” |
| Postcondición | |
| Secuencia normal | 1. El actor indica que quiere finalizar la ronda.<br>2. El sistema inhabilita las pantallas de los actores jugadores, redirige a la vista del CU “Monitorizar partida”, muestra la puntuación obtenida hasta el momento de todos los jugadores. |
| Excepciones | |
| Comentarios | Este caso de uso sustituye al CU13 “Terminar partida” en esta versión del proyecto. |

**Tabla 3.29. Descripción del caso de uso CU25, Terminar ronda**

## 3.4. Análisis de requisitos

En esta sección se detallan los modelos de casos de uso, de dominio, de flujo de la aplicación y, por último, el diagrama de iteración con el usuario.

### 3.4.1. Modelo de casos de uso

A continuación, en la figura 3.2, se muestra el diagrama de casos de uso del sistema.

Se debe aclarar que el caso de uso "Salir de la partida" es ejecutado por el usuario "jugador" siempre y cuando se haya realizado el caso de uso "Terminar partida".

### 3.4.2. Modelo de dominio

En la figura 3.3 se muestra el modelo de domino del sistema. Aclarar que el estereotipo *persistente* señala las clases del domino que serán almacenadas en la base de datos.

En la clase "Jugador" los atributos "cartas" y "penalizaciones" almacenan los puntos obtenidos por cada jugador cada vez que recogen una carta de punto o una carta de penalización, respectivamente. El valor de estos atributos se muestran en las pantallas de los jugadores en los marcadores y, cuando se finaliza la partida se calcula el valor del atributo "puntuación" como $cartas - penalizaciones$.

La clase "Variante" recoge las modificaciones especificadas en la sección 2.1.5 *Fin de la ronda* y de la sección 2.3 *Variaciones del juego*.

### 3.4.3. Modelo de flujo

En las figuras 3.4 y 3.5 se muestra el diagrama de flujo general del sistema y en la figura 3.6 el modelo de flujo que hace referencia al caso de uso "Realizar jugada", especificado en la tabla 3.10 de la sección 3.3.4, *Casos de uso*. Se ha dividido la secuencia en tres diagramas para facilitar su compresión.

### 3.4.4. Modelos de interacción en el usuario

Para mostrar el flujo de cada vista bocetada, las cuales se muestran en el apartado 6.2.1: bocetaje, se ha creado un modelo de interacción con el usuario según el rol del mismo: para el moderador se especifica en la figura 3.7, para el profesor se muestra en la figura 3.8 y para el jugador en la figura 3.9.

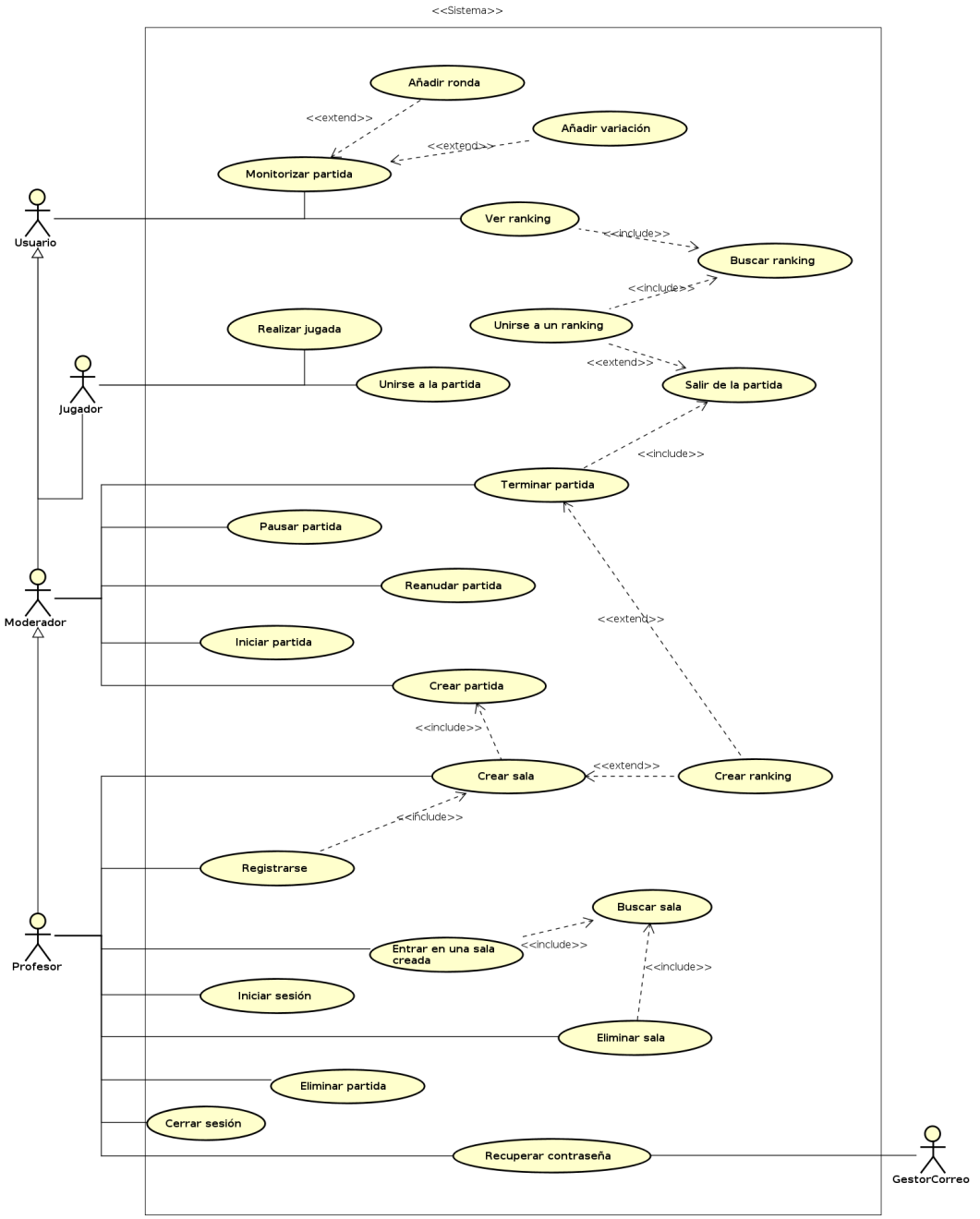


**Figura 3.2. Modelo de casos de uso del sistema.**

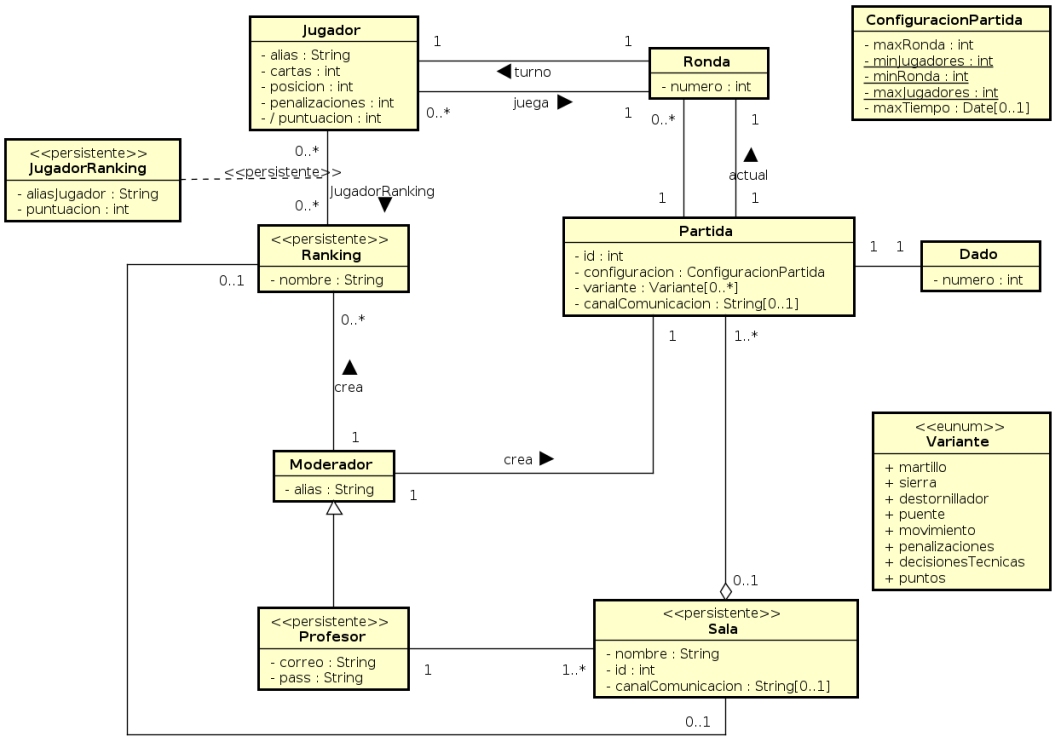


**Figura 3.3. Modelo de dominio del sistema.**

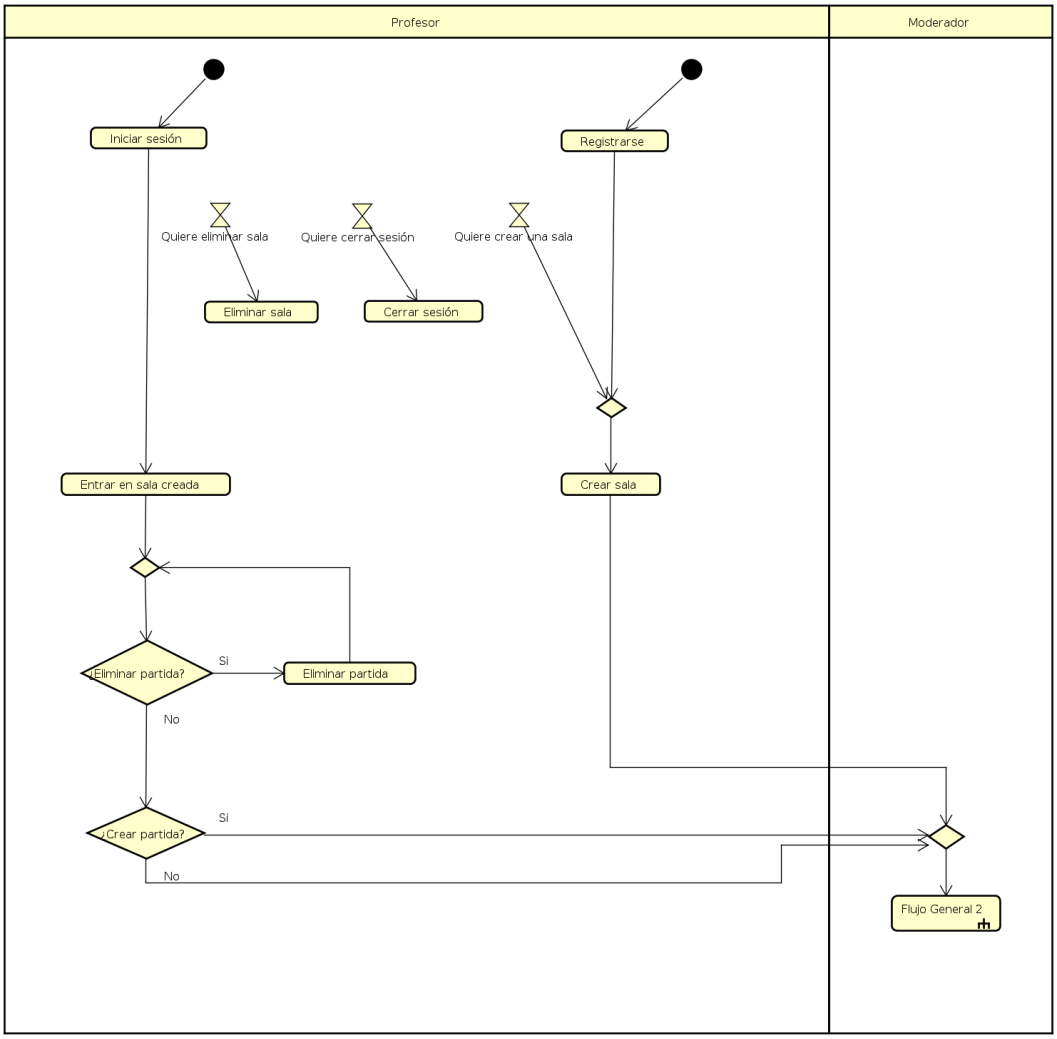


**Figura 3.4. Modelo de flujo del sistema parte I.**

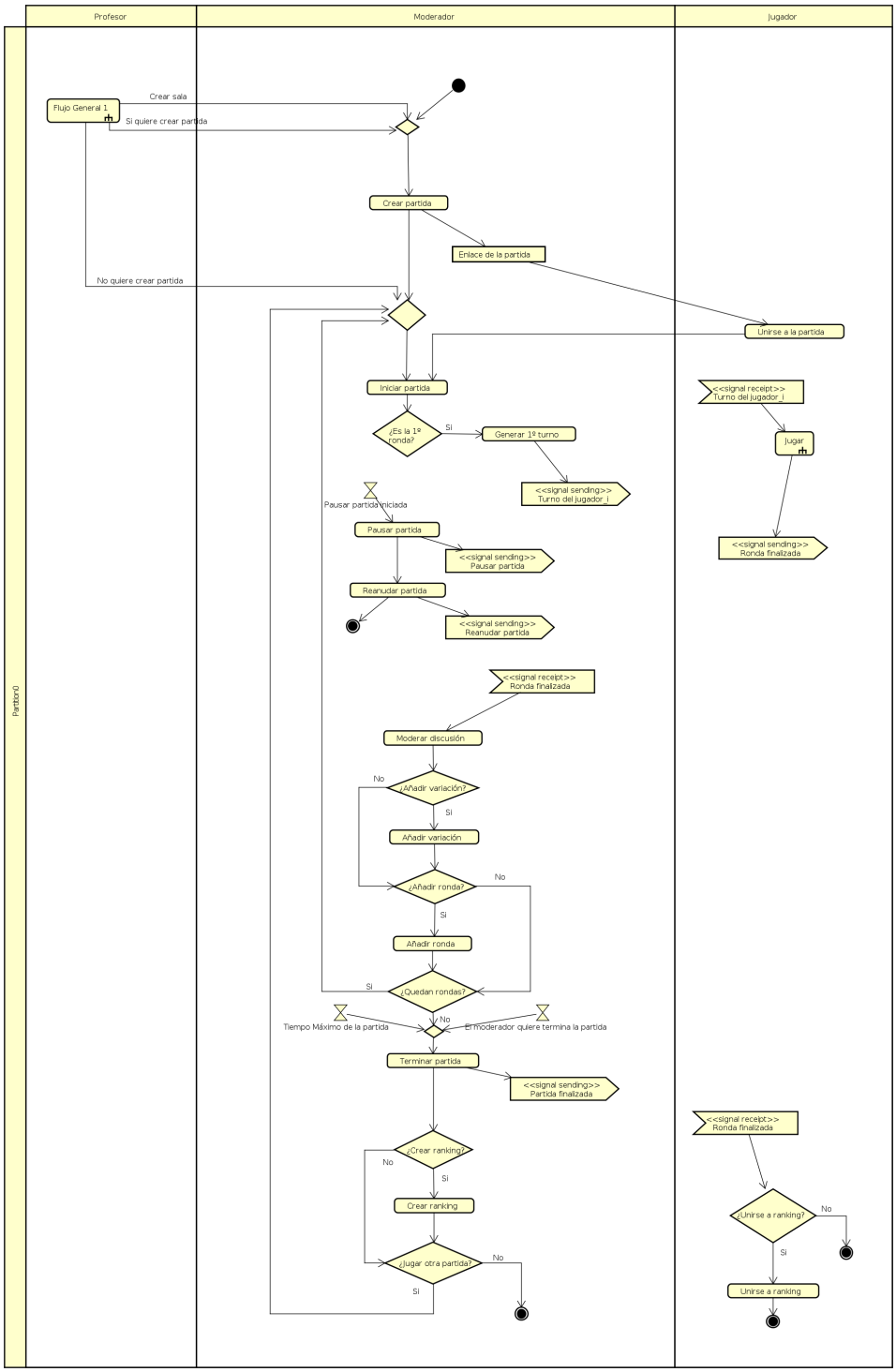


**Figura 3.5. Modelo de flujo del sistema parte II.**

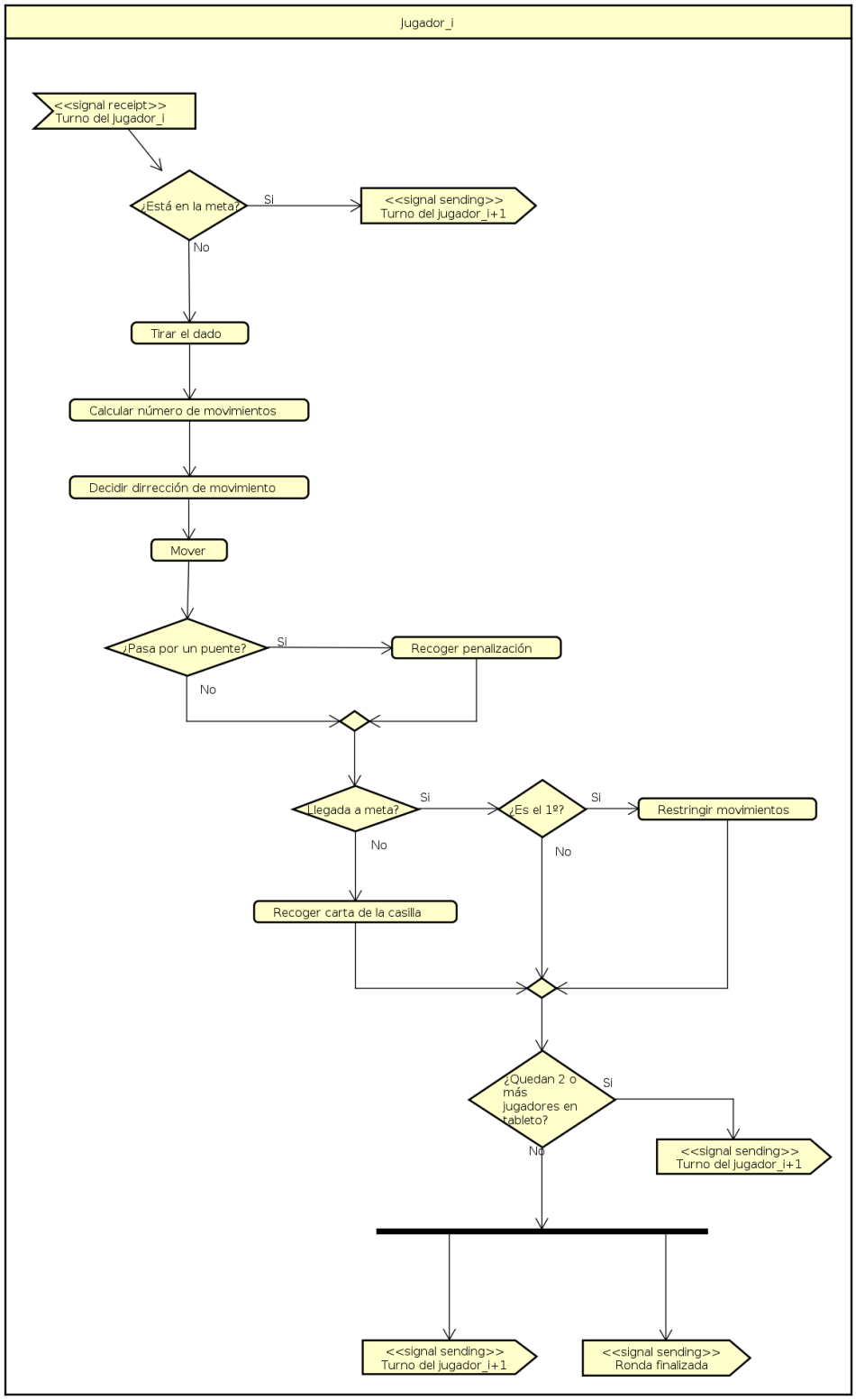




**Figura 3.6. Modelo de flujo del caso de uso jugar ronda.**

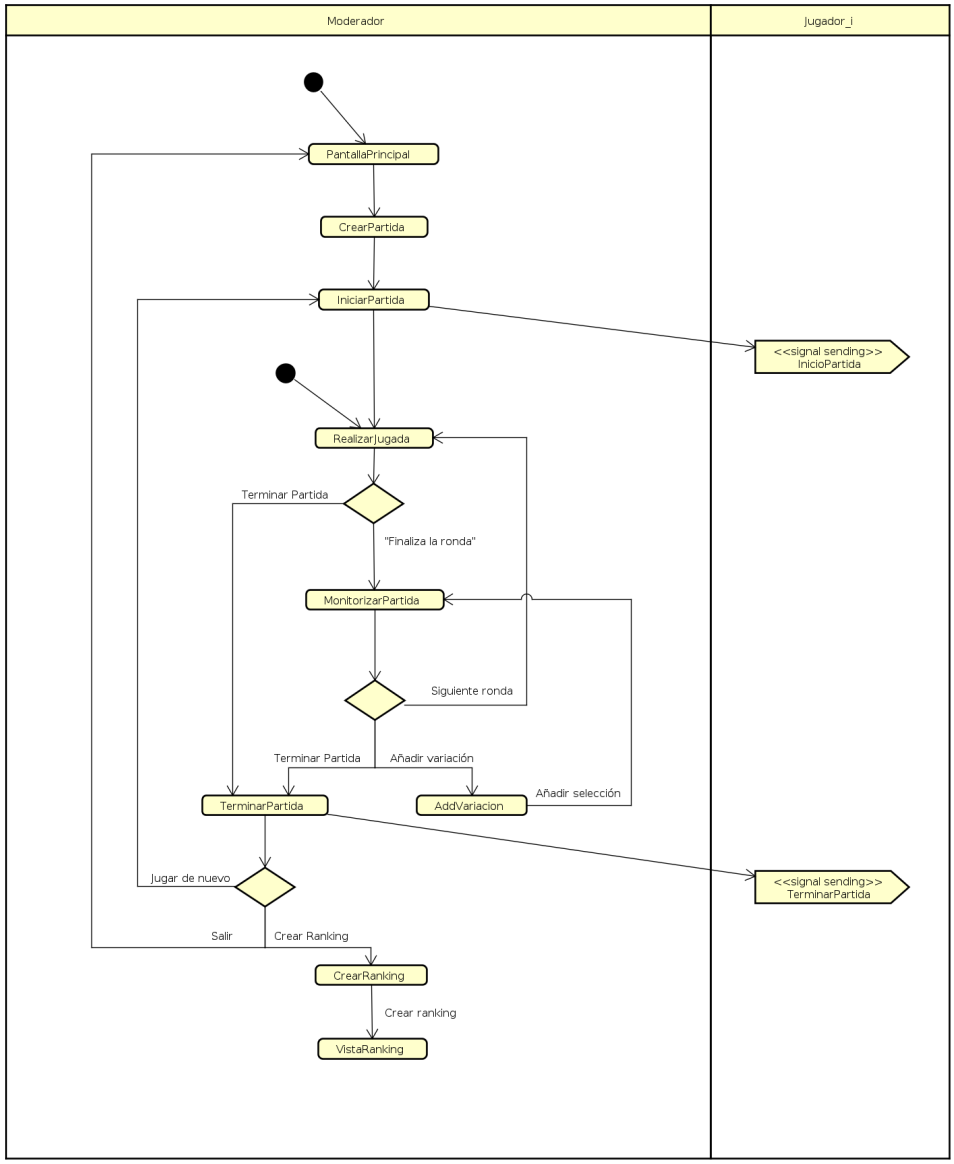


**Figura 3.7. Modelo de interacción para el usuario moderador**

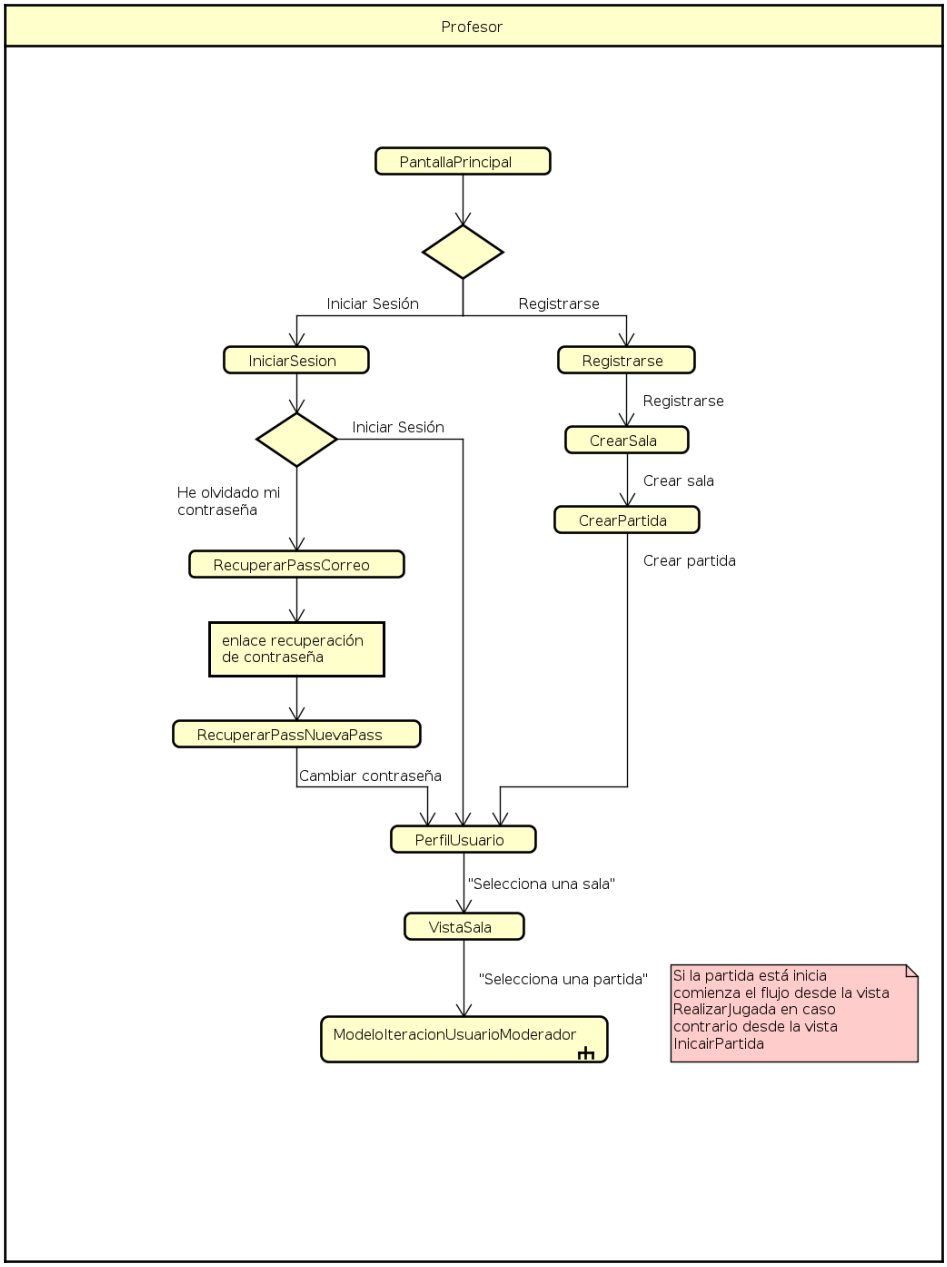


**Figura 3.8. Modelo de interacción para el usuario profesor**

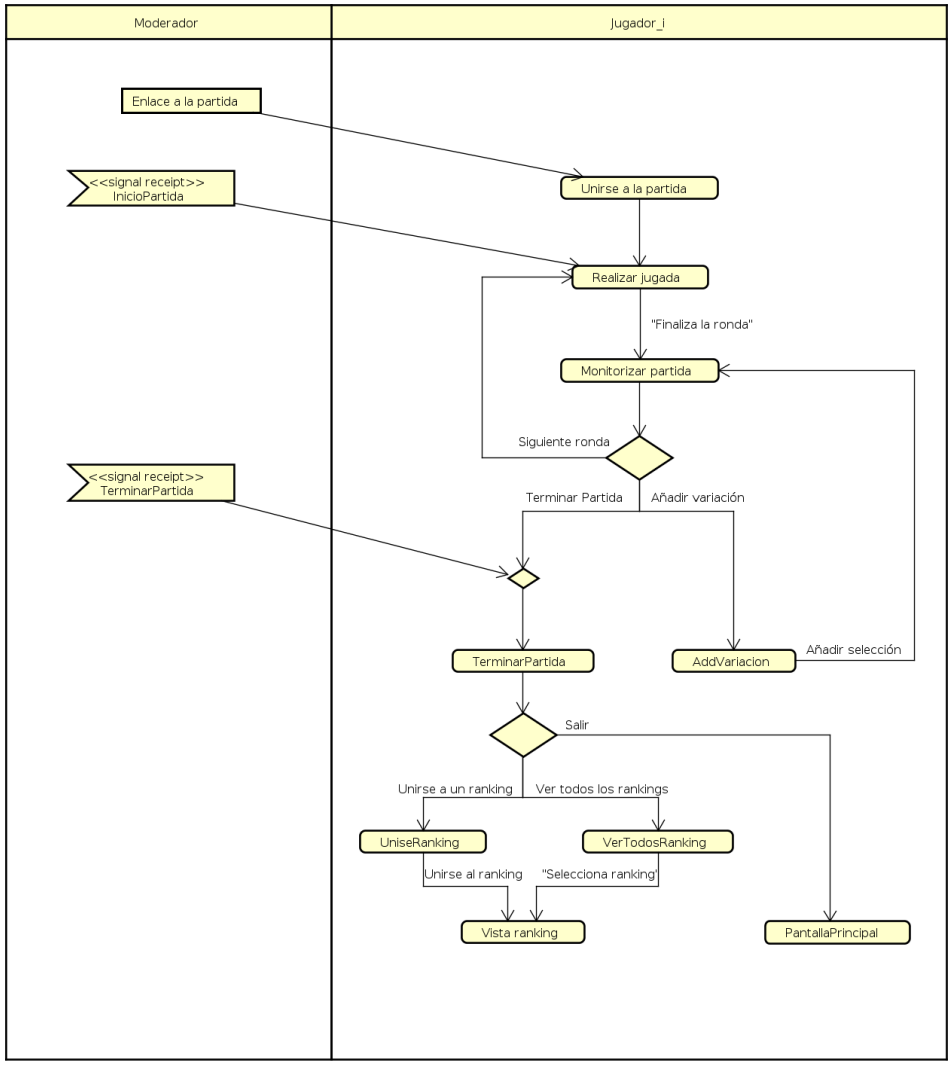


**Figura 3.9. Modelo de interacción para el usuario jugador**

# Capítulo 4

# Plan de proyecto

## 4.1. Resumen del proyecto

Se ha usado la metodología RUP para el desarrollo de este proyecto. Esta metodología se caracteriza por ser un proceso iterativo e incremental, dirigido por los casos de uso, centrado en la arquitectura y en la gestión de riesgos [14].

Las fases en las que se divide el proceso son:

**Fase de Inicio:** Se define el alcance del proyecto teniendo en cuenta los requisitos funcionales, se identifican los riesgos, los recursos y se realizan los planes para la fase correspondiente. Se fijan las metas, los costes, el modelo y los plazos.

**Fase de Elaboración:** Se refina lo planteado en la fase anterior y se toman las decisiones y medidas para reducir y paliar los riesgos con mayor impacto.

**Fase de Construcción:** Se desarrolla la versión correspondiente del producto buscando la mayor calidad y la optimización de los costes.

**Fase de Transición:** Se realizan las pruebas, se despliega y se muestra al cliente para que de su aprobación e indique nuevas mejoras a incluir. Por último, se buscan alternativas para que no sucedan los mismos errores en iteraciones futuras.

En la imagen 4.1 se muestra la estructura del proceso que se ha seguido.

### 4.1.1. Propósito, alcance y objetivos

El propósito de este proyecto es adaptar el juego de tablero *Hard Choices* a una página web con una interfaz que se adapte tanto a ordenadores, como a móviles y tablets para facilitar el uso del servicio en las aulas y con ello, ayudar en el aprendizaje sobre el concepto de

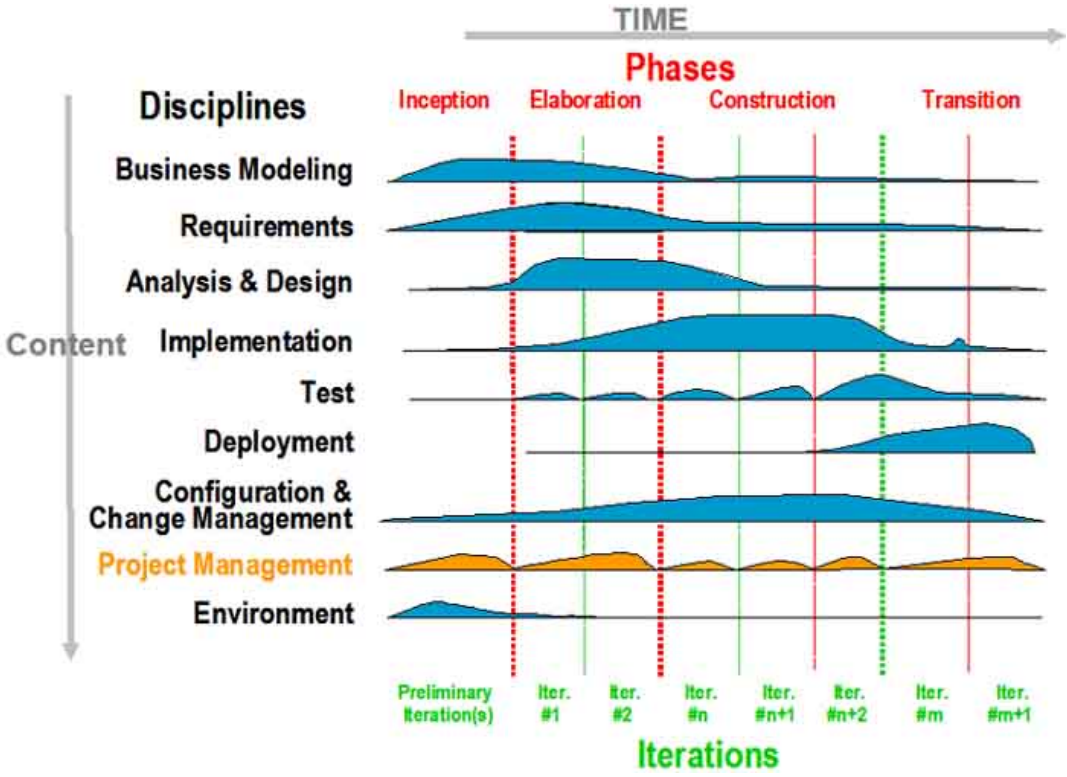


**Figura 4.1. Estructura de RUP [14]**

la deuda técnica a alumnos que en un futuro se dediquen al desarrollo software.

El objetivo es que los alumnos puedan jugar tanto al juego base como incluir variaciones antes de empezar nuevas rondas, además podrán guardar su puntuación en uno o varios rankings. Los profesores podrán crear salas con múltiples tableros de juego y gestionarlas desde un perfil creado previamente. Además, en cada sala y cada partida se podrá añadir, opcionalmente, un enlace a un canal de comunicación externo por si la partida se realiza de manera no presencial.

### 4.1.2. Definición y acrónimos

| Nombre | Significado |
|---|---|
| AOP | "Aspect-Oriented Programming", Programacion Orientada a Aspectos, paradigma de programación que aísla las funciones de apoyo de la lógica empresarial del programa principal. |

| | |
|---|---|
| API | "Application Programming Interface", conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software. |
| CSS | "Cascading Style Sheets", lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. |
| DAO | "Data Access Object", componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos. |
| HTML | "HyperText Markup Language", lenguaje de marcado para la elaboración de páginas web. |
| INEED | Red social de empleo colaborativo, creada para dar solución a la situación de desempleo actual. |
| JEE | "Java Platform, Enterprise Edition", plataforma de programación Java para desarrollar para ejecutar software de aplicaciones en el lenguaje de programación Java. |
| JPA | "Java Persistence API", colección de clases y métodos para almacenar de forma persistente grandes cantidades de datos en una base de datos. |
| JSON | "JavaScript Object Notation", Un formato de texto constituido por notación literal de objetos JavaScript. |
| NoSQL | Que no está basado en SQL: "Structured Query Language", Lenguaje específico para intercambiar información con una base de datos. |
| REST | un estilo arquitectónico basado en la transferencia de la representación de los recursos desde un servidor a un cliente |
| RUP | "Rational Unified Process", Proceso Racional Unificado, metodología de desarrollo de software impulsada por IBM. |
| SDK | "Software Development Kit ", kit de desarrollo de software. Conjunto de herramientas de desarrollo de software que permite crear una aplicación. |
| SEI | "Software Engineering Institute", centro de investigación y desarrollo estadounidense. |
| UML | "Unified Modeling Language", Lenguaje de modelado unificado, estándar para la modelización de sistemas. |
| BSD | "Berkeley Software Distribution", sistema operativo derivado de Unix. |

**Tabla 4.1. Acrónimos**

### 4.1.3. Artefactos del proyecto

RUP es un proceso iterativo por lo que en cada ciclo se obtiene los mismos artefactos cada vez más completos hasta llegar a los entregables finales.

De cada fase obtenemos:

- **Fase de Inicio:**
  - Contexto y objetivos del proyecto.

  - Gestión y plan de proyecto.
  - Planes de gestión iniciales.
  - Especificación de requisitos software.
  - Especificación de casos de uso.

- **Fase de Elaboración:**
  - Modelo de análisis.
  - Modelo de diseño.
  - Modelo de despliegue.
  - Prototipo interfaz de usuario.
  - Documento de casos de prueba.

- **Fase de Construcción:**
  - Prototipo de la aplicación.

- **Fase de Transición:**
  - Resultados de las pruebas.
  - Manual de instalación y despliegue.
  - Manual de usuario.

Todos los artefactos dichos anteriormente se encuentran en esta memoria a excepción del prototipo final que se encuentran en el archivo *.zip* adjunto.

### 4.1.4. Planificación de las tareas

Se ha hecho una planificación inicial de las tareas a realizar que se detalla en la tabla 4.2.

Como se puede ver en la tabla de la planificación se ha programado una primera fase, denominada iteración 0, donde se investiga sobre el concepto de la deuda técnica, el juego *Hard Choices* y todas las herramientas y metodologías necesarias para llevar a cabo el proyecto. Esta iteración es la más larga y tras terminarla se espera tener el análisis y diseño del proyecto para, en las siguientes iteraciones, centrarse en el desarrollo de la aplicación.

Para la iteración 1 y 2 se han planificado los casos de uso que se espera desarrollar, numerados por orden de prioridad, son:

| **Iteración 1** | 2. Unirse a la partida | 4. Realizar jugada |
|---|---|---|
| 1. Crear partida | 3. Iniciar partida | 5. Terminar partida |

6. Salir de la partida
7. Monitorizar partida
8. Añadir ronda
9. Añadir variación
10. Pausar partida
11. Reanudar partida
12. Registrarse
13. Iniciar sesión

**Iteración 2**

14. Cerrar sesión
15. Crear sala
16. Buscar sala
17. Entrar en sala creada
18. Crear ranking
19. Buscar ranking
20. Unirse a un ranking
21. Ver ranking
22. Eliminar partida
23. Eliminar sala
24. Cerrar sesión
25. Recuperar contraseña

Existe una 3ª iteración prevista como colchón por posibles retrasos, por tanto se volcarán en ella las tareas que no se hayan realizado en las iteraciones pasadas. Esta última fase se ha planificado con una duración aproximada del 10 % del total del proyecto [8].

Se ha detallado la duración prevista de cada fase y la duración total del proyecto: 337 horas y 30 minutos. Aproximadamente 38 horas más de las estipuladas en la planificación de la asignatura *Trabajo de Fin de Grado* de la Universidad de Valladolid [28].

Por último, destacar que se espera terminar el proyecto dos semanas antes de la fecha límite de solicitud de defensa: 28 de junio 2021 [4]. Para así, contar con otras 2 semanas de colchón.

| Iteración | Inicio Iteración | Fin Iteración | Fase | Tarea | Duración estimada |
|---|---|---|---|---|---|
| 0 | 08/02/2021 | 07/04/2021 | Inicio | Documentarse sobre la deuda técnica | 5 h 00 m |
| | | | | Documentarse sobre juegos educativos | 5 h 00 m |
| | | | | Explicar Hard Choices | 2 h 30 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Elicitación inicial de requisitos | 4 h 00 m |
| | | | | Primera versión de los capítulos 1,2 y 3 de la memoria | 26 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Redactar capítulo planificación | 6 h 00 m |
| | | | | Modificar introducción | 3 h 00 m |
| | | | | Modelo inical CU | 2 h 00 m |
| | | | | Modelo inicial de Dominio | 3 h 00 m |
| | | | | Modelo de proceso de negocio | 3 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Descripción CU | 3 h 00 m |
| | | | | Repaso de los diagramas CU, Dominio y proceso de negocio | 3 h 00 m |
| | | | | Rehacer planificación | 3 h 00 m |
| | | | | Documentarse sobre las tecnologías a usar en el proyecto | 5 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Terminar sección identificación de riesgos | 1 h 00 m |
| | | | | Terminar sección de requisitos | 2 h 00 m |
| | | | | Decidir tecnologías a utilizar y documentar | 4 h 00 m |
| | | | | Análisis arquitectónico | 2 h 00 m |

| | | | | | |
|---|---|---|---|---|---|
| | | | Elaboración | Diseño arquitectónico | 3 h 00 m |
| | | | | Bocetaje inicial | 3 h 00 m |
| | | | | Diseño del almacenamiento persistente | 4 h 00 m |
| | | | | Diseño detallado | 3 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Terminar sección de tecnologías a utilizar | 1 h 30 m |
| | | | | Terminar diseño arquitectónico | 2 h 00 m |
| | | | | Terminar bocetaje | 1 h 30 m |
| | | | | Terminar diseño de almacenamiento persistente | 2 h 00 m |
| | | | | Terminar diseño detallado | 2 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Tiempo total en la iteración | 110 h 30 m |
| 1 | 08/04/2021 | 02/05/2021 | Elaboración | Realizar casos de prueba | 20 h 00 m |
| | | | | Realizar diseño de interfaz de usuario | 12 h 00 m |
| | | | Construcción | Desarrollar CU | 80 h 00 m |
| | | | Transición | Realizar pruebas para cada CU | 10 h 00 m |
| | | | | Desplegar prototipo y documentar | 8 h 00 m |
| | | | | Redactar manual de usuario | 4 h 00 m |
| | | | | Tutoría | 2 h 00 m |
| | | | | Tiempo total en la iteración | 136 h 00 m |
| 2 | 03/05/2021 | 31/05/2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m |
| | | | Elaboración | Realizar casos de prueba | 15 h 00 m |
| | | | | Realizar diseño de interfaz de usuario | 8 h 00 m |
| | | | Construcción | Desarrollar CU | 50 h 00 m |

| | | | | | |
|---|---|---|---|---|---|
| | | | Transición | Realizar pruebas para cada CU | 8 h 00 m |
| | | | | Desplegar prototipo y documentar | 3 h 00 m |
| | | | | Redactar manual de usuario | 2 h 00 m |
| | | | | Tutoría | 2 h 00 m |
| | | | | Tiempo total en la iteración | 91 h 00 m |
| 3 | 01/06/2021 | 13/06/021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m |
| | | | Elaboración | Terminar casos de prueba | 5 h 00 m |
| | | | | Terminar diseño de interfaz de usuario | 3 h 00 m |
| | | | Construcción | Terminar desarrollo | 10 h 00 m |
| | | | Transición | Realizar todas las pruebas | 4 h 00 m |
| | | | | Desplegar prototipo y documentar | 2 h 00 m |
| | | | | Terminar manual de usuario | 2 h 00 m |
| | | | | Tutoría | 2 h 00 m |
| | | | | Tiempo total en la iteración | 31 h 00 m |
| | | | | | 337 h 30 m |

**Tabla 4.2. Planificación inicial.**

### 4.1.5. Replanificación de las tareas

En la iteración 1 no se realizó ningún caso de uso de los planeados ya que el tiempo empleado en el estudio y compresión de las tecnologías utilizadas en el proyecto fue más largo de lo esperado. Por esto, se ha realizado una replanificación de las tareas a partir de la iteración 2, esto queda reflejado en la tabla 4.3.

Destacar de la replanificación que:

- Se han eliminado las tareas de "Realizar diseño de interfaz de usuario" de todas las iteraciones ya que esto se hizo en la iteración 0.
- De la iteración 2 se han eliminado las tareas "Desplegar prototipo y documentar" y "Redactar manual de usuario" ya que se espera terminar de comprender las tecnologías y desarrollar los casos de uso que se debían haber sido realizados en la iteración pasada.
- Debido a todos los retrasos acumulados se ha decidido presentar este trabajo de fin de grado en la convocatoria de julio, se planifica terminar el proyecto el 12/07/2021 para que la tutora, Yania Crespo, pueda corregir esta memoria antes de su entrega el 19/07/2021. Por esto se han modificado las fechas de inicio y fin de las iteraciones 2 y 3 y se ha añadido una cuarta iteración.

| Iteración | Inicio Iteración | Fin Iteración | Fase | Tarea | Duración estimada |
|---|---|---|---|---|---|
| 0 | | | | Tiempo total en la iteración | 110 h 30 m |
| 1 | | | | Tiempo total en la iteración | 136 h 00 m |
| 2 | 13/05/2021 | 09/06/2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m |
| | | | Elaboración | Replanificar y documentarlo | 2 h 00 m |
| | | | | Estudiar las tecnologías a usar | 40 h 00 m |
| | | | Construcción | Desarrollar CU | 50 h 00 m |
| | | | Transición | Realizar pruebas para cada CU | 10 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Tiempo total en la iteración | 106 h 00 m |
| 3 | 10/06/2021 | 30/06/2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m |
| | | | Elaboración | Realizar casos de prueba | 15 h 00 m |
| | | | Construcción | Desarrollar CU | 100 h 00 m |
| | | | Transición | Realizar pruebas para cada CU | 8 h 00 m |
| | | | | Desplegar prototipo y documentar | 3 h 00 m |
| | | | | Redactar manual de usuario | 2 h 00 m |
| | | | | Tutoría | 1 h 00 m |
| | | | | Tiempo total en la iteración | 132 h 00 m |
| 4 | 01/07/2021 | 12/07/021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m |
| | | | Elaboración | Terminar casos de prueba | 5 h 00 m |
| | | | Construcción | Terminar desarrollo | 50 h 00 m |
| | | | Transición | Realizar todas las pruebas | 4 h 00 m |

| | |
|---|---|
| Desplegar prototipo y documentar | 2 h 00 m |
| Terminar manual de usuario | 2 h 00 m |
| Tutoría | 1 h 00 m |
| Tiempo total en la iteración | 67 h 00 m |
| Total | 484 h 30 m |

**Tabla 4.3. Replanificación inicial.**

### 4.1.6. Plan de gestión de riesgos

Un riesgo es un evento o condición que, si ocurre, puede tener efectos positivos o negativos en los objetivos del proyecto. En la gestión de riesgos se intenta identificar el mayor número posible de ellos para, tras una evaluación, planificar si aceptarlos, evitarlos, reducirlos o transferirlos fuera de la organización. Una vez tomada estas decisiones se elaboran planes de contingencia y mitigación para los riesgos que se han decidido abordar [10].

Para realizar esta planificación se han seguido los siguientes pasos:

- Identificar los riesgos
- Priorizar los riesgos identificados. Esto se hace gracias a la matriz de probabilidad de impacto mostrada en la figura 4.2 que, tras analizar la probabilidad de ocurrencia y el impacto, indica la exposición al riesgo. Los riesgos que se sitúen encima de la línea de tolerancia son los que tendrán una mayor prioridad.
- Planificar el riesgo. Como se ha dicho antes se decide si aceptarlos, evitarlos, reducirlos o transferirlos.
- Especificar los planes de mitigación y contingencia de los riesgos.

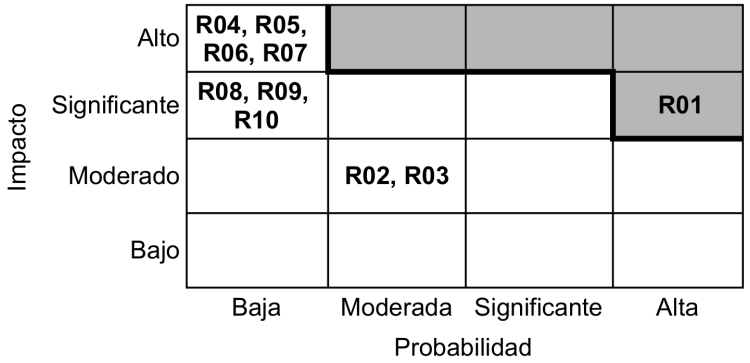


**Figura 4.2. Matriz de probabilidad de impacto [10]**

Para asociar un nivel de probabilidad a los riesgos se tiene en cuenta la tabla 4.4.

| Nivel de probabilidad | Posible suceso( %) |
|---|---|
| Alto | 50 % |
| Significante | 30 %-50 % |
| Moderado | 10 %-29 % |
| Bajo | 10 % |

**Tabla 4.4. Asociación de un nivel de probabilidad según la posibilidad de que suceda el riesgo [10]**

Para asociar un nivel de impacto al riesgo se tiene en cuenta la tabla 4.5

| Nivel de impacto | Posible gasto de presupuesto(%) |
|---|---|
| Alto | 30 % |
| Significante | 20 %-29 % |
| Moderado | 10 %-19 % |
| Bajo | 10 % |

**Tabla 4.5. Asociación de un nivel de impacto según el posible presupuesto que gaste [10]**

En la tabla 4.6 se muestran los riesgos identificados para este proyecto, la probabilidad de que sucedan, su impacto, así como sus planes de contingencia y mitigación.

Como se indica en la matriz de probabilidad de impacto, figura 4.2, el único riesgo que hay que priorizar es el riesgo R01, Fallo de planificación, además este riesgo ocurrió por lo que se puso en marcha el plan de contingencia: se prioriza en la planificación los casos de uso que hacen la línea base del juego y se dejan para el final los de gestión del perfil del usuario profesor.

| ID | Nombre | Descripción | Probabilidad | Impacto | Acciones de mitigación | Acciones de contingencia |
|---|---|---|---|---|---|---|
| R01 | Fallo de planificación | Los tiempos estimados en la planificación son insuficientes para realizar la totalidad de los objetivos del proyecto | Alta | Significante | Cumplir con el horario de trabajo y los plazos de cada ciclo | Priorizar CU a desarrollar, terminar la línea base del juego y dejar variaciones para próximas versiones |
| R02 | Enferma la alumna | La alumna a cargo del proyecto enferma lo que la impide seguir desarrollando el proyecto | Moderada | Moderado | No retrasarse en las tareas planificadas e incluir un colchón en la planificación de entorno a semanas | Cambiar horario de trabajo para dedicar más horas en el día al proyecto una vez recuperada de la enfermedad |
| R03 | Enferma la tutora | La tutora a cargo del proyecto enferma lo que la impide seguir desarrollando el proyecto | Moderada | Moderada | No retrasarse en las tareas planificadas e incluir un colchón en la planificación de entorno a semanas | Atrasar las tutorías hasta que la tutora se recupere y seguir con los siguientes ciclos planeados |

| | | | | | | |
|---|---|---|---|---|---|---|
| R04 | Tecnología escogida inadecuada | La tecnología que se ha escogido para el desarrollo del proyecto no permiten llevar a cabo el trabajo de la manera esperada | Baja | Alto | Estudiar minuciosamente las tecnologías, comparar con otras y pedir consejo a personas que las hayan utilizado o hayan estado involucrados en un proyecto similar a este | Buscar una tecnología que sustituya a la inadecuada, preferiblemente una que ya conozca la alumna y no requiera de mucho aprendizaje |
| R05 | Mala comunicación | Falta de comunicación entre la tutora y la alumna, esto provocaría retrasos en el proyecto y fallos en los requisitos ya que la tutora es también el cliente | Baja | Alto | Planificar las reuniones en un espacio en la que las dos personas puedas asistir sin dificultad y establecer un canal de mensajería instantánea como rocket chat | Cambiar la forma de realizar las reuniones para que se adapte a los horarios de ambas personas y replanificar tareas y horarios de la alumna |
| R06 | Perdida de datos | Se pierden datos y/o documentación ya desarrollados | Baja | Alto | Tener todos los datos y documentación que se vaya creando repositorios en la nube como son Gitlab, para el código y las tareas, Overleaf, para la memoria y Google Drive para documentos de apoyo | Replanificar las tareas para realizar nuevamente lo perdido |
| R07 | Avería del ordenador | El ordenador de la alumna se rompe impidiéndola continuar con la realización del proyecto | Baja | Alto | Usar repositorios en la nube para poder acceder al trabajo realizado desde otro ordenador y continuar con la planificación | Hacer uso de la garantía del ordenador actual y usar el antiguo para continuar con el desarrollo |

| | | | | | | |
|---|---|---|---|---|---|---|
| R08 | Mala elicitación de requisitos | Los requisitos especificados no corresponde con lo pedido por el cliente y, con ello, los casos de uso no representan lo que el sistema ha de hacer | Baja | Significante | Comunicar cualquier duda al cliente sobre los requisitos y así realizar un análisis lo más completo posible | Replanificar tareas y corregir errores |
| R09 | Cambios en los requisitos | El cliente pide cambios en los requisitos del sistema | Baja | Significante | Comunicar cualquier duda al cliente sobre los requisitos y así realizar un análisis lo más completo posible | Replanificar tareas y realizar los cambios pedidos |
| R10 | Dificultad en el aprendizaje de las tecnologías escogidas | A la alumna la cuesta más de lo esperado aprender a usar las tecnologías escogidas para el desarrollo del proyecto | Baja | Significante | Escoger las tecnologías en las que la alumna tenga una buena base y exista documentación de apoyo en internet | Cambiar tecnologías y replanificar tareas |

**Tabla 4.6. Gestión de riesgos para este proyecto**

### 4.1.7. Presupuesto

A continuación se detalla el presupuesto estimado para el desarrollo del proyecto. Para ello se especifican los recursos utilizados y la estimación de costes.

#### Asignación de recursos

Un recurso es una persona u objeto requerido para llevar a cabo el proyecto. Se dividen en 7 categorías: mano de obra, equipamiento, materiales, espacio, servicios, tiempo y dinero [9].

A continuación se especifican los recursos necesarios para el desarrollo del proyecto:

- **Mano de obra.** Como este proyecto es un trabajo de fin de grado solo hay dos personas en él: la tutora, Yania Crespo, como cliente, ya que la idea del proyecto es suya, y jefa de proyecto y la alumna, Rebeca Hernando, que desempeña el resto de roles: analista, desarrolladora y gestor del proyecto.

- **Equipamiento.** El ordenador y móvil personal de la alumna y un dispositivo Apple para las pruebas en el navegador Safari.
- **Materiales.** Licencias para el uso de algunos servicios.
- **Espacio.** El espacio de trabajo para este proyecto será la habitación de la alumna y, en algunas ocasiones, los espacios públicos de estudio como las bibliotecas y salas de trabajo de la universidad.
- **Servicos.**
  - Gitlab
  - Overleaf
  - Hojas de cálculo de Google
  - Rocket Chat
  - Jitsi
  - Balsamiq
  - Astah professional
  - Angular
  - Framework Spring
  - Firebase
  - NetBeans
  - Visual Studio Code
- **Tiempo.** Especificado en la sección 4.1.4 *Planificación de las tareas.*
- **Dinero.** Especificado en la sección 4.1.7 *Estimación de costes.*

#### Estimación de costes

Como este es un proyecto para un trabajo de fin de grado el coste real del mismo es de 0€ ya que no se va a invertir dinero en él. Por tanto, se supone que los recursos de equipamiento ya están amortizados o son prestados y los recursos materiales son licencias gratuitas aportadas por la Escuela de Ingeniería Informática de la universidad de Valladolid para que, los servicios necesarios sean gratuitos siempre y cuando su fin sea educativo.

Sin embargo, se ha realizado una estimación de los costes como si se tratara de un proyecto de una entidad privada con el fin de poner en práctica los conocimientos aprendidos sobre planificación de proyectos. Este presupuesto se muestra en la tabla 4.9 de esta misma sección.

Se ha buscado en la página web de la INEED [11] los salarios base promedio de cada recurso de mano de obra implicado en el proyecto, quedando recogidos en la tabla 4.7. Para calcular el coste total de cada recurso de este tipo se ha usado la duración total del proyecto calculada en la sección 4.1.4, Planificación de las tareas: 337 horas y 30 minutos.

| Recurso | Salario base promedio (€/hora) |
|---|---|
| Analista | 13,50 € |
| Desarrolladora | 3,30 € |
| Gestor de proyecto | 15,86 € |

**Tabla 4.7. Salario base de cada recurso de mano de obra (€/hora)**

Se buscó en las páginas oficiales de cada recurso servicio cuánto cuesta comprar las licencias que ofrece gratuitamente la Escuela de Ingeniería Informática y, estimando que la realización del proyecto se hará desde febrero hasta julio, 5 meses, se escogió la opción más económica. Esta información queda reflejada en la tabla 4.8. Aclarar que se ha supuesto que si no se dispone del servicio GitLab, por ser un proyecto de una entidad privada, se usaría GitHub que es un repositorio gratuito.

| Recurso | Coste | Nota |
|---|---|---|
| Balsamiq | 7,60 € | €/mes durante un año |
| Astah professional | 45,00 € | Por 6 meses |

**Tabla 4.8. Coste de cada recurso servicio.**

A continuación se detalla el presupuesto estimado.

| Categoría | Recurso | Coste/hora | Coste total |
|---|---|---|---|
| Mano de obra | Analista | 13,50 € | 4.556,25 € |
| | Desarrolladora | 3,30 € | 1.113,75 € |
| | Gestor de proyecto | 15,86 € | 5.352,75 € |
| | **SubTotal** | | **11.022,75 €** |
| Equipamiento | Portatil Lenovo IDeaPad S145 | | 598,99 € |
| | Samsung Galaxy J7 | | 220,00 € |
| | iPad | | 375,00 € |
| | **SubTotal** | | **1.193,99 €** |
| Material | | | |
| Servicios | Firebase | | - € |
| | GitLab / GitHub | | - € |
| | Overleaf | | - € |
| | Hojas de cálculo de Google | | - € |
| | Rocket Chat | | - € |
| | Jitsi | | - € |
| | Angular | | - € |
| | Framework Spring | | - € |
| | NetBeans | | - € |
| | Visual Studio Code | | - € |
| | Balsamiq | | 38,00 € |
| | Astah professional | | 45,00 € |
| | **SubTotal** | | **83,00 €** |
| | | **Total** | **12.299,74 €** |

**Tabla 4.9. Simulación del presupuesto del proyecto.**

# Capítulo 5

# Tecnologías utilizadas

A continuación se detallan las tecnologías utilizadas durante la realización del proyecto tanto para la gestión, análisis y diseño, como para el desarrollo del mismo.

## 5.1. Tecnologías para gestión, análisis y diseño del proyecto

Las herramientas que se han utilizado para la gestión, el análisis y el diseño del proyecto han sido las aprendidas en el grado en Ingeniería Informática ya sea en las asignaturas cursadas o de forma autodidacta.

A continuación se detallan las tecnologías utilizadas:

- **Astah professional** para la realización de los modelos de análisis y diseño en UML.
- **Balsamiq** para el bocetaje de las interfaces de usuario.
- **Gitlab** como repositorio remoto.
- **Hojas de cálculo de Google** es la versión equivalente a Microsoft Excel que implementa Google en Google Drive. Se ha escogido esta opción ya que a lo largo del grado se ha aprendido de forma autodidacta y hay diferencias, aunque pequeñas, con la original de Microsoft.
  Esta herramienta se ha utilizado para el calculo de horas en la planificación y el calculo del presupuesto.
- **Overleaf** para la redacción de la memoria y repositorio remoto de la misma.
- **Rocket Chat** y **Jitsi** como canales de comunicación entre la tutora y la alumna.

## 5.2. Tecnologías para el desarrollo frontend

Para el desarrollo frontend se ha decidido utilizar Angular, una plataforma *TypeScript* que incluye *frameworks* para el desarrollo de aplicaciones web escalables y bibliotecas integradas que cubren el enrutamiento, administración de formularios, comunicación cliente-servidor y más. Además, incluye herramientas para el desarrollo que ayudan a compilar, probar y actualizar el código creado.

Con Angular se construyen aplicaciones basadas en componentes que usan *TypeScript* para los decoradores e inyección de dependencias, HTML para las plantillas de las interfaces y CSS, opcionalmente, para los estilos [2]. Además, en este proyecto se usa la biblioteca *Bootstrap*, un kit de herramientas de código abierto que facilita el desarrollo de interfaces adaptativas y dispone de una gran de variedad de componentes para el estilo de las mismas [3].

El desarrollo frontend se ha realizado en el editor de código *Visual Studio Code* siguiendo las recomendaciones de los profesores de la asignatura "Desarrollo basado en componentes y seervicios" del grado cursado.

## 5.3. Tecnologías para el desarrollo backend

Las tecnologías utilizadas en el desarrollo backend han sido el *framework Spring*, editado con *NetBeans* y para la base de datos y la autenticación de usuarios, *Firebase Realtime*.

### 5.3.1. Framework Spring

Para el desarrollo backend se ha escogido el *framework Spring* de código abierto que facilita la creación de aplicaciones JEE ya que proporciona funcionalidades ya desarrolladas para la creación de las mismas [26].

*Spring* permite crear aplicaciones utilizando los principios de alta cohesión y bajo acoplamiento fomentando el uso de capas bien diferenciadas en toda la aplicación y clases simples de Java (POJO) para la capa de negocio. Por tanto, insisten en el uso del paradigma AOP (Programación Orientada a Aspectos) el cual propone la modularización de las aplicaciones y el reparto de la funcionalidad entre los diferentes módulos. Para ello, facilitan un conjunto de módulos con distintas responsabilidades pero no fuerzan el uso de todos ellos en las aplicaciones, dejan que el programador seleccione cuales son necesario para su sistema.

En la figura 5.1 se muestra un esquema de los módulos proporcionados por el *framework*.

Aunque los desarrolladores de *Spring* permiten el uso de diferentes patrones facilita la utilización de los patrones Inversión de Control e Inyección de Dependencias [25], explicados

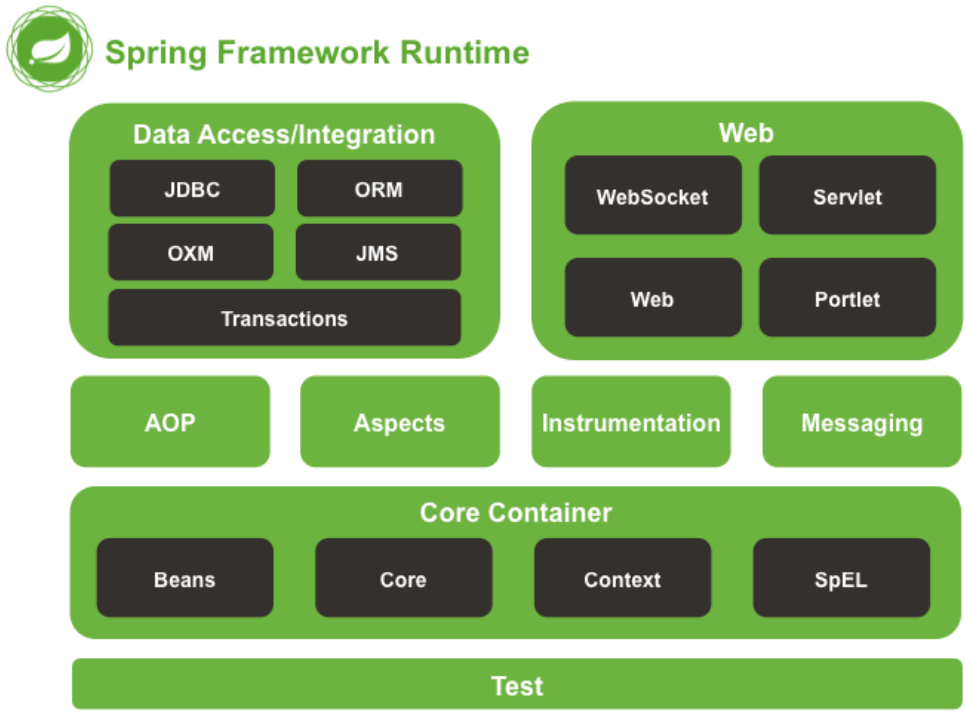


**Figura 5.1. Módulos proporcionados por Spring Framework [25]**

en la secciones 6.4.2 y 6.4.3 respectivamente.

### 5.3.2. Firebase

*Firebase* es una plataforma para el desarrollo de aplicaciones web y móviles comprada por *Google* en 2014 que facilita una API para guardar y sincronizar datos en la nube en tiempo real. Además, permite sincronizar nuestra aplicación con otros productos de *Google* [12] pero esta funcionalidad no es de interés para esta aplicación.

Para este proyecto se ha decidido utilizar *Firebase Realtime Database*, una base de datos NoSQL la cual almacena los datos en formato JSON y permite sincronizar los datos con todos los clientes a tiempo real. Destacar también que esta herramienta continúa respondiendo incluso sin conexión a internet, dado que el SDK de *Realtime Database* hace que los datos persistan en el disco y cuando se restablece la conexión se efectúa la sincronización.

*Firebase Realtime Database* junto a *Firebase Authentication* optimiza el proceso de autenticación en la aplicación y permite dividir la información en distintas instancias de la base de datos haciendo así más escalable el proyecto [5].

Como se ha dicho, para la autenticación en la aplicación se utilizará *Firebase Authentication* ya que proporciona servicios de backend, SDK sencillos y bibliotecas ya elaboradas tanto para identificar a los usuario como para la recuperación de contraseñas y vinculación de cuentas asegurando que no haya ninguna repercusión de seguridad en el proceso.

*Firebase Authentication* proporciona *FirebaseUI Auth* como solución completa de autenticación directa. Permite tanto la identificación por correo y contraseña, la que se usa en este proyecto, como la identificación por número de teléfono y por proveedores de identidad como *Google* y *Facebook* [6].

Por último, destacar que como en este proyecto se usa el plan gratuito de *Firebase* hay que tener en cuenta las limitaciones que esto conlleva, las cuales quedan reflejadas en la tabla 5.1 y se consideran asumibles [7].

| **Realtime Database** | |
|---|---|
| Conexiones simultáneas | 100 |
| GB almacenados | 1 GB |
| GB descargados | 10 GB/mes |
| Varias bases de dato por proyecto | NO |

**Tabla 5.1. Limitaciones del plan gratuito de *Firebase***

# Capítulo 6

# Diseño

## 6.1. Decisiones de diseño

Para el desarrollo de este proyecto se han tomado las siguientes decisiones de diseño:

- El desarrollo backend será con Java 11 y el *framework Spring*.
- El desarrollo frontend se realizar con el *framework AngularCLI*.
- Para el estilo de las interfaces de usuario se usará el repositorio Bootstrap.
- Se utilizará JPA aportado por *Spring Boot* para la especificación de las entidades, mapeo con la base de datos y los servicios REST.
- Se usará una arquitectura Cliente-Servidor con "cliente grueso" (*fat client*) de tres niveles con una estructuras por capas que sigue el patrón Modelo-Vista-Controlador.
- La base de datos a utilizar será *Firebase realtime*, no relacional, a la cual se accederá mediante DAOs.
- El despliegue de la aplicación se lanzará sobre un servidor *Tomcat* 8.

## 6.2. Diseño de la interfaz de usuario

A continuación se muestran los bocetos creados para las interfaces de usuario mediante la herramienta *Balsamiq*. Estos bocetos corresponden con los diagramas de análisis de la sección 3.4.4: modelos de interacción con el usuario.

### 6.2.1. Bocetaje

En esta sección se muestran todas las pantallas de la página web con algunas aclaraciones pertinentes.

Los bocetos muestran una pantalla de ordenador y una de teléfono móvil ya que las interfaces de esta aplicación pretenden ser adaptativa.

En la pantalla principal o de inicio, figura 6.1, se puede iniciar el caso de uso "Crear partida" introduciendo un alias público y tomando así el rol de moderador de la partida. Otra opción que puede tomar el usuario es la de crear una nueva cuenta o iniciar sesión para gestionar más de una partida desde el rol de profesor.


**Figura 6.1. Boceto de la pantalla principal**

Si el usuario opta por registrarse se mostrará la pantalla de la figura 6.2 y tras cumplimentar el formulario correctamente y dar al botón "registrarse" será redirigido a la pantalla de la figura 6.3 para que cree una sala, ya que cada cuenta de profesor ha de tener, al menos, una sala creada. Cada sala creada ha de tener, mínimo, una partida por lo que, una vez completado el formulario para crear una sala, se redirigirá al usuario a la figura 6.4 para que cree una partida.

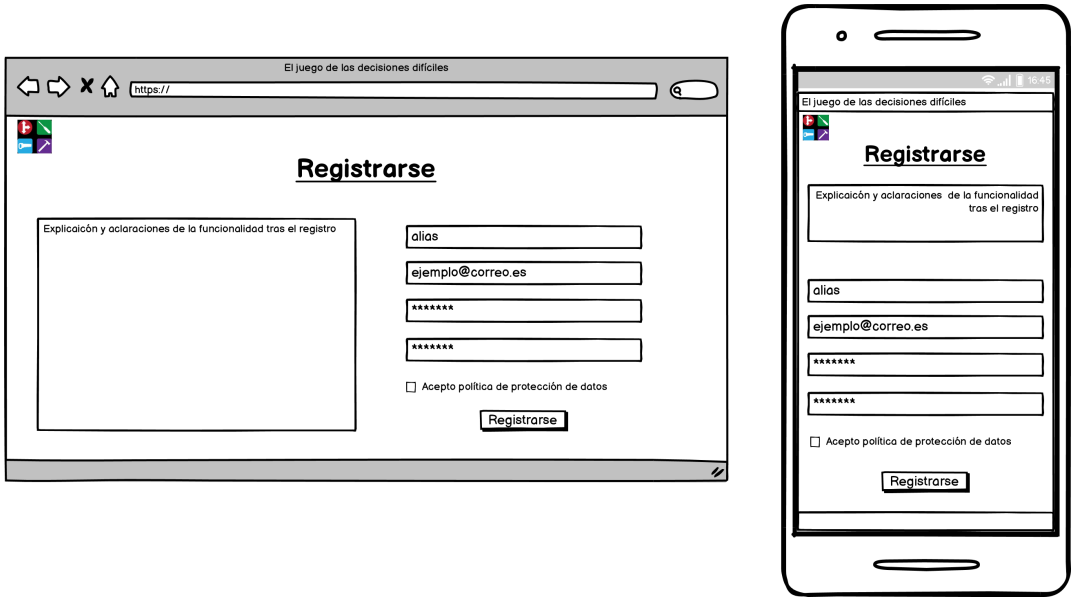


**Figura 6.2. Boceto de la pantalla registrarse**

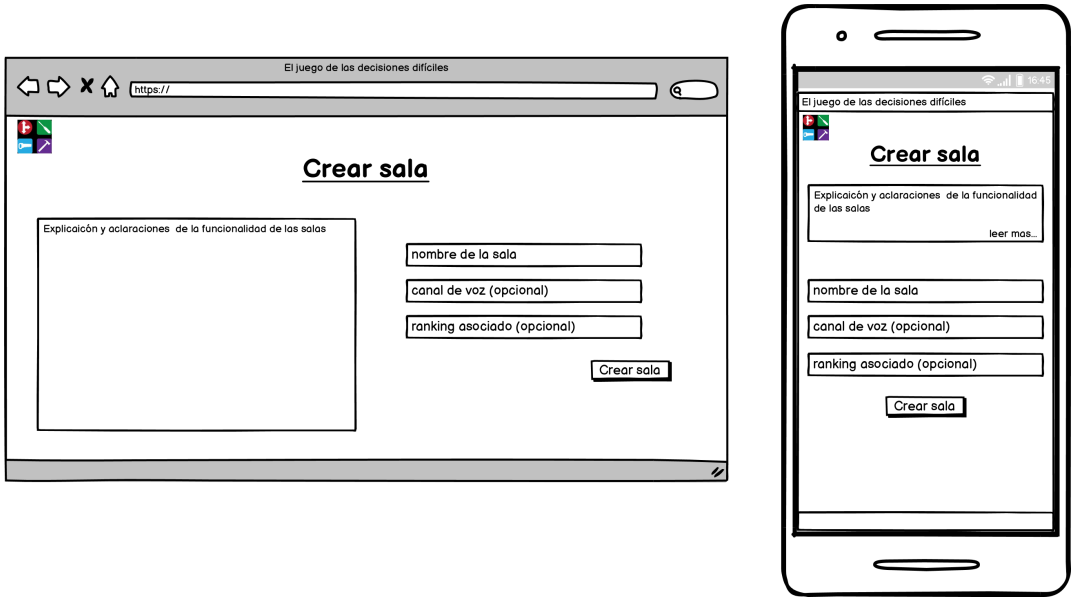


**Figura 6.3. Boceto de la pantalla para crear una sala**

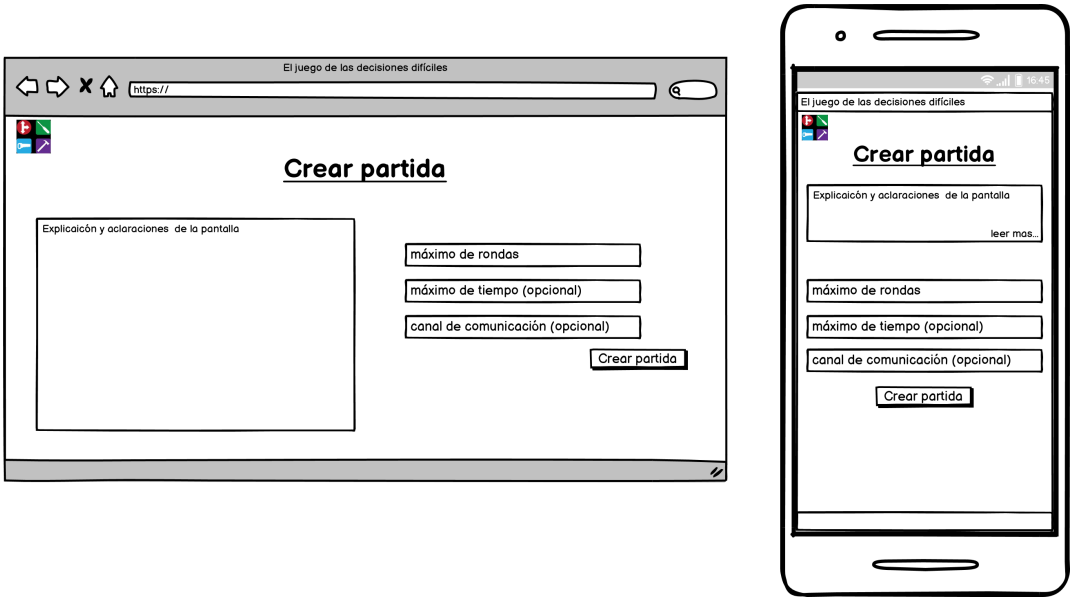


**Figura 6.4. Boceto de la pantalla para crear una partida**

Si el usuario quiere ver su perfil creado se le mostrará la figura 6.5 donde tendrá un menú desde el que podrá realizar distintas acciones sobre las salas creadas en su perfil. Además, podrá verlas pinchando sobre la que desee ver.

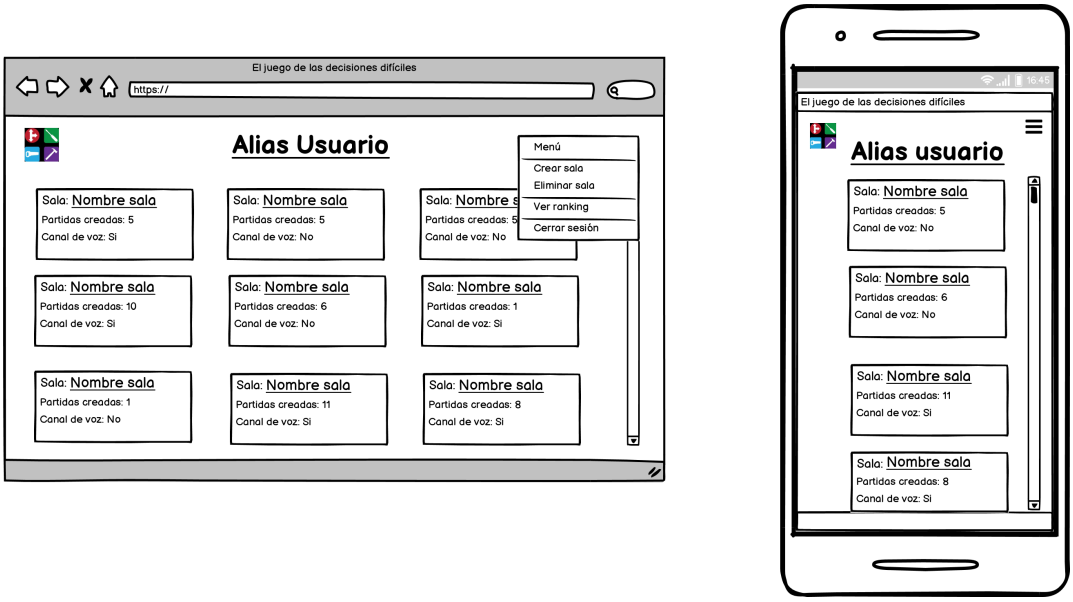


**Figura 6.5. Boceto del perfil de un usuario**

Una vez seleccionada la sala se mostrará la figura 6.6. Si cierra sesión será redirigido a la pantalla principal, figura 6.1, y si desea ver un ranking primero se le mostrará la pantalla de la figura 6.7 donde deberá seleccionar un ranking para ver los pares jugador-puntuación como se muestra en la figura 6.8.


**Figura 6.6. Boceto de la vista de una sala de un perfil de un usuario**


**Figura 6.7. Boceto de la vista de todos los rakings creados en la aplicación**

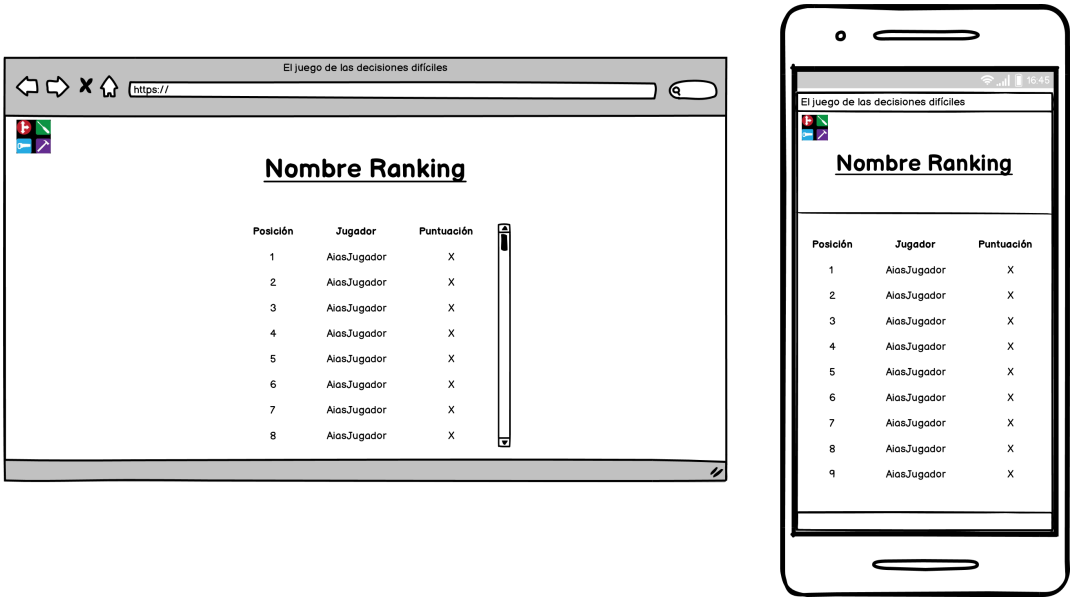


**Figura 6.8. Boceto de la vista de los pares jugador-puntuación de un ranking**

Si un usuario indica que quiere iniciar sesión se le mostrará la pantalla de la figura 6.9.

Desde la pantalla anterior podrá iniciar sesión en una cuenta previamente creada y recuperar su contraseña, en caso de haberla olvidado, mediante dos pasos:

1. Introduciendo su correo en el formulario de la pantalla 6.10 para que, si la dirección de correo está registrada, la aplicación le envíe un enlace de recuperación de contraseña.
2. El usuario ha de pinchar en el enlace del correo mandado e introducir la nueva contraseña en el formulario de la pantalla mostrada en la figura 6.11.

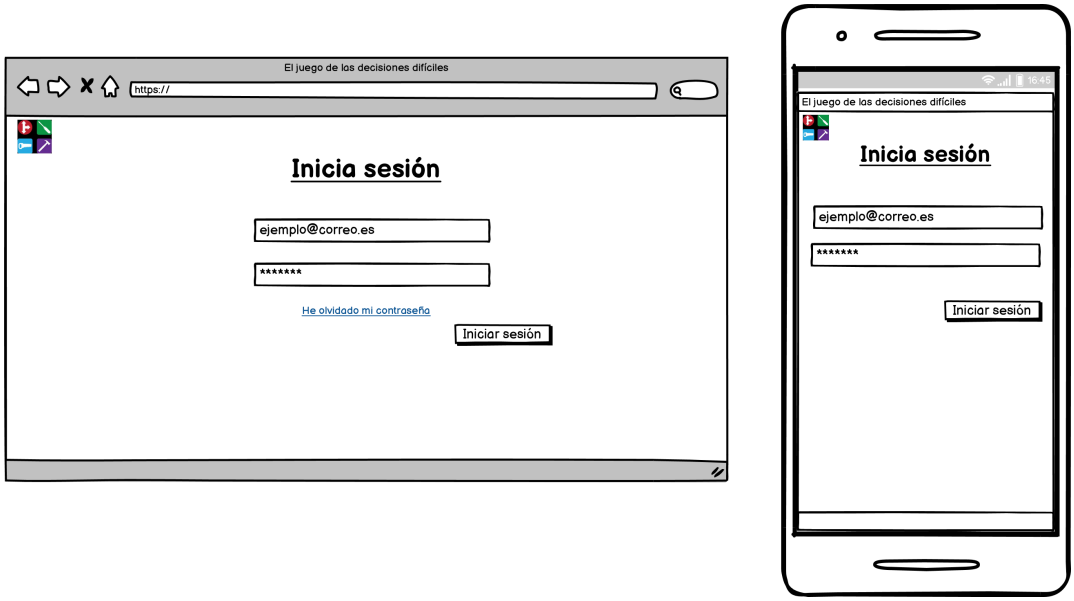


**Figura 6.9. Boceto de la vista para iniciar sesión**

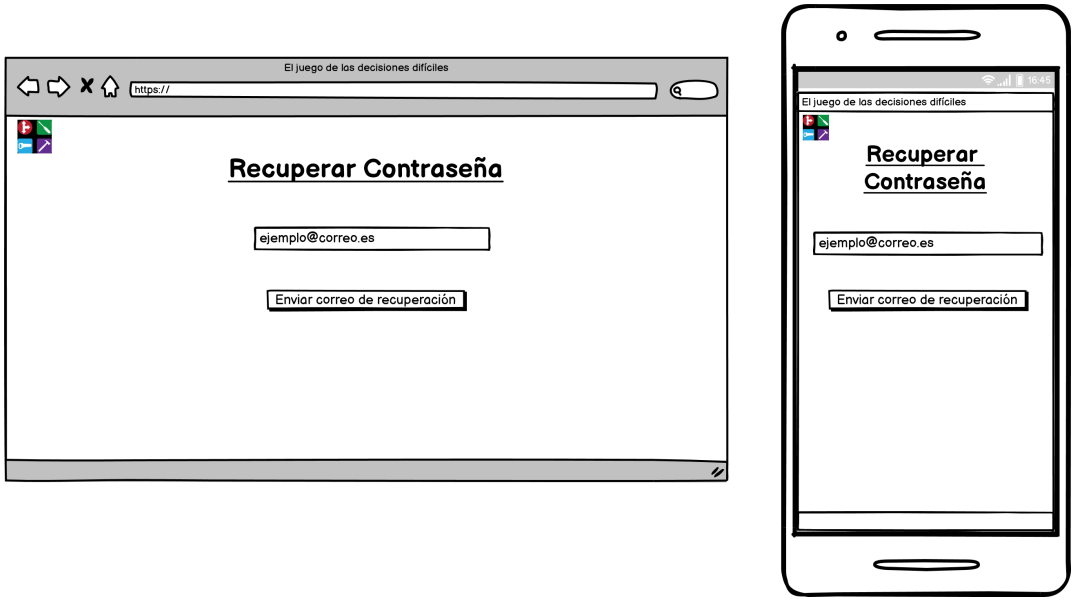


**Figura 6.10. Boceto de la vista del primer paso para recuperar la contraseña**

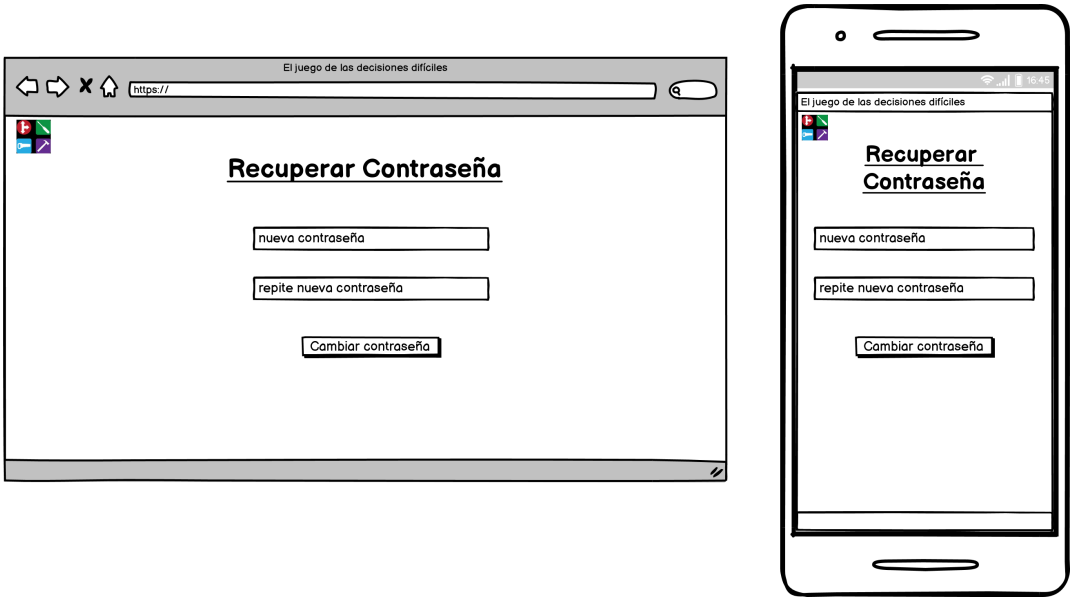


**Figura 6.11. Boceto de la vista del segundo paso para recuperar la contraseña**

Una vez se haya creado una partida el moderador que la haya creado verá la vista de la figura 6.12. Desde esta pantalla podrá copiar el enlace desde el cual los jugadores podrán acceder a la partida, ver cuantos jugadores se han unido e iniciar la partida. Los jugadores que accedan al enlace dicho anteriormente verán la pantalla mostrada en el boceto 6.13


**Figura 6.12. Boceto de la vista para iniciar una partida creada**


**Figura 6.13. Boceto de la vista para unirse a una partida como jugador**

Existen 2 vistas para representar el caso de uso "Realizar jugada", ambas detalladas en la figura 6.14. Una interfaz es para los jugadores, los cuales verán su puntuación y las flechas para mover su ficha, y otra para el moderador, donde no verá ningún marcador ni flechas pero si verá los detalles de la partida y tendrá la opción de pausar y terminar la partida. Destacar también de esta interfaz que:

- Las fichas son de distinto color y forma para facilitar a los usuarios con daltonismo el uso de la aplicación.
- Si se pasa el ratón por encima de las fichas se muestra el alias introducido por el jugador correspondiente.
- El botón con una interrogación que se ve abajo a izquierda despliega un pop up con las instrucciones y reglas del juego.
- La pantalla del ordenador del jugador muestra el marcador inicial mientras que la pantalla del teléfono móvil muestra otro posible marcador según las variaciones que se hayan añadido a la partida.

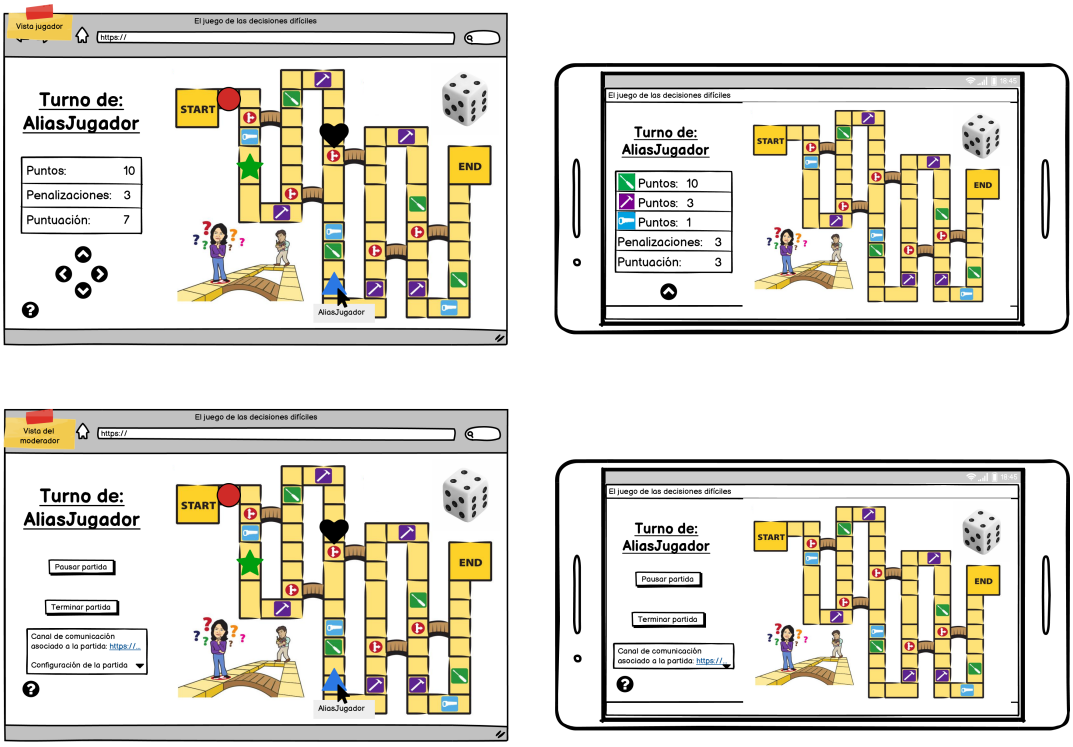


**Figura 6.14. Boceto de la vista en la que se realiza las jugadas**

Se ha creado una vista para el caso de uso "Monitorizar partida", figura 6.15, desde donde se realiza el caso de uso "Añadir ronda" introduciendo un número y dando al botón correspondiente. Sin embargo, para el caso de uso "Añadir variación" si que se ha creado una interfaz, figura 6.16, para dar más información sobre las variaciones que se pueden añadir desplegando dicha información desde la flecha de la derecha. Este boceto da la posibilidad de incluir más de una variación a la vez.

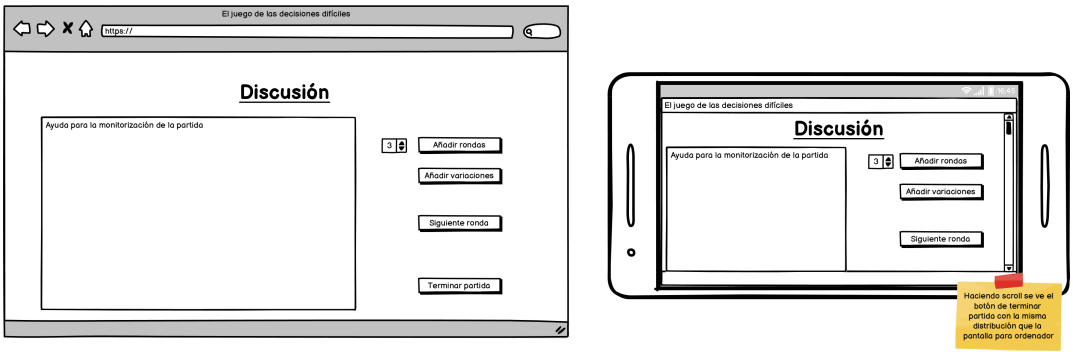


**Figura 6.15. Boceto de la vista para monitorizar una partida**

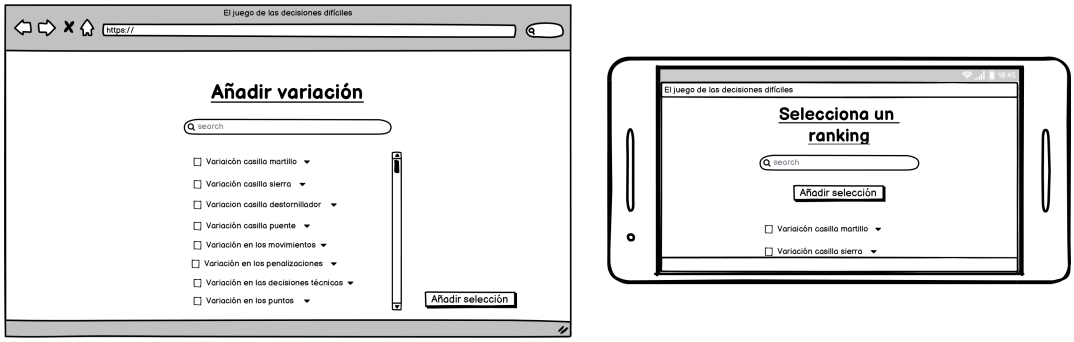


**Figura 6.16. Boceto de la vista para añadir variaciones**

Cuando se termine la partida se mostrará la pantalla de la figura 6.17 la cual tiene dos apariencias:

- La del jugador, que permite unirse a un ranking o ver los que ya existen.
- La del moderador, la cual permite crear un ranking nuevo y reiniciar la partida.


**Figura 6.17. Boceto de la vista para añadir variaciones**

Por último, quedan la vista que permite crear un nuevo ranking a los moderadores, figura 6.18, y la que permite a los jugadores unirse a uno o más rankings, figura 6.19. Esta última es muy parecida a la figura 6.7 en la que se muestran todos los rankings por tanto, estas dos vistas se crearán a través de un mismo componente.


**Figura 6.18. Boceto de la vista para añadir variaciones**

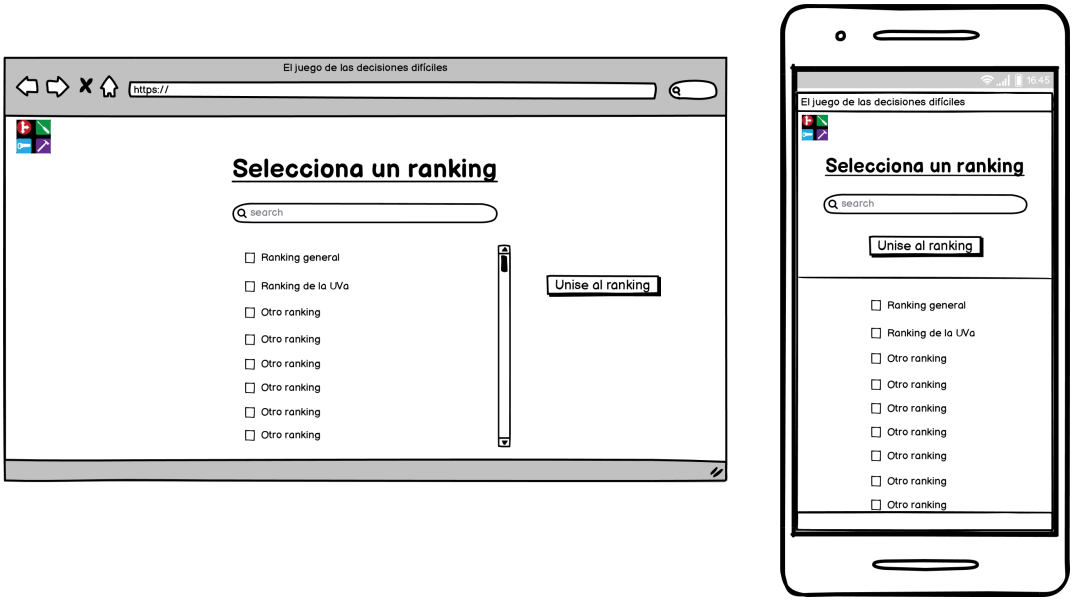


**Figura 6.19. Boceto de la vista para añadir variaciones**

## 6.3. Patrones arquitectónicos

En esta sección se especifican los patrones arquitectónicos que se han decidido utilizar para el diseño de la arquitectura del proyecto, en concreto se habla del patrón cliente-servidor y del Modelo-Vista-Controlador.

### 6.3.1. Patrón cliente-servidor

Con este patrón se intenta equilibrar la carga de procesamiento entre la máquina cliente y la del servidor delegando en cada una de ellas distintas funcionalidades de la implementación para facilitar el mantenimiento y la escalabilidad del mismo.

En este caso, se ha decidido utilizar el patrón cliente - servidor con "cliente grueso" o *fat client* ya que el peso de la lógica de la aplicación recae en él.

Destacar que gracias a la decisión de utilizar Firebase Realtime como base de datos ha sido muy sencillo implementar un cliente-servidor de tres niveles, siendo el último la nube en la que se almacenan los datos de la aplicación [30] [22].

### 6.3.2. Patrón MVC

Se ha utilizado el patrón Modelo-Vista-Controlador para que las clases de la vista no tenga conocimiento directo de las del dominio y viceversa. De esta forma se desacopla los objetos del dominio de las interfaces de usuario, se evita que los cambios de la interfaz afecten al dominio y se mejora la reusabilidad de ambos. [30]

La vista se ha implementado con el *framework* AngularCLI que hace de contenedor en el diagrama de estructura compuesta [21] mostrado en la imagen **??** ya que implementa de nuevo el patrón MVC, desplegando los 3 elementos del patrón:

- **La vista**: Encargada de recoger las peticiones del usuario y trasmitirla a su controlador.
- **El controlador**: Controla el flujo entre vista y dominio y actúa como facha ante el servidor.
- **El dominio**: Contiene la información de los datos del sistema y su lógica de negocio.

La capa controlador y modelo están implementadas en Java 11 con el *framework Spring* y se encargan de recuperar las peticiones del cliente y la comunicación con la base de datos, respectivamente.

## 6.4. Patrones de diseño

A continuación se detallan los patrones utilizados en el diseño de la implementación de la aplicación. Se habla del patrón DAO, singleton, inversión de control e inyección de dependencia, estos dos últimos impuestos por el uso del *framework Spring*.

### 6.4.1. Patrón DAO

Para el acceso a la base de datos se ha utilizado el patrón DAO: *Data Access Object*, así se consigue reducir la complejidad del código, centralizar el acceso a datos y facilitar futuras migraciones de los componentes de datos.

Cada DAO se encarga de gestionar las operaciones contra la base de datos para un objeto del dominio concreto para ello ha de implementar una interfaz con las operaciones CRUD y el resto de consultas necesarias serán implementas por el desarrollador en la clase DAO correspondiente [30] [20].

En este caso se ha decidido utilizar un proyecto *open source* que ha desarrollado una interfaz para una base de datos Firebase Realtime usando el *framework Spring* por lo que los DAOs de este proyecto lo único que hacen es encapsular estos métodos ya creados en el proyecto heredado [1].

### 6.4.2. Patrón Inversión de Control

"No nos llame, nosotros le llamaremos" es el principio en el que se base el patrón de inversión de control el cual consiste en que otro agente externo controle el flujo de la aplicación.[27]

En este caso el agente externo que controla el flujo del proyecto es el *framework Spring* que utiliza el patrón de Inyección de dependencias, sección 6.4.3, para ello.

### 6.4.3. Patrón Inyección de Dependencias

Como se ha dicho en el apartado anterior, para implementar el patrón de inversión de control el *framework Spring* utiliza el patrón de inyección de dependencias (DI) el cual consiste en crear una capa de abstracción entre las clases de bajo nivel y las clases alto nivel para que las clases de alto nivel, más complejas que las otras, no utilicen directamente las de bajo nivel y hereden de una interfaz abstracta permitiendo una mayor flexibilidad en código y facilitando la refactorización de las clases, si fuera necesario [17].

El *framework Spring* permite implementar el patrón DI mediante anotaciones y la definición de Beans, así el core container de *Spring* se encarga de la creación y el control de la vida de los recursos pedidos mediante dichas anotaciones.

### 6.4.4. Patrón Singleton

El patrón *Singleton* involucra a una única clase la cual se asegura de que no haya más de una instancia de si misma. Para ello, se instancia a si misma y, al mismo tiempo, proporciona un punto de acceso global a esa instancia pudiendo así, utilizar la misma instancia de la clase desde cualquier punto [18].

En nuestro proyecto se utilizará para la clase que inicializa la base de datos y en los servicios de la máquina cliente, en todos aquellos que utilicen la anotación *@Injectable*.

## 6.5. Arquitectura del sistema

En esta sección se verá la arquitectura general propuesta para el sistema y para su despliegue. Además se detallará la arquitectura del lado cliente y del servidor.

A continuación se verá la arquitectura lógica propuesta de todo el sistema y la conexión con la base de datos, además de la planteada para el despliegue de la aplicación.

### 6.5.1. Arquitectura lógica

Como ya se ha dicho se ha decidido utilizar una arquitectura Cliente-Servidor para este proyecto donde la lógica del proyecto recae enteramente en el cliente ya que se ha decidido implementar una arquitectura Cliente-Servidor con el cliente "grueso" (*fat*).

En la figura 6.20 se puede observar que la máquina cliente implementa el patrón Modelo-Vista-Controlador, para gestionar las paginas de la aplicación y el enrutamiento de las mismas, como parte del componente vista del patrón MVC que abarca a las dos máquinas.

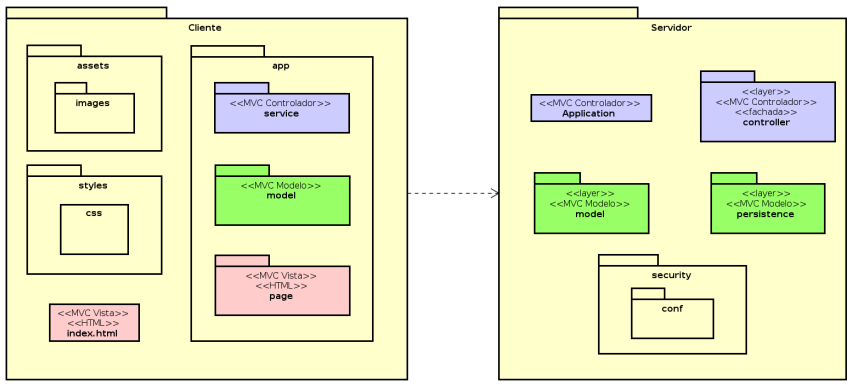


**Figura 6.20. Arquitectura lógica**

### 6.5.2. Arquitectura física: Despliegue

En la figura 6.21 se muestra la arquitectura propuesta para el despliegue.

Como se puede observar es una arquitectura Cliente-Servidor de tres niveles donde el cliente será un navegador web ejecutado desde un ordenador, tablet o teléfono móvil. Este realizará peticiones a nuestra aplicación, la cual se ejecuta sobre un servidor *Tomcat*, facilitado por el *framework Spring*, que a su vez hará peticiones a la base de datos, *Firebase Realtime*, alojada en una nube externa proporcionada por Google.

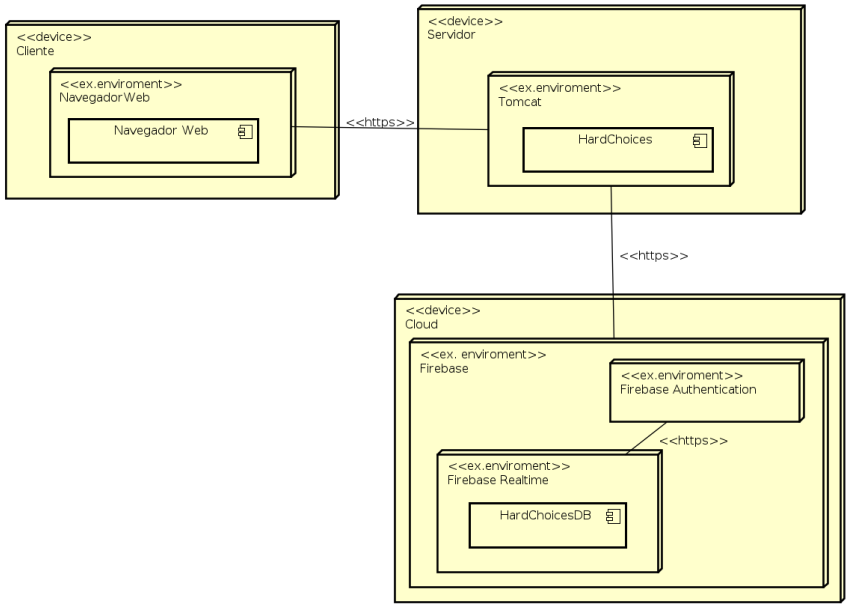


**Figura 6.21. Modelo de despliegue.**

### 6.5.3. Arquitectura del cliente

De la arquitectura cliente, mostrada en la figura 6.22, destacar la estructura MVC dicha anteriormente además de la distribución dada por Angular CLI que obliga a tener una primera clase *index.html* la cual es llamada al inicio de la ejecución y sirve como contenedor para el resto de vistas utilizadas en el proyecto. Estas vistas, junto a sus controladores y modelos, se encuentran en el paquete *app*, también impuesto por Angular.

En la sección 6.6.1 se explica más detalladamente los artefactos incluidos en cada paquete y su funcionalidad.

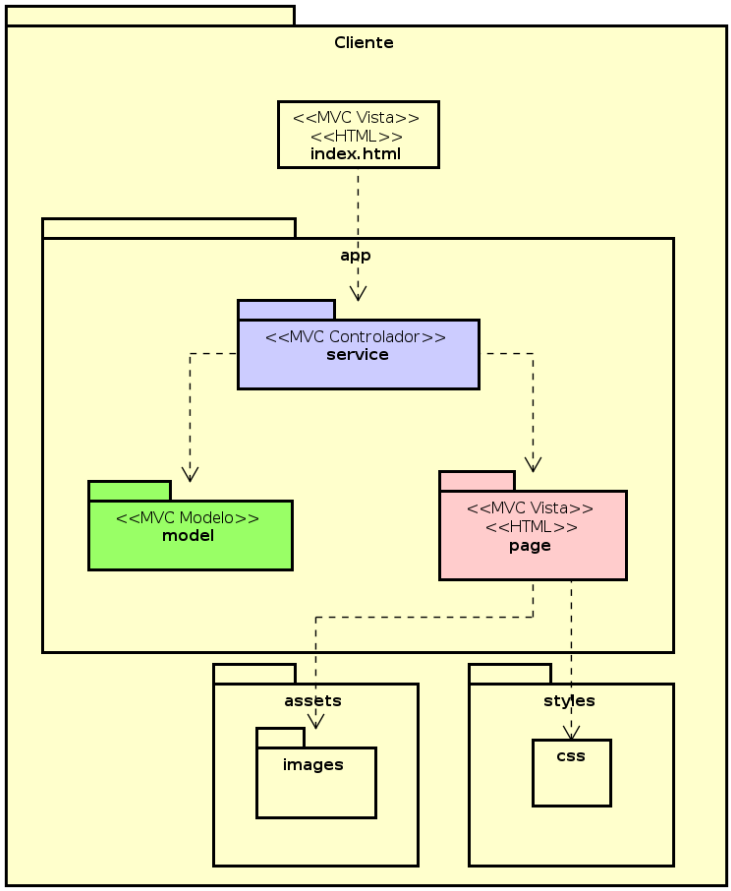


**Figura 6.22. Arquitectura de la máquina cliente**

### 6.5.4. Arquitectura del servidor

En la figura 6.23 se muestra una estructura parecida a la explicada en el apartado anterior. En este caso es el *framework Spring* el que obliga a tener una primera clase a la que se invoca en primer lugar y se encarga de despertar a las clases anotadas como configuración y *Bean* para ejecutarlas.

En la sección 6.6.2 se explica más detalladamente los paquetes, las clases y su funcionalidad.

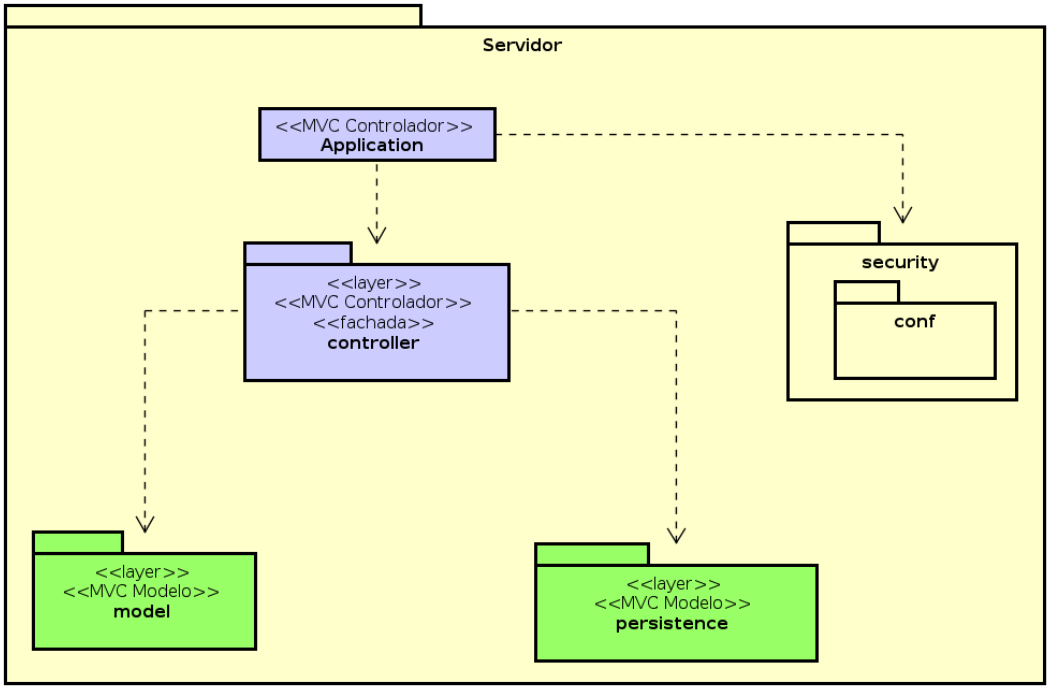


**Figura 6.23. Arquitectura de la máquina servidor**

## 6.6. Diseño detallado

A continuación se verá con más detalle el diseño explicado en la sección anterior.

Antes de empezar señalar que el diseño del proyecto está guiado por los casos de uso, es por ello que tanto en el lado cliente como en el del servidor existe un controlador por cada caso de uso los cuales se comunican entre si para el paso de peticiones de un lado a otro.

### 6.6.1. Diseño detallado cliente

Como se puede ver, en la imagen 6.24, en el lado del cliente se ha diseñado un componente por cada interfaz de usuario especificada en la sección 6.2.1. Las clases para la presentación de la vista se encuentran en el paquete *page*, en este paquete existe un sub-paquete por componente que contiene un archivo *.html* que compone la vista y una clase TypeScript que contienen la lógica de la vista.

Cada componente dicho en el apartado anterior utiliza una clase del paquete *service* que

contiene todos los métodos para las peticiones al servidor y una común a todos, *dataService.ts*, que contiene los datos comunes de la sesión. Todos los servicios heredan de un servició común *restService.ts* que implementa las funciones para las peticiones a la API REST del proyecto.

Por último, el paquete *model* contiene las interfaces de los objetos del dominio. Este paquete es similar al paquete *model* del servidor para facilitar la propiedad *serializable* y, con ello, la lectura de los ficheros JSON en las llamadas entre el cliente y el servidor.

### 6.6.2. Diseño detallado servidor

La imagen 6.25 muestra la arquitectura detalla del lado del servidor. Como se puede observar es similar a la del lado cliente explicada en el apartado anterior.

Destacar que, como en el caso del Cliente, las clases del controlador son independientes entre si, gracias a que el *framework Spring* utiliza anotaciones que especifican en que ruta debe actuar cada controlador.

El paquete de persistencia está formado por un paquete de *accesoBD*, que contiene una clase singleton que genera una única instancia para acceder a la base de datos, y un paquete *dao*, que contienen todas las clases necesarias para acceder a los objetos de la base de datos. El diseño del almacenamiento persistente se detalla en la sección 6.8.

## 6.7. Diseño de la API REST

En la imagen 6.26 se muestra el diseño de la API REST pensada para este proyecto.

## 6.8. Diseño del almacenamiento persistente

La base de datos utilizada, *Firebase Realtime*, no es relacional, como ya se explicó en la sección 5.3.2, por lo que se ha realizado un diseño de la una base de datos no-SQL. Este diseño se muestra en la imagen 6.27

Como a lo largo de las pantallas del proyecto se muestra por separado los puntos, las penalizaciones y el resultado total de puntos de cada jugador se ha decidido guardar en la clase *Jugador* los atributos *puntos* y las *penalizaciones* y calcular en el sistema la puntuación total como *puntos - penalizaciones*.

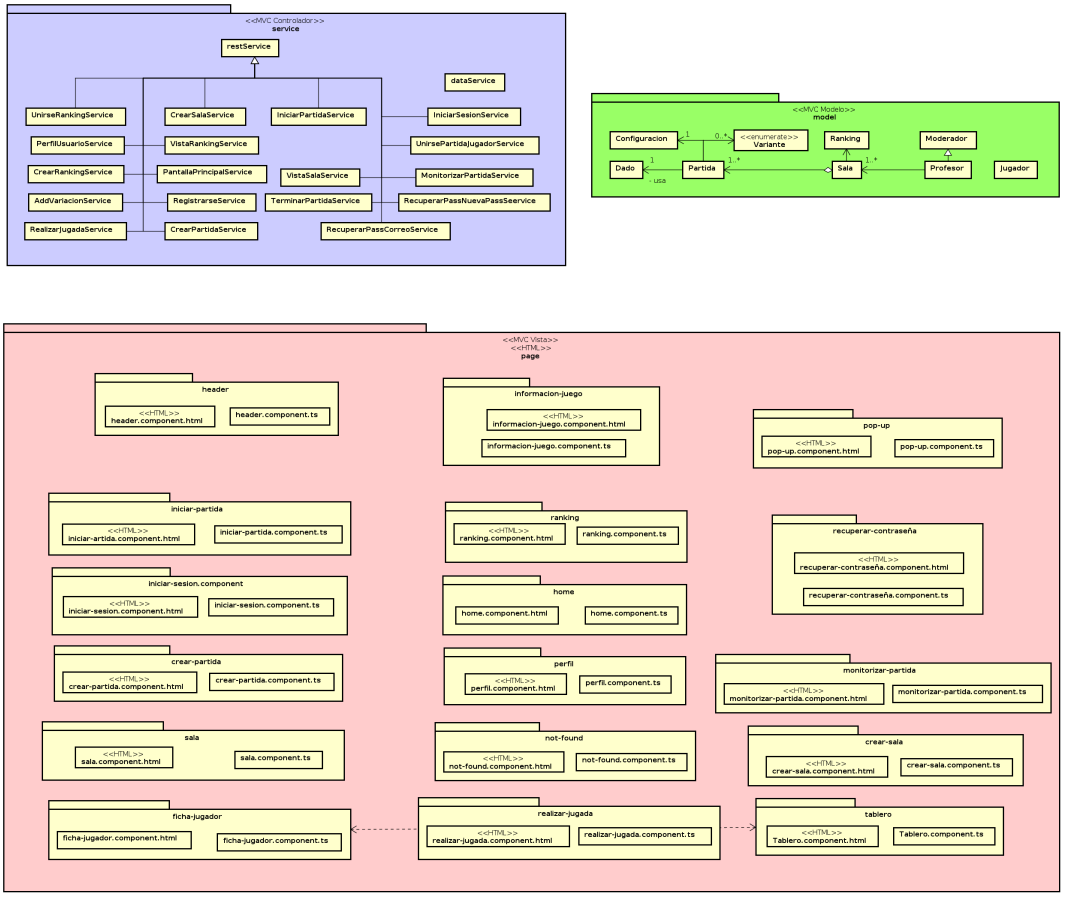


**Figura 6.24. Arquitectura detallada del cliente**

El atributo *ficha* de la clase *Jugador* y el atributo *configuración* de la clase *Partida* son objetos *JSON* con los atributos de cada clase del domino.

## 6.9. Diseño basado en componentes

En la imagen 6.28 se muestra el diagrama de componentes angular que forman la aplicación desarrollada.

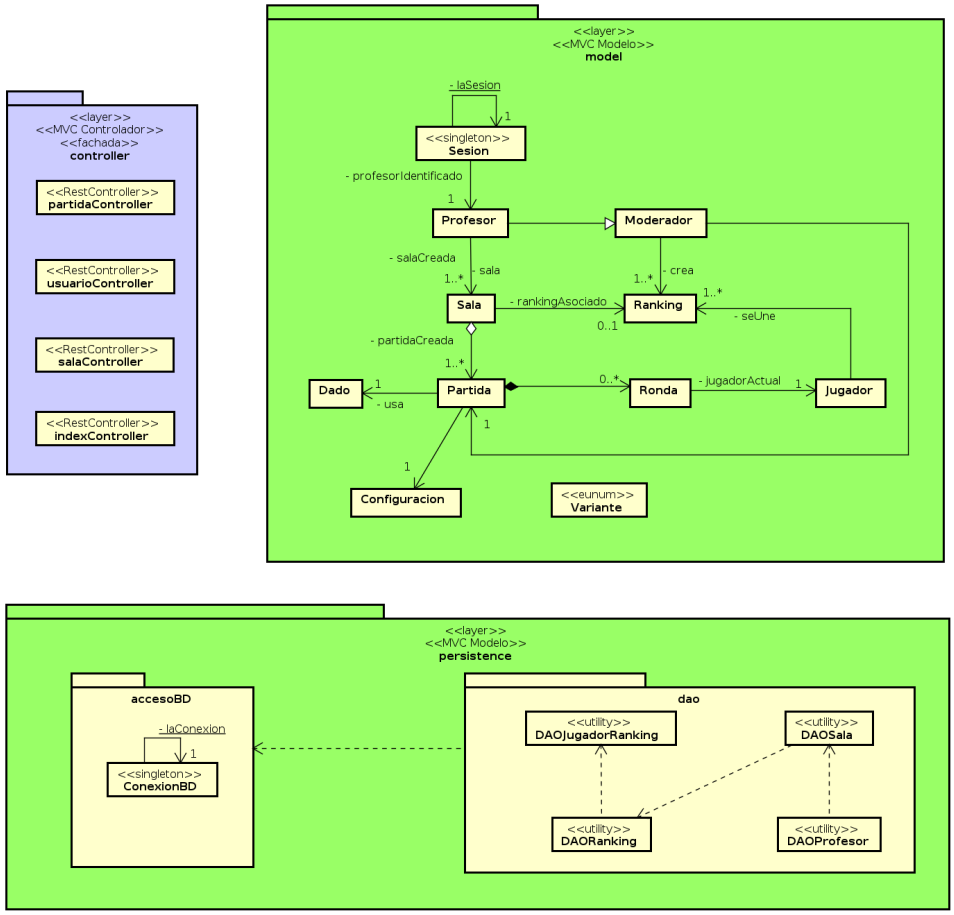


**Figura 6.25. Arquitectura detallada del servidor**

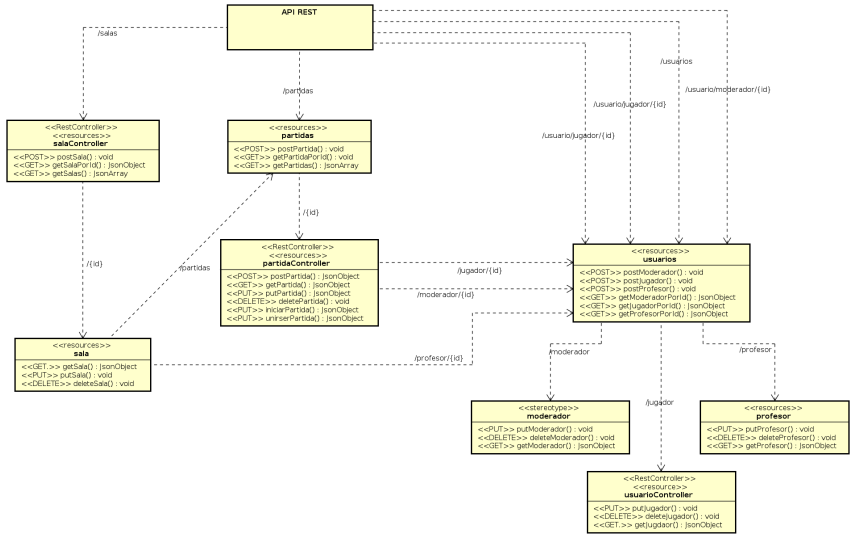


**Figura 6.26. Diseño del API REST**

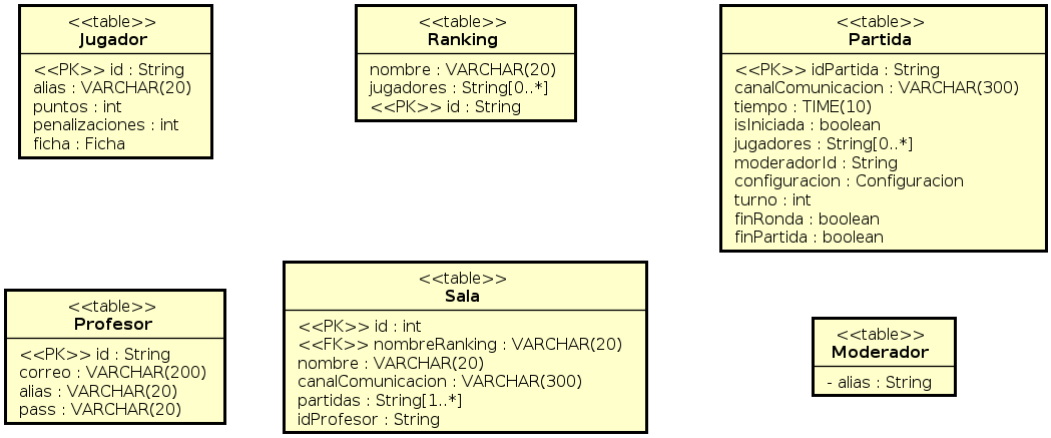


**Figura 6.27. Diseño no relacional del almacenamiento persistente**

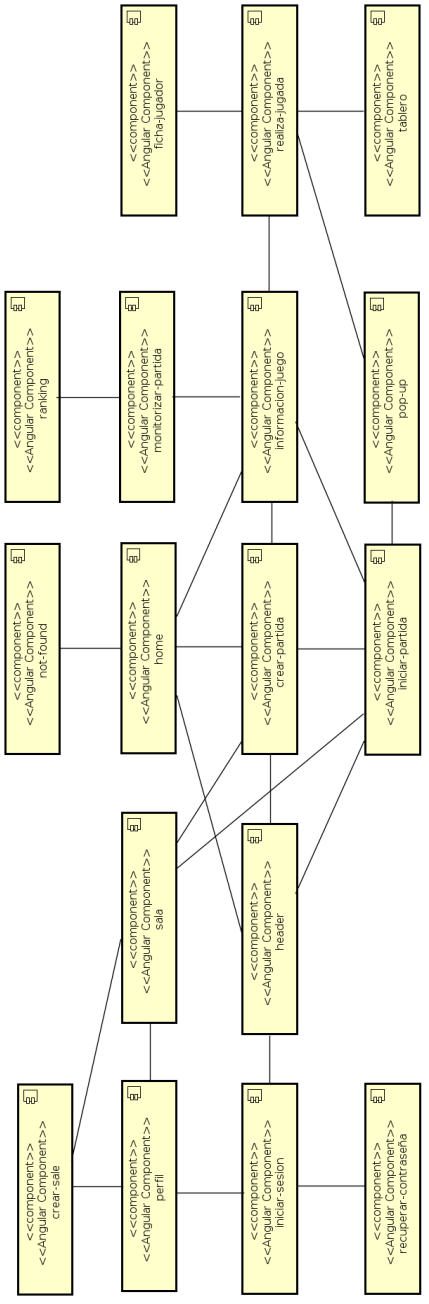


**Figura 6.28. Diagrama de componentes**

# Capítulo 7

# Implementación y pruebas

En este capítulo se comenta todo lo relacionado con la implementación: herramientas utilizadas, organización del código, cambios respecto al diseño y la implementación de las interfaces de usuario, y los casos de prueba que se han realizado, sus resultados y las soluciones a los fallos que se presentaron durante los procesos de prueba.

## 7.1. Implementación

A continuación se detalla todo lo realizado para esta versión final del proyecto.

### 7.1.1. Entornos de desarrollo

Como entorno de desarrollo se ha utilizado IDE Eclipse para la implementación del *backend* con *Spring Boot* y *Java* y *Visual Studio Code* para el *frontend* con *Angular CLI*.

Se ha decidido utilizar dos entornos de desarrollo siguiendo las recomendaciones de los profesores del grado cursado y de las guías oficiales de los dos frameworks utilizados. Estas recomendaciones se basan en que cada herramienta tiene unos *plugins* específicos que facilitan la escritura del código en un lenguaje concreto.

### 7.1.2. Implementación de la Base de Datos

Como ya se ha dicho, se ha utilizado como SGBD *Firebase Realtime*, una base de *Google* no relacional.

El diseño de esta ha sido explicado en la sección 6.8 y los métodos CRUD han sido heredados de un proyecto *OpenSource* ajeno encontrado en *GitHub* [1]. Se tomó la decisión de utilizar este proyecto para ahorrar tiempo en la implementación pero el creador comentó en el mismo que ha dejado de mantenerlo y con el uso del mismo se ha visto que no es tan potente como se esperaba por lo que se cambiará en futuras versiones del proyecto.

### 7.1.3. Control de versiones

Para el control de versiones se ha utilizado Git, en concreto el repositorio ofrecido por la escuela: GitLab, el cual se ha utilizado durante todo el periodo de estudios del grado y se ha adquirido una gran habilidad. A parte de esta razón, se decidió utilizar este repositorio por la facilidad de seguimiento que aporta a la tutora.

La URL del proyecto es:

`https://gitlab.inf.uva.es/rebhern/tfg-RebecaHernando`

### 7.1.4. Organización del código

A continuación se detalla la estructuras del repositorio y de los de directorios que tiene el proyecto tanto la parte del cliente como la del servidor.

#### Organización del repositorio

Antes de explicar la organización del código indicar que el repositorio Git en el que se encuentra este proyecto tiene en la raiz dos directorios y dos archivos:

- Directorio *AnalisisDisenno* el cual contiene un archivo *.astah* con los diagramas de la aplicación que se muestran en esta memoria y otro archivo *.bmpr* con los bocetos de las interfaces de usuario producido con la aplicación *Balsamiq*.
- Directorio *hardChoicesGame* que contiene el código de la aplicación tanto del lado cliente como del lado servidor. A continuación se detallan todos los archivos de este directorio.
- Archivo *LICENSE*, la licencia software del código que se explica en la sección 7.1.6
- Archivo *README.md* con el resumen de lo que contiene el repositorio y con qué fin se ha hecho.

La decisión de esta arquitectura ha sido dirigida por los *frameworks* utilizados ya que al seguir la guía de inicialización de un proyecto web con *Spring Boot* [24] se crean automáticamente los paquetes principales. Además, ambos definen una estructura por capas por ello,

se han desarrollado las diferentes capas controladores, servicios y vistas en ambos códigos además de añadir paquetes de estilo y recursos.

Existe un único proyecto que aloja las dos partes: cliente y servidor, tienen como directorio padre la ruta *hardChoicesGame/src/*. Inmediatamente debajo de esta se encuentran los recursos comunes:

- En *hardChoicesGame/src/styles* se encuentra el fichero *styles.css* para el estilo de las interfaces de usuario.
- En *hardChoicesGame/src/assets* se encuentra una única carpeta, *images*, que contiene las imágenes del logo del juego, las casillas del tablero, en */tablero*, y las fichas para cada jugador en la partida, en */fichas*.

### Código del cliente

En la ruta */src/app* se encuentra el código del lado cliente del proyecto el cual está dividido en los siguiente paquetes:

- *model*, representa la capa de dominio del lado cliente.
- *page*, guarda los componente que constituyen las vistas del proyecto. Por debajo de este paquete hay una carpeta por cada componente creado que agrupo el fichero *.html* y el controlador del mismo.
- *service*, son los servicios que actúan como fachada para las peticiones REST al servidor. Hay un servicio por componente además de un archivo *data.service.ts* que almacena las instancias del dominio comunes a todos los componentes.

Además de estos paquetes, en la ruta *hardChoicesGame/src/app* se encuentran los archivos *app-routing.module.ts*, *app.component.html*, *app.component.ts* y *app.module.ts* generados automáticamente por Angular y que sirven como punto de entrada en el despliegue del cliente y almacenan la información de enrutamiento y los componentes, propios y ajenos, utilizados.

### Código del servidor

En */src/main/java/es/uva/inf/tfg/rebhern/hardChoicesGame* se encuentra el código del servidor del proyecto el cual está dividido en los siguiente paquetes:

- *controller*, con todos los controladores REST que actúan de fachada en este lado.
- *model*, como todas las entidades del dominio necesarias.
- *persistence/accesoBD*, con la instancia que inicializa la conexión con la base de datos.

- *persistence/dao*, agrupa las clases DAO que actuan de fachada entre servidor y la base de datos.

Como en el caso del cliente, en la ruta */src/main/java/es/uva/inf/tfg/rebhern/hardChoicesGame* hay un archivo,*HardChoicesGameApplication.java*, fuera del resto de paquetes, que sirve como punto de arranque en el despliegue del servidor.

Por último, en la raíz de este directorio, en */src/main*, se encuentra otro paquete, *resources*, con un archivo de configuración para la inicialización de la base de datos junto a *Srping Boot*.

### 7.1.5. Interfaces de usuario

Se ha decidido hacer un proyecto con interfaces adaptativas para facilitar el uso del mismo en cualquier dispositivo independientemente del tamaño de su pantalla. Por esto se decidió utilizar Angular CLI, en lugar de Angular JS, ya que este *framework* fue diseñado pensado en los diseños *responsive*. Además, se ha utilizado la *Bootstrap* y CSS.

Antes de explicar las vistas, destacar que se han creado cuatros componente comunes a la mayoría de las vistas:

- **header**: conforma la barra superior con el icono del juego y los botones de "iniciar sesión" y "registrarse".
- **informacion-juego**: un desplegable con la información que se ha creído necesaria sobre el juego *Hard Choices*.
- **pop-up**: un vista emergente que muestra un *spiner* y un mensaje dado.
- **not-found**: pantalla que muestra un mensaje de error 400 y un botón para volver al inicio.

Se han creado cinco vistas:

1. **Home**: Es la pantalla de inicio formada por el componente *home*, *header* y *informacion-juego*.
2. **Crear partida**: Vista en la que el moderador puede crear una partida, está formada por el componente *crear-partida*,*header* y *informacion-juego*.
3. **Iniciar partida**: Esta vista es común para el CU iniciar partida, actor moderador, y para el CU unirse a la partida, actor jugador, dependiendo del usuario se muestra una interfaz u otra. Está compuesta por los componentes *iniciar-partida*,*header*, *informacion-juego* y *pop-up*.

4. **Juego**: Pantalla en la que se puede realizar las jugadas de la partida, está compuesta por los componentes *juego*, *tablero*, *informacion-juego* y *pop-up*.
5. **Sesión informativa**: Es la vista que aparece cuando se terminan las rondas para que se pueda discutir sobre la deuda técnica, muestra las puntuaciones de los jugadores y permite añadir rondas y continuar o terminar la partida. La componen *monitorizar-partida*,*header*, *informacion-juego* y *pop-up*.

Se ha intentado hacer vistas adaptativas pero no se ha conseguido que el tablero lo fuera.

### 7.1.6. Licencia software

El código implementado y la documentación cuenta con una licencia software libre BSD de 3 cláusulas que permite el uso, la distribución y modificación del código y documentación del proyecto siempre y cuando se mencione a la autora del mismo. Además, la exime de cualquier tipo de responsabilidad e impide que se use el nombre de la misma sin su permiso.

Se ha decidido elegir esta licencia para facilitar la implementación de futuras versiones del juego y así asegurar que sea de utilidad para el uso docente.

La licencia se ha asignado al proyecto creado en el repositorio GitLab de la Escuela de Ingeniería Informática de la Universidad de Valladolid. Puede encontrarse el enlace en el anexo B B. A continuación se muestra una copia de la misma:

BSD 3-Clause License:

Copyright 2021, Hernando Brecht, Rebeca

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

## 7.2. Plan de pruebas y evaluación

En esta sección se recoge los casos de prueba realizados durante la implementación del proyecto y sus resultados.

### 7.2.1. Casos de prueba

A continuación se describen los casos de prueba que se han realizado para dar por finalizado cada caso de uso implementando. En la sección 7.2.2, se detallan los problemas que han surgido en la realización de cada caso de prueba y las soluciones aportadas.

#### Crear partida

| **CP-01** | **Crear partida valores predeterminados** |
|---|---|
| Descripción | Crear una partida. |
| Escenario | Vista "Home". |
| Esquema del escenario | Se indica crear partida.<br>Se introduce 2 rondas.<br>Sin tiempo máximo.<br>Sin canal de comunicación. |
| Resultado esperado | Se muestre la vista "Iniciar Partida" con la configuración introducida y el código de la partida. |

**Tabla 7.1. Crear partida valores predeterminados**

| **CP-02** | **Crear partida con tiempo** |
|---|---|
| Descripción | Crear una partida introduciendo la opción de tiempo. |
| Escenario | Vista “Home”. |
| Entrada | Se indica crear partida.<br>Se introduce 2 rondas.<br>Se introduce 1h 30min de tiempo máximo.<br>Sin canal de comunicación. |
| Resultado esperado | Se muestre la vista ”Iniciar Partida” con la configuración introducida y el código de la partida. |

**Tabla 7.2. Crear partida con tiempo**

| **CP-03** | **Crear partida más de dos rondas** |
|---|---|
| Descripción | Crear una partida introduciendo más rondas que las mínimas. |
| Escenario | Vista “Home”. |
| Entrada | Se indica crear partida.<br>Se introduce 3 rondas.<br>Sin tiempo máximo.<br>Sin canal de comunicación. |
| Resultado esperado | Se muestre la vista ”Iniciar Partida” con la configuración introducida y el código de la partida. |

**Tabla 7.3. Crear partida más de dos rondas**

| **CP-04** | **Crear partida con canal de comunicación** |
|---|---|
| Descripción | Crear una partida introduciendo la opción de enlace de comunicación. |
| Escenario | Vista “Home”. |
| Entrada | Se indica crear partida.<br>Se introduce 2 rondas.<br>Sin tiempo máximo.<br>Se introuduce un canal de comunicación. |
| Resultado esperado | Se muestre la vista ”Iniciar Partida” con la configuración introducida y el código de la partida. |

**Tabla 7.4. Crear partida con canal de comunicación**

**Unirse a la partida**

| CP-05 | Unirse a la partida valido |
|---|---|
| Descripción | Los jugadores se unen a la partida a través de un código dado. |
| Escenario | Vista "IniciarPartida"del jugador. |
| Entrada | Código de partida valido. |
| Resultado esperado | Se muestra un pop up de espera en la vista del jugador.<br>Se muestra que un jugador más se ha unido en la pantalla del moderador. |

**Tabla 7.5. Unirse a la partida valido**

| CP-06 | Unirse a la partida superando el máximo de jugadores |
|---|---|
| Descripción | El máximo de jugadores esperan a que se inicie la partida y un jugador más solicita unirse a la partida. |
| Escenario | Vista "IniciarPartida" del jugador. |
| Entrada | 5 jugadores diferentes introducen un código de partida valido. |
| Resultado esperado | Se muestra un pop up de espera en la vista de los 4 primeros jugadores.<br>No se permite que se una el 5º jugador y se muestra en su pantalla un mensaje de error.<br>El contador de jugadores de la vista del moderador aumenta de 1 en 1 mientras se unen los jugadores. |

**Tabla 7.6. Unirse a la partida superando el máximo de jugadores**

| CP-07 | Unirse a la partida enlace no válido |
|---|---|
| Descripción | Un jugador quiere unirse a una partida con un código no válido . |
| Escenario | Vista "IniciarPartida"del jugador. |
| Entrada | Código de partida no registrado en la base de datos. |
| Resultado esperado | No se permite que unirse al jugador y se muestra en su pantalla un mensaje de error. |

**Tabla 7.7. Unirse a la partida enlace no válido**

| CP-08 | Unirse a una partida iniciada |
|---|---|
| Descripción | Un jugador quiere unirse a una partida ya iniciada. |
| Escenario | Vista "IniciarPartida" del jugador. |
| Entrada | Código de partida valido de una partida ya iniciada. |
| Resultado esperado | No se permite que unirse al jugador y se muestra en su pantalla un mensaje de error. |

**Tabla 7.8. Unirse a una partida iniciada**

### Iniciar partida

| CP-09 | Iniciar partida con el mínimo de jugadores |
|---|---|
| Descripción | El moderador inicia la partida creada cuando 2 jugadores se han unido a la partida. |
| Escenario | Vista “IniciarPartida” del moderador. |
| Entrada | Seleccionar la opción “Iniciar partida”. |
| Resultado esperado | Se muestra la pantalla “Juego” al moderador y los jugadores. |

**Tabla 7.9. Iniciar partida con el mínimo de jugadores**

| CP-10 | Iniciar partida sin el mínimo de jugadores |
|---|---|
| Descripción | El moderador inicia la partida creada cuando menos de 2 jugadores se han unido a la partida |
| Escenario | Vista “IniciarPartida” del moderador |
| Entrada | Seleccionar la opción “Iniciar partida”. |
| Resultado esperado | No se inicia la partida.<br>Se muestra un mensaje de error. |

**Tabla 7.10. Iniciar partida sin el mínimo de jugadores**

### Realizar jugada

Se ha modificado el caso de uso Realizar jugada: en lugar de esperar a que todos los jugadores excepto uno, la ronda termina cuando el primer jugador llega al final del tablero. Se mantiene los puntos que obtiene el primer jugador por tanto, se sumará a su tablero siente puntos.

Se puede ver el caso de uso original en la tabla 3.10.

| CP-11 | Realizar jugada correctamente |
|---|---|
| Descripción | El jugador realiza una turno de partida correctamente |
| Escenario | Vista ”Juego”del jugador |
| Entrada | Es el turno del jugador<br>Lanza el dado<br>Mueve la ficha a una casilla permitida<br>Repite hasta que no le quede movimientos<br>Recoge los puntos y penalizaciones |
| Resultado esperado | La ficha avanza hasta la casilla esperada.<br>El marcador muestras los puntos y las penalizaciones correspondientes. |

**Tabla 7.11. Realizar jugada correctamente**

| CP-12 | Realizar jugada cuando no es el turno del jugador |
|---|---|
| Descripción | El jugador intenta realiza una turno de partida cuando no es su turno |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador intenta lanzar el dado.<br>El jugador intenta mover la ficha. |
| Resultado esperado | Los botones están bloqueados y no puede realizar la acción. |

**Tabla 7.12. Realizar jugada cuando no es el turno del jugador**

| CP-13 | Realizar jugada sin lanzar el dado |
|---|---|
| Descripción | El jugador intenta realiza un turno de partida sin lanzar el dado |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador pulsa un botón de flecha para mover la ficha |
| Resultado esperado | Se muestra un mensaje de error.<br>La ficha no se mueve. |

**Tabla 7.13. Realizar jugada sin lanzar el dado**

| CP-14 | Realizar jugada con un movimiento no válido |
|---|---|
| Descripción | El jugador intenta realiza en un turno de partida un movimiento no válido |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador lanza el dado.<br>Pulsa un botón de flecha para moverse a un lugar sin casilla. |
| Resultado esperado | Se muestra un mensaje de error.<br>La ficha no se mueve. |

**Tabla 7.14. Realizar jugada con un movimiento no válido**

| CP-15 | Realizar jugada pasando por una casilla puente |
|---|---|
| Descripción | El jugador realiza un turno de partida pasando por, al menos, una casilla de puente |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador lanza el dado.<br>Mueve la ficha a una casilla válida.<br>Pasa por una casilla puente. |
| Resultado esperado | Se suma una penalización al marcador del jugador.<br>Se actualizan los puntos. |

**Tabla 7.15. Realizar jugada pasando por una casilla puente**

| CP-17 | Realizar jugada llegando a la casilla de meta |
|---|---|
| Descripción | El jugador realiza un turno de partida llegando a la meta. |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador lanza el dado.<br>Mueve la ficha a una casilla válida.<br>Pasa llega a la casilla meta. |
| Resultado esperado | Se suma tres puntos al marcador del jugador.<br>Se actualizan los puntos.<br>Se finaliza la ronda y se muestra la vista de monitorizar partida. |

**Tabla 7.17. Realizar jugada llegando a la casilla de meta**

| CP-18 | Las fichas se muestran en todas las pantallas. |
|---|---|
| Descripción | Los movimientos de las fichas se muestran en las pantallas de los usuarios en la partida en tiempo real. |
| Escenario | Vista "Juego"del jugador |
| Entrada | Lanza el dado.<br>Realiza un movimiento válido. |
| Resultado esperado | Se mueve la ficha.<br>Se actualiza el movimiento en las pantallas de todos los jugadores. |

**Tabla 7.18. Las fichas se muestran en todas las pantallas.**

| CP-16 | Realizar jugada pasando por una casilla de herramienta |
|---|---|
| Descripción | El jugador realiza un turno de partida pasando por, al menos, una casilla de herramienta |
| Escenario | Vista "Juego"del jugador |
| Entrada | El jugador lanza el dado.<br>Mueve la ficha a una casilla válida.<br>Pasa por una casilla de herramienta. |
| Resultado esperado | Se suma un punto al marcador del jugador.<br>Se actualizan los puntos. |

**Tabla 7.16. Realizar jugada pasando por una casilla de herramienta**

| CP-19 | Turno asignado correctamente. |
|---|---|
| Descripción | El turno del primer jugador se asigna aleatoriamente y luego rota con el mismo orden por todos los jugadores de la partida. |
| Escenario | Vista "Juego"del jugador |
| Entrada | Se inicia la partida.<br>Cada jugador realiza un turno. |
| Resultado esperado | El primer jugador es asignado aleatoriamente.<br>Se rota entre todos los jugadores de manera ordenada. |

**Tabla 7.19. Turno asignado correctamente.**

### Terminar partida/Terminar ronda

Como no se han realizado todos los casos de uso para completar este caso de uso se ha decidido cambiar y realizar "Terminar ronda". El caso de uso "Terminar partida" se puede ver en la tabla 3.17 y el nuevo caso de uso en la tabla 3.29.

| **CP-21** | **Terminar ronda mediante botón.** |
|---|---|
| Descripción | El moderador pulsa le botón "terminar ronda". |
| Escenario | Vista "Juego"del moderador |
| Entrada | Se pulsa el botón "Terminar ronda" |
| Resultado esperado | Se pausan todas las pantallas de los jugadores.<br>Se redirige a todos los usuarios a la pantalla "monitorizar partida"donde se muestran las puntuaciones de todos los jugadores. |

**Tabla 7.20. Dado aleatorio.**

| **CP-22** | **Terminar ronda mediante llegada a meta.** |
|---|---|
| Descripción | Un jugador llega a meta y se termina la ronda. |
| Escenario | Vista "Juego"del jugador |
| Entrada | Un jugador llega a meta. |
| Resultado esperado | Se pausan todas las pantallas de los jugadores.<br>Se redirige a todos los usuarios a la pantalla "monitorizar partida"donde se muestran las puntuaciones de todos los jugadores. |

**Tabla 7.21. Terminar ronda mediante llegada a meta.**

| **CP-23** | **Terminar ronda mediante botón con un turno sin terminar.** |
|---|---|
| Descripción | El moderador termina la ronda mediante el botón sin que un jugador haya agotado todos los movimientos de su turno. |
| Escenario | Vista "Juego"del moderador. |
| Entrada | Pulsa el botón "Terminar ronda". |
| Resultado esperado | Se pausan todas las pantallas de los jugadores.<br>Se redirige a todos los usuarios a la pantalla "monitorizar partida"donde se muestran las puntuaciones de todos los jugadores. |

**Tabla 7.22. Terminar ronda mediante botón con un turno sin terminar.**

### Monitorizar partida

El caso de uso original, tabla 3.5, permite añadir variaciones y rondas al final la sesión informativa pero en esta versión solo se permitirá añadir más rondas.

| CP-24 | Quedan rondas por jugar. |
|---|---|
| Descripción | Se termina la sesión informativa y quedan rondas por jugar, un usuario selecciona que quiere jugar la siguiente ronda. |
| Escenario | Vista "Monitorizar partida" |
| Entrada | Pulsa el botón "Siguiente ronda". |
| Resultado esperado | Se vuelve a la vista "Juego".<br>Todas las fichas están en la casilla de salida.<br>Cada jugador conserva los puntos acumulados en las anteriores rondas. |

**Tabla 7.23. Quedan rondas por jugar.**

| CP-25 | No quedan rondas por jugar. |
|---|---|
| Descripción | Se termina la sesión informativa y no quedan rondas por jugar, un usuario selecciona que quiere finalizar la siguientepartida. |
| Escenario | Vista "Monitorizar partida" |
| Entrada | Pulsa el botón "Terminar partida". |
| Resultado esperado | Se vuelve a la vista "Home".<br>Se eliminan todos los datos generados. |

**Tabla 7.24. No quedan rondas por jugar.**

### Añadir ronda

| CP-26 | Añadir una ronda. |
|---|---|
| Descripción | Se añade una ronda más a la partida actual. |
| Escenario | Vista "Monitorizar partida" |
| Entrada | Se indica 1 ronda.<br>Pulsa el botón .ªñadir rondas". |
| Resultado esperado | Se actualiza la partida con una ronda más.<br>Se cambia el número de rondas en las pantallas de todos los jugadores. |

**Tabla 7.25. Añadir una ronda.**

| CP-27 | Añadir una ronda entrada inválida. |
|---|---|
| Descripción | Se indica una entrada inválida para definir el número de rondas que se quieren añadir. |
| Escenario | Vista "Monitorizar partida" |
| Entrada | Se indica una entrada inválida.<br>Pulsa el botón .ªñadir rondas". |
| Resultado esperado | Se muestra un mensaje de error.<br>No se actualiza la partida. |

**Tabla 7.26. Añadir una ronda.**

### 7.2.2. Resultado de las pruebas

En esta sección se recogen los resultados de las pruebas que se han realizado, qué fallos han dado y cómo se han solucionado.

Por problemas ajenos no se han podido realizar las pruebas en el navegador Safari y como no se han encontrado ningún problema específico de un navegador la tabla 7.27 reprsenta los resultados y soluciones tanto para el navegador *Firefox* como para el navegador *Chorme*.

| Caso de prueba | Resultado | Solución |
|---|---|---|
| CP-01 | Fallo: no se muestra la configuración. | Se ha añadido el código necesario para que se muestre la configuración. |
| CP-02 | Fallo: no muestra el tiempo correctamente. | Como el tiempo no es necesario para ningún CU desarrollado se ha falseado con una *String*. |
| CP-03 | OK | - |
| CP-04 | OK | - |
| CP-05 | OK | - |
| CP-06 | OK | - |
| CP-07 | OK | - |
| CP-08 | Fallo: se puede unir a una partida iniciada. | Cambiada la condición *or* por *and*. |
| CP-09 | OK | - |
| CP-10 | OK | - |
| CP-11 | OK | - |
| CP-12 | OK | - |
| CP-13 | OK | - |
| CP-14 | OK | - |
| CP-15 | OK | - |
| CP-16 | OK | - |
| CP-17 | OK | - |
| CP-18 | Fallo: no muestra el movimiento de las fichas correctamente. | Solución: Se hace una subscripción para cada ficha de cada jugador en lugar de hacer una que solo apunte al jugador del turno actual. |
| CP-19 | OK | - |
| CP-20 | OK | - |
| CP-21 | Fallo: so descuenta rondas ni guarda los datos de los jugadores. | Se añaden las llamadas al servidor que faltaban. |
| CP-22 | OK | - |
| CP-23 | Fallo: no se guardan los puntos del jugador que no ha terminado el turno. | Guardar el jugador en cada movimiento y no solo al final del turno. |

| | | |
|---|---|---|
| CP-24 | Fallo: no se muestra la ficha en la casilla de salida. | Solución: Se actualiza la posición de cada ficha a la casilla de salida antes de guardar la partida en el servidor y redirigir la página. |
| CP-25 | Fallo: solo redirige en la pantalla del usuario que ha dado al botón. | Se añade la llamada a la base de datos que actualiza el atributo "finPartida" |
| CP-26 | OK | - |
| CP-27 | OK | - |

**Tabla 7.27. Resultados y soluciones de los casos de prueba**

# Capítulo 8

# Seguimiento del proyecto

## 8.1. Seguimiento de los riesgos

En la sección 4.1.6 se elaboró la tabla 4.6, donde se especificaban las causas que podían retrasar el proyecto, su probabilidad, su impacto, que hacer para evitar que sucedieran y el plan de acción si llegaran a ocurrir. A pesar de intentar llevar a cabo todos los planes de mitigación especificados en dicha tabla se han producido dos riesgos durante la iteración 1 del desarrollo de este trabajo de fin de grado:

- **R**01-Fallo de planificación.
- **R**10-Dificultad en el aprendizaje de las tecnologías escogidas.

### 8.1.1. R01-Fallo de planificación.

Se planificó que al finalizar la iteración 1 había de haber implementados 13 casos de uso pero no se tuvo en cuento el tiempo que se iba a dedicar a configurar el entorno de desarrollo, la correcta inicialización del proyecto y de todas las herramientas que se utilizan en él para que trabajen conjuntamente.

Todo esto más la comprensión de las tecnologías no fue representado en la planificación de la primera iteración lo que hizo que se gastara el tiempo en esto y no en la realización de los casos de uso.

### 8.1.2. R10-Dificultad en el aprendizaje de las tecnologías escogidas.

A pesar de la amplia documentación que existe en internet sobre *Spring Boot* y de haber cursado la asignatura "Diseño Basado en Componentes" donde se sientan las bases del *framework Angular CLI* se ha dedicado más tiempo de lo esperado al estudio y comprensión de todas las herramientas.

Este riesgo sucedió en la iteración 1 y tras aplicar el el plan de contingencia, explicado a continuación, volvió a suceder en la siguiente iteración, la 2.

### 8.1.3. Plan de contingencia

Como ambos riesgos sucedieron en la iteración 1, en la reunión de seguimiento se decidió implementar los planes de contingencia para ambos riesgos:

- **R01-Fallo de planificación.**
  Priorizar CU a desarrollar, terminar la línea base del juego y dejar variaciones para próximas versiones.
- **R10-Dificultad en el aprendizaje de las tecnologías escogidas.**
  Cambiar tecnologías y replanificar tareas.

Como en esta reunión se había conseguido, dedicando mucho tiempo y esfuerzo, configurar enteramente el entorno de desarrollo se decidió únicamente replanificar y priorizar los casos de uso y no cambiar las tecnologías que ya estaban siendo usas.

En la iteración 2 volvió a suceder el riesgo R10 por lo que se decidió volver a replanificar, decidiendo realizar la linea base del juego, dejar la gestión de usuarios para futuras versiones y entregar en segunda convocatoria. A estas alturas se consideró arriesgado cambiar las tecnología usadas por lo que no se hizo.

## 8.2. Seguimiento de la planificación

En la tabla 8.1 se muestra el tiempo real dedicado a cada tarea hasta la iteración 1 a partir de la cual se hizo la replanificación comentada en la sección anterior, mostrada en la tabla 8.2.

En estas dos tablas se pueden ver los retrasos en la iteración 1 y 2 debidos a los riesgos ocurridos y como en la iteración 3 y 4 aumentan las horas dedicadas a cada iteración. Esto hace que haya una gran diferencia de horas totales dedicadas al trabajo de fin de grado, en la primera iteración se estimaron 337h 30m, en la replanificación aumentaron hasta 484h 30m y finalmente se han dedicado 647h 35m, casi el doble de lo estimado inicialmente.

| Iteración | Inicio Iteración | Fin Iteración | Fase | Tarea | Duración estimada | Duración real |
|---|---|---|---|---|---|---|
| 0 | 08-02-2021 | 07-04-2021 | Inicio | Documentarse sobre la deuda técnica | 5 h 00 m | 7 h 00 m |
| | | | | Documentarse sobre juegos educativos | 5 h 00 m | 7 h 00 m |
| | | | | Explicar Hard Choices | 2 h 30 m | 2 h 55 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Elicitación inicial de requisitos | 4 h 00 m | 4 h 30 m |
| | | | | Primera versión de los capítulos 1,2 y 3 de la memoria | 26 h 00 m | 40 h 51 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Redactar capítulo planificación | 6 h 00 m | 11 h 10 m |
| | | | | Modificar introducción | 3 h 00 m | 1 h 26 m |
| | | | | Modelo inical CU | 2 h 00 m | 0 h 28 m |
| | | | | Modelo inicial de Dominio | 3 h 00 m | 0 h 59 m |
| | | | | Modelo de proceso de negocio | 3 h 00 m | 0 h 57 m |
| | | | | Tutoría | 1 h 00 m | 0 h 23 m |
| | | | | Descripción CU | 3 h 00 m | 3 h 28 m |
| | | | | Repaso de los diagramas CU, Dominio y proceso de negocio | 3 h 00 m | 4 h 11 m |
| | | | | Rehacer planificación | 3 h 00 m | 1 h 28 m |
| | | | | Documentarse sobre las tecnologías a usar en el proyecto | 5 h 00 m | 2 h 26 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Terminar sección identificación de riesgos | 1 h 00 m | 0 h 30 m |
| | | | | Terminar sección de requisitos | 2 h 00 m | 1 h 30 m |
| | | | | Decidir tecnologías a utilizar y documentar | 4 h 00 m | 1 h 43 m |
| | | | | Análisis arquitectónico | 2 h 00 m | 3 h 26 m |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | Elaboración | Diseño arquitectónico | 3 h 00 m | 5 h 18 m |
| | | | | Bocetaje inicial | 3 h 00 m | 4 h 26 m |
| | | | | Diseño de almacenamiento persistente | 2 h 00 m | 2 h 24 m |
| | | | | Diseño detallado | 3 h 00 m | 1 h 06 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Terminar sección de tecnologías a utilizar | 1 h 30 m | 0 h 39 m |
| | | | | Terminar diseño arquitectónico | 2 h 00 m | 1 h 19 m |
| | | | | Terminar bocetaje | 1 h 30 m | 1 h 58 m |
| | | | | Terminar de documentar iteración | 4 h 00 m | 6 h 16 m |
| | | | | Terminar diseño detallado | 2 h 00 m | 1 h 00 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Tiempo total en la iteración | 110 h 30 m | 125 h 47 m |
| 1 | 22-04-2021 | 12-05-2021 | Inicio | Terminar de documentar iteración anterior | - | 21 h 10 m |
| | | | Elaboración | Realizar casos de prueba | 20 h 00 m | 3 h 40 m |
| | | | | Realizar diseño de interfaz de usuario | 12 h 00 m | 3 h 05 m |
| | | | Construcción | Compresión de las tecnologías | - | 25 h 48 m |
| | | | | Configuración del entorno de desarrollo | - | 53 h 26 m |
| | | | | Desarrollar CU | 80 h 00 m | 27 h 03 m |
| | | | Transición | Realizar pruebas para cada CU | 10 h 00 m | - |
| | | | | Desplegar prototipo y documentar | 8 h 00 m | - |
| | | | | Redactar manual de usuario | 4 h 00 m | - |

| | | |
|---|---|---|
| Tutoría | 2 h 00 m | 0 h 30 m |
| Tiempo total en la iteración | 136 h 00 m | 134 h 42 m |

**Tabla 8.1. Tiempo dedicado a cada tarea de la planificación inicial**

| Iteración | Inicio Iteración | Fin Iteración | Fase | Tarea | Duración estimada | Duración real |
|---|---|---|---|---|---|---|
| 0 | | | | Tiempo total en la iteración | 110 h 30 m | 125 h 47 m |
| 1 | | | | Tiempo total en la iteración | 136 h 00 m | 134 h 42 m |
| 2 | 13-05-2021 | 09-06-2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m | 4 h 28 m |
| | | | | Replanificar y documentarlo | 2 h 00 m | 1 h 56 m |
| | | | Elaboración | Estudiar las tecnologías a usar | 40 h 00 m | 71 h 14 m |
| | | | Construcción | Desarrollar CU | 50 h 00 m | 22 h 03 m |
| | | | Transición | Realizar pruebas para cada CU | 10 h 00 m | - |
| | | | | Tutoría | 1 h 00 m | 0 h 30 m |
| | | | | Tiempo total en la iteración | 106 h 00 m | 100 h 11 m |
| 3 | 10-06-2021 | 30-06-2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m | 2 h 06 m |
| | | | Elaboración | Realizar casos de prueba | 15 h 00 m | 8:40:00 |
| | | | Construcción | Desarrollar CU | 100 h 00 m | 126 h 32 m |
| | | | Transición | Realizar pruebas para cada CU | 8 h 00 m | 3 h 44 m |
| | | | | Desplegar prototipo y documentar | 3 h 00 m | - |
| | | | | Redactar manual de usuario | 2 h 00 m | - |
| | | | | Tutoría | 1 h 00 m | 0 h 40 m |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | Tiempo total en la iteración | 132 h 00 m | 141 h 42 m |
| 4 | 01-07-2021 | 13-07-2021 | Inicio | Terminar de documentar iteración anterior | 3 h 00 m | 12 h 29 m |
| | | | Elaboración | Terminar casos de prueba | 5 h 00 m | 3 h 43 m |
| | | | Construcción | Terminar desarrollo | 50 h 00 m | 119 h 51 m |
| | | | Transición | Realizar todas las pruebas | 4 h 00 m | 5 h 26 m |
| | | | | Desplegar prototipo y documentar | 2 h 00 m | - |
| | | | | Terminar manual de usuario | 2 h 00 m | 2 h 44 m |
| | | | | Tutoría | 1 h 00 m | 1 h 00 m |
| | | | | Tiempo total en la iteración | 67 h 00 m | 145 h 13 m |
| | | | | | Total | 647 h 35 m |

**Tabla 8.2. Tiempo dedicado a cada tarea de la replanificación inicial**

# Capítulo 9

# Conclusiones y trabajo futuro

En este capítulo se destacan los objetivos cumplidos, se detallan las líneas de trabajo para versiones futuras y se da una valoración personal por parte de la alumna sobre todo su trabajo realizado.

## 9.1. Objetivos conseguidos

A continuación se detallan los objetivos que se han logrado:

- Se ha realizado un estudio y comprensión sobre el concepto de la deuda técnica.
- Se ha estudiado y entendido un *framework* nuevo: *Spring Boot*.
- Se han afianzado conocimientos sobre el *framework* Angular CLI y herramientas utilizadas durante el proyecto.
- Se ha implementado la línea base del juego de forma funcional.
- Se ha conseguido que el juego sea multijugador en tiempo real.
- Se ha realizado un proyecto bien estructurado de forma que se pueda continuar con su implementación de forma sencilla.
- Cuatro de cinco vistas se han implementado de forma adaptativa.
- Se han realizados vistas sencillas e intuitivas.
- Se ha conseguido que la aplicación funcione tanto en el navegador *Mozilla Firefox* como en *Google Chrome*.
- Se ha conseguido realizar todas las fases del proyecto de forma individual por parte de la alumna.

## 9.2. Líneas de trabajo futuras

No se han cumplido con todos los objetivos planteados para este proyecto por lo que para futuras versiones se debería:

- Realizar los casos de uso que permiten pausar la partida, reanudarla y añadir variaciones.
- Implementar la gestión de perfiles de usuarios.
- Implementar los rankings.
- Diseñar interfaces más atractivas y que todas se adapten a distintos dispositivos.

## 9.3. Valoración personal

Este proyecto me ha permitido aprender nuevas tecnologías y afianzar conocimientos ya aprendidos durante la carrera, tanto a nivel de programación como de análisis, diseño y planificación de proyectos. Sobre todo me ha hecho reflexionar sobre el concepto de la deuda técnica, el cual me parece muy importante que todos los desarrolladores de software tengamos presente en todos nuestros proyectos.

Por último, me gustaría remarcar que, a pesar de las noches largas y de las dificultades que me he encontrado con el paso de los días, estoy satisfecha con mi esfuerzo y el trabajo realizado y he disfrutado con el reto que ha supuesto enfrentarme sola por primera vez a un trabajo de este tipo.

# Bibliografía

[1] alperkurtul. An Easy Way to Access Firebase Realtime Database in Spring Boot. https://github.com/alperkurtul/spring-boot-starter-firebase-realtime-database, 2020-04-24. Accessed: 2021-06-16.

[2] Angular. What is Angular? https://angular.io/guide/what-is-angular, 2021-03-08. Accessed: 2021-04-20.

[3] Bootstrap. Build fast, responsive sites with Bootstrap. https://getbootstrap.com/, 2021. Accessed: 2021-04-20.

[4] Escuela de Ingeniería Informática, Universidad de Valladolid. Trabajos Fin de Grado Calendario de depósito y defensa Curso 2020-2021. https://www.inf.uva.es/wp-content/uploads/2020/11/Calendario_TFG_TFM_2020-21.pdf, 2021. Accessed: 2021-04-18.

[5] Firebase. Firebase Realtime Database. https://firebase.google.com/docs/database, 2020-12-18. Accessed: 2021-04-27.

[6] Firebase. Firebase Realtime Database. https://firebase.google.com/docs/auth, 2020-12-18. Accessed: 2021-04-27.

[7] Firebase. Planes de precios: Comienza con un plan gratuito y luego opta por uno prepago. https://firebase.google.com/pricing, 2021. Accessed: 2021-04-27.

[8] Bob Hughes and Mike Cotterell. Activity planning. In *Software project management*, volume 5, chapter 6. McGraw-Hill Education, 2009.

[9] Bob Hughes and Mike Cotterell. Resource allocation. In *Software project management*, volume 5, chapter 8. McGraw-Hill Education, 2009.

[10] Bob Hughes and Mike Cotterell. Risk management. In *Software project management*, volume 5, chapter 7. McGraw-Hill Education, 2009.

[11] INEED. Buscar sueldos. https://es.indeed.com/career/salaries, 2021. Accessed: 2021-03-08.

[12] jangelzamora. ¿Qué es Firebase? La mejorada plataforma de desarrollo de Google. https://elandroidelibre.elespanol.com/2016/05/firebase-plataforma-desarrollo-android-ios-web.html, 2016-05-19. Accessed: 2021-04-27.

[13] Javier Garzas. La deuda técnica. Todo el mundo debería saber que la mala calidad software al final se paga. https://www.javiergarzas.com/2012/11/deuda-tecnica-2.html, 2012-11-22. Accessed: 2021-02-22.

[14] jcbmarqz. Standards, compliance, and Rational Unified Process. https://jcbmarqz.blogspot.com/2010/02/furps.html, 2004-05-26. Accessed: 2021-03-05.

[15] Miguel A. Laguna. 2. Modelado de Requisitos: Procesos de negocio, 2019. Consulted: 2021-02-28.

[16] Nanette Brown, Philippe Kruchten, Erin Lim, Robert L. Nord e Ipek Ozkaya. The Hard Choices Game Explained. https://resources.sei.cmu.edu/asset_files/WhitePaper/2017_019_001_28931.pdf, 2017. Accessed: 2021-02-14.

[17] OODesign. Dependency Inversion Principle. https://www.oodesign.com/dependency-inversion-principle.html. Accessed: 2021-07-01.

[18] OODesign. Singleton Pattern. https://www.oodesign.com/singleton-pattern.html. Accessed: 2021-07-01.

[19] Open Source Initiative. The 3-Clause BSD License. https://opensource.org/licenses/BSD-3-Clause. Accessed: 2021-07-16.

[20] Oracle. Data Access Object . https://www.oracle.com/java/technologies/data-access-object.html, 2021. Accessed: 2021-06-17.

[21] Visual Paradigm. What is Composite Structure Diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-composite-structure-diagram/, 2021. Accessed: 2021-07-18.

[22] Priya Pedamkar. What is MVC Design Pattern? https://www.educba.com/what-is-mvc-design-pattern/, 2020. Consulted: 2021-06-17.

[23] Pablo Rodríguez. La deuda técnica, un lastre para las tecnológicas: un estudio señala que los informáticos pierden casi un día de trabajo a la semana para solventarlas. https://cutt.ly/YmNu1pS, 2021-07-09. Accessed: 2021-07-18.

[24] Spring. Building a RESTful Web Service. https://spring.io/guides/gs/rest-service/, 2016. Accessed: 2021-05-11.

[25] Spring. Introduction to the Spring Framework. https://docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html/overview.html, 2016. Accessed: 2021-04-20.

[26] Spring. Spring Framework Overview. https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#overview, 2021-04-13. Accessed: 2021-04-20.

[27] TeacherTutorials. Inversion of Control. https://www.tutorialsteacher.com/ioc/inversion-of-control, 2020. Accessed: 2021-07-01.

[28] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2019-2020 (Mención Ingeniería de Software). `https://albergueweb1.uva.es/guia_docente/uploads/2021/545/46976/1/Documento.pdf`, 2021. Accessed: 2021-07-06.

[29] Ward Cunningham. The WyCash Portfolio Management System. `http://c2.com/doc/oopsla92.html`, 1992-03-26. Accessed: 2021-02-22.

[30] Yania Crespo González-Carvajal. Tema 3 Arquitectura del Software y Tema 5 Patrones de diseño. Consulted: 2021-06-17.

# Apéndice A

# Manuales

## A.1. Manual de despliegue

A continuación se detallan las dependencias requeridas y los pasos a seguir para el despliegue del proyecto.

### A.1.1. Requisitos de previous al despliegue

Para la puesta en marcha del proyecto es necesario tener instalado:

- JDK, versión 1.8 o superior
- Apache Maven, versión 3
- Angular CLI, versión 11, Node 16
- Apache Tomcat 8

Para la base de datos es necesario crear las variables de entorno con la ruta al fichero .json con la configuración de la misma y otra para la token de acceso:

- AUTH_KEY_HCG = "LumPfW3z6e4I79VfMJ2FtBCwmCQfQ87LXeeSSIAP"
- CONF_BD_HCG_JSON = "<ruta absoluta>/hardChoicesGame/src/main/resources/dbKey.json"

### A.1.2. Ejecución

Acceder a la carpeta */hardChoicesGame* y desde ahí ejecutar `mvn spring-boot:run`, tras la ejecución se podrá acceder al proyecto desde *localhost:8080* si está instalado en local.

Se ha utilizado una máquina virtual provista por la escuela para probar el despliegue fuera del entorno de desarrollo habitual.

## A.2. Manual de usuario

### A.2.1. Crear partida

El usuario que desee crear una partida será el moderador de la misma.

Primero ha de introducir un alias, este será público para todos los jugadores que se unan posteriormente a la partida, y darle al botón "Crear partida". Imagen: A.1.

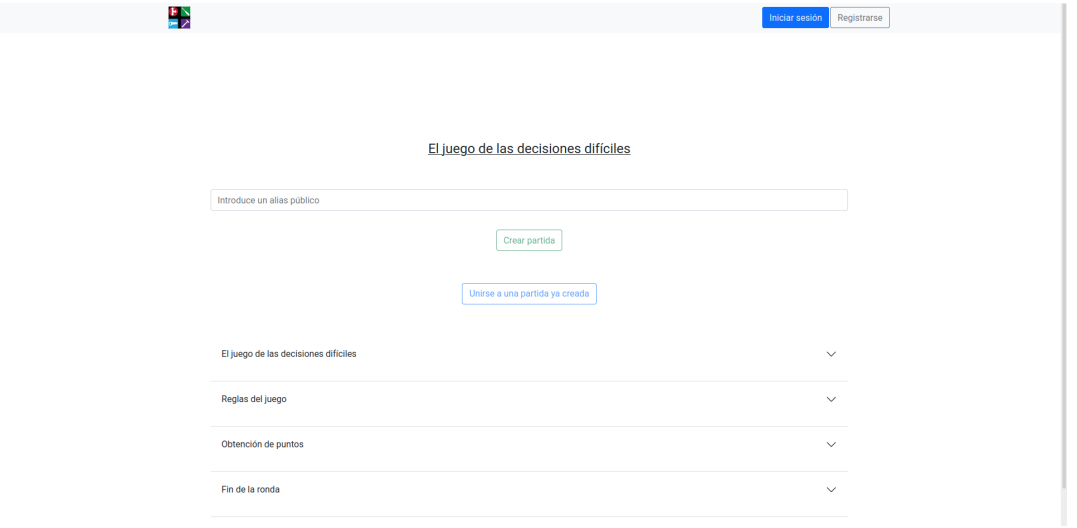


**Figura A.1. Pantalla principal**

A continuación ha de rellenar los datos del formulario de la imagen A.2 y darle nuevamente al botón "Crear partida". Para el canal de comunicación se puede introducir un enlace a una plataforma de comunicación online, este será accesible para todos los jugadores de la partida.

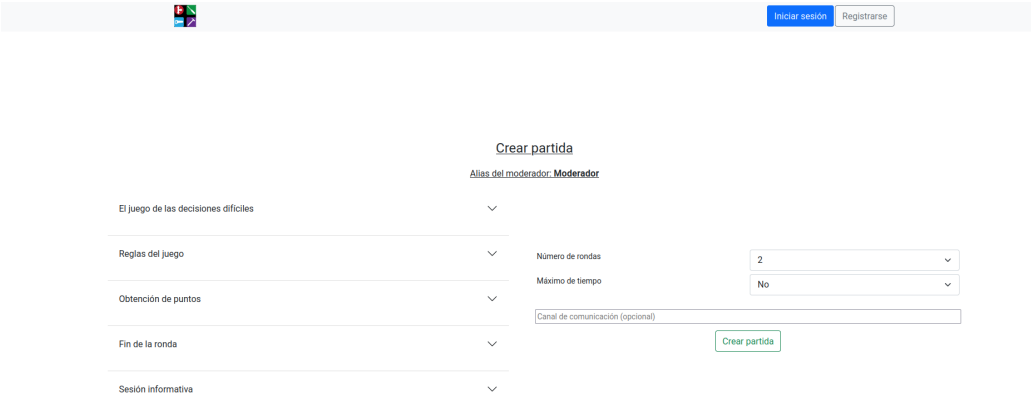


**Figura A.2. Crear partida**

### A.2.2. Iniciar partida

Para iniciar la partida ha de esperar a que unan entre dos y cuatro jugadores. En la vista, imagen A.3, hay un contador de jugadores que especifica cuantos jugadores se han unido. Además se muestra el código de la partida, el cual ha de hacer llegar a los jugadores para que se unan a la partida.

Cuando todos los jugadores se hayan unido presionar el botón “Iniciar partida”.

### A.2.3. Unirse a una partida

Los usuarios que se unan a una partida ya creada serán jugadores de la partida.

Primero ha de introducir un alias, este será público para todos los usuarios unidos a la partida, y darle al botón ”Unirse a una partida ya creada” de la imagen A.1.

A continuación, en la vista de la imagen A.4, ha de introducir el código de la partida que le ha hecho llegar el moderador. Aparecerá un *pop-up* cuando se haya unido correctamente a la partida.

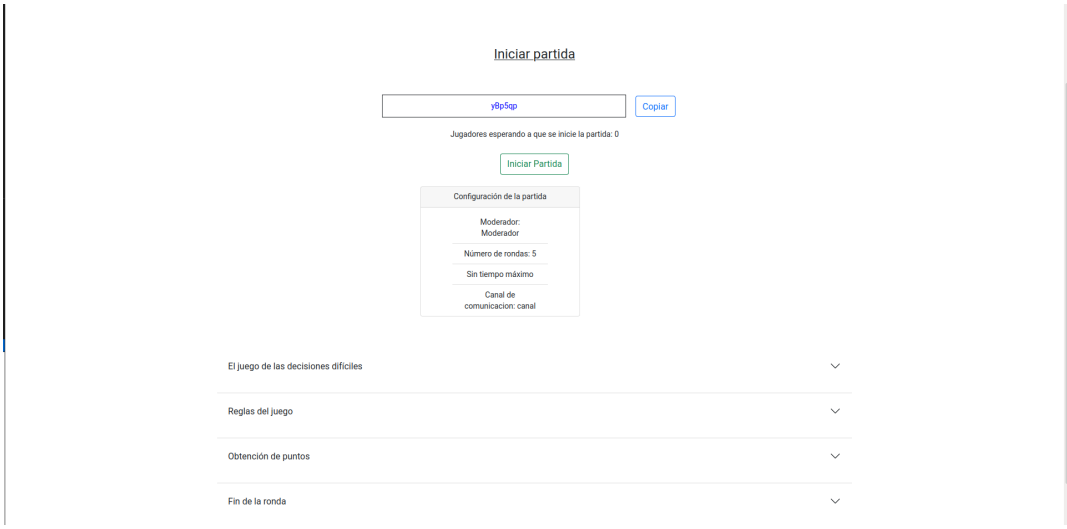


**Figura A.3. Iniciar partida**

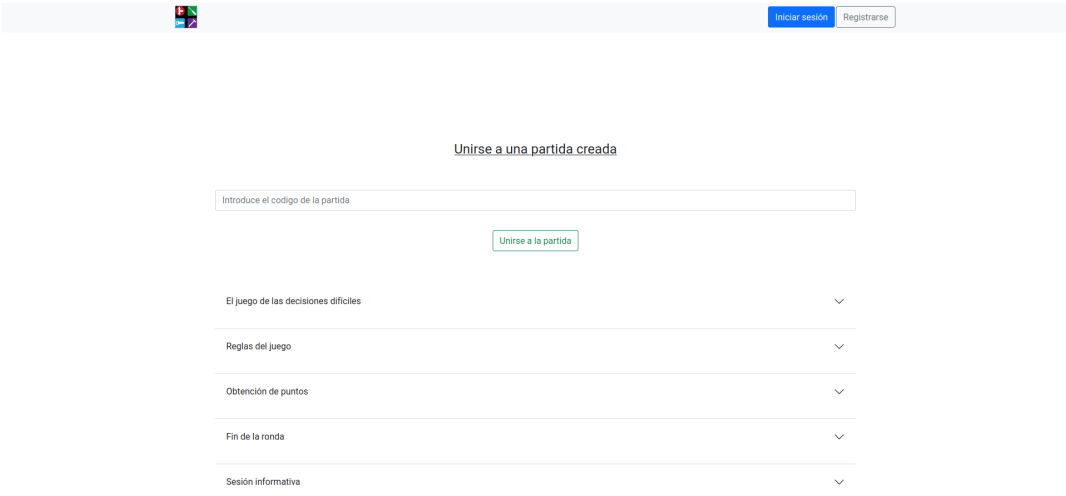


**Figura A.4. Unirse a una partida ya creada**

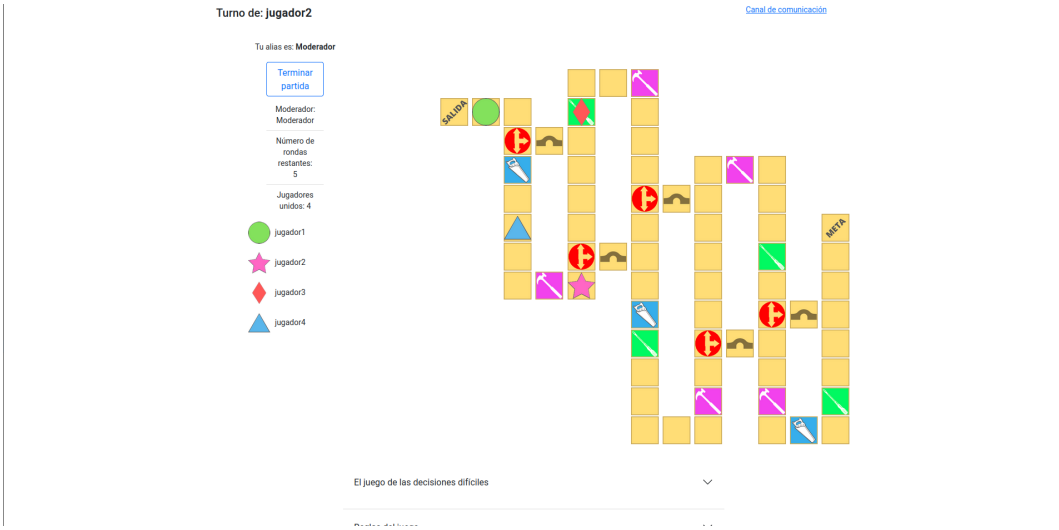


**Figura A.5. Pantalla de juego para el moderador**

## A.2.4. Jugar

### Moderador

El moderador de la partida verá una pantalla como la mostrada en la imagen A.5 donde verá a todos los jugadores unidos a la partida junto a sus fichas, el tablero de juego y, haciendo *scroll*, un desplegable con información del juego.

Si desea terminar la ronda antes de que un jugador llegue a la meta puede presionar el botón de la pantalla. Se guardarán los puntos de todos los jugadores y se redirigirá a la pantalla para realizar la sesión informativa.

### Jugador

Los jugadores verán el tablero de juego y, a la izquierda el botón para lanzar el dado, los botones con flechas para mover su ficha, su ficha y las de sus compañeros y su tablero de puntos. Además podrán unirse al canal de comunicación mediante el enlace mostrado en la esquina superior derecha.

Todo lo anterior se muestra en la imagen A.6.

*Sesión informativa* En esta pantalla, imagen A.7, se puede ver la puntuación total de todos los jugadores. Además, cualquier jugador podrá añadir más rondas a la partida, continuar a la siguiente ronda o terminar la partida por completo.

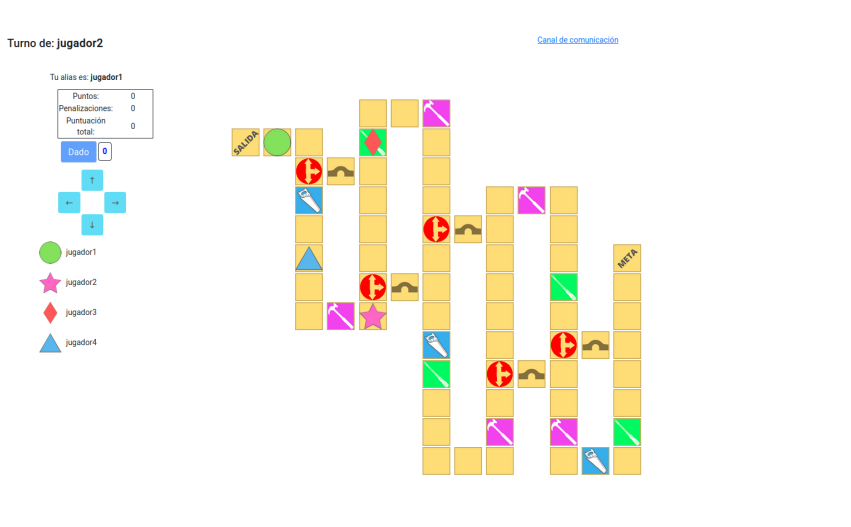


**Figura A.6. Pantalla de juego para el jugador**

Esta vista está dedicada a la sesión informativa donde se discute sobre la deuda técnica y las posibles estrategias para las rondas siguientes, a estas alturas todos los jugadores han de tener toda la información posible por eso se ha añadido la información del juego para que todos accedan a ella haciendo *scroll*.

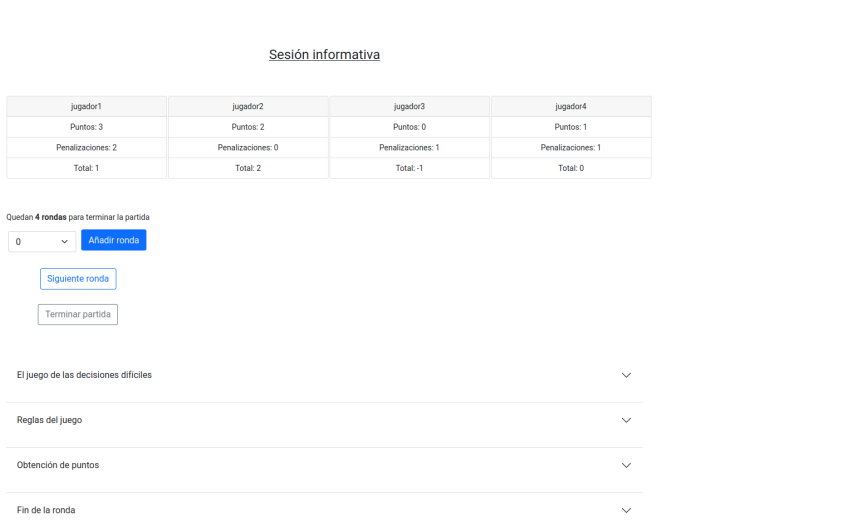


**Figura A.7. Sesión informativa**

# Apéndice B

# Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código: `https://gitlab.inf.uva.es/rebhern/tfg-RebecaHernando`.