Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

Diseño de la interacción y desarrollo del back-end de Crossroads 2.0, un juego educativo para concienciar sobre el cambio climático

---

Alumno: Lucas Matías González Calderón

Tutores: Yania Crespo González-Carvajal
David Escudero Mancebo

Agradecimientos

En primer lugar agradecer a mis padres y a mi hermano por aguantarme durante todo este viaje que ha sido la universidad, sobretodo durante este último y complicado año. Que, aunque las cosas no hayan sido fáciles, siempre han estado ahí para echarme una mano.
A mis amigos más cercanos, a cada uno de *Partidillos*, que pese el tiempo que pase seguimos siendo como hermanos, estando ahí para lo bueno y para lo malo. Gracias de corazón.
A mis compañeros de universidad, que día tras día, semana tras semana hemos hecho piña para ayudarnos los unos a los otros, incluso después de la carrera, manteniendo el compañerismo siempre por encima.
A mis tutores, David y Yania, por la increíble paciencia que han tenido conmigo, por mis despistes, por mis preguntas o cambios, por todo lo que han tenido que aguantar y que aún así me han ayudado en este tiempo.
Al proyecto europeo de LOCOMOTION por la financiación del proyecto, a los investigadores que forman parte del mismo y al grupo de ECO-Profes por su colaboración en el proyecto.
A toda la gente que me haya brindado su apoyo o ayuda durante este tiempo, a todos aquellos que me han hecho ser quien soy ahora, acompañándome durante este viaje, a los que menciono y a los que no, gracias por vuestro tiempo.

Resumen

La gamificación es una técnica de aprendizaje que tiene como objetivo combinar las mecánicas de los juegos con la actividad educativa y profesional, mejorando la comprensión de los conocimientos y aumentando el interés. En este proyecto, se aplicará esta técnica al juego de Crossroads, un juego ya existente, jugado con un simulador y grupos en papel. Ésta nueva version, Crossroads II, pasará a ser un juego multijugador online por equipos en una aplicación web multiplataforma, modificando la dinámica del juego e introduciendo nuevas posibilidades al proyecto. En este documento se pretende reflejar la vida del proyecto, sus diferentes fases, tecnologías utilizadas y resultados obtenidos para la creación del backend de esta aplicación.

Abstract

Gamification is a learning technique that aims to combine the mechanics of games with educational and professional activity, improving the understanding of knowledge and increasing interest. In this project, this technique will be applied to the Crossroads game, an already existing game played with a simulator and groups, all done in paper. This new version, Crossroads II, will become an online multiplayer team game in a multiplatform web application, modifying the game dynamics and introducing new possibilities to the project. This document aims to reflect the life of the project, its different phases, technologies used and results obtained for the creation of the backend of this application.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto

Actualmente, el cambio climático es uno de los temas más comentados por la gente y los medios. Como han confirmado estudios científicos [3], estamos al borde del colapso. Científicos de todo el mundo han hablado sobre la fecha límite de 20 años para rectificar el camino que hemos tomado [23]. De lo contrario, el daño será tal que aunque detengamos todo la sociedad en seco, el destino de la vida, tal y como la conocemos, estará sentenciado. Gracias a la concienciación de la población, estamos empezando a generar los primeros cambios, pero este es el primer paso de un largo camino. Existen varios planes y proyectos de medidas contra el cambio climático y de concienciación sobre el mismo, y el proyecto europeo LOCOMOTION [13] es uno de ellos. Dicho proyecto pretende hacer más robusto, transparente, accesible y confiable el modelo MEDEAS [4], centrándose este documento en la accesibilidad del mismo.

Este Trabajo de Fin de Grado (TFG), se centra en el diseño de un juego web multijugador cooperativo y social y su posterior implementación del *backend* de la aplicación. Como cualquier desarrollo software, se realizará una planificación del proyecto, análisis de los requisitos que debe tener la aplicación, un diseño del mismo y pruebas para corroborar su funcionamiento. Se elaborará un seguimiento de cada una de las etapas del desarrollo con su correspondiente documentación.

1.2. Motivación

Aunque gran parte de la población conoce acerca del cambio climático y quiere hacer algo para cambiar el curso del mismo, poca gente sabe cómo se debe actuar y cuál es el impacto

de las medidas políticas en la sociedad. Este proyecto pretende concienciar a la gente de los efectos de estas políticas y de los sacrificios que debemos hacer como sociedad para poder frenar la catástrofe a la que nos enfrentamos.

1.3. Breve presentación de Crossroads

La versión actual de Crossroads [5] es un juego cooperativo y social que se realiza en grupos. Se desarrolla de forma presencial, rellenando un formulario en papel. El formulario que completa cada equipo se entrega al moderador que dirige la partida. Éste introduce las respuestas proporcionadas por los jugadores en un simulador desarrollado en Vensim [30] y devuelve unas gráficas como respuesta. Estas simulaciones tardan bastante tiempo, entre 5 y 10 minutos, puesto que cada vez que se introducen los datos se realiza una simulación. Si son varios grupos los que juegan a la vez, esto supone un tiempo de espera bastante alto. Por eso surge la idea de Crossroads II [6]. Se pretende informatizar el juego, para poder jugar tanto presencial como a distancia, y sin necesidad de esperar para obtener los resultados.

1.4. Revisión de juegos relacionados

Varios juegos han sido los que han inspirado el desarrollo de este proyecto, ya sea por su gestión de salas, o gestión de los usuarios, sistemas de puntuación o por cómo permiten la interacción de las personas en el transcurso de la partida.

1.4.1. Avalon

Este juego [15], tanto de mesa como de dispositivos móviles, es un juego por equipos. En él, un conjunto de soldados formado por los jugadores, irán embarcándose en misiones. Dentro del grupo hay unos traidores que intentarán sabotear las misiones.

La dinámica de juego es la siguiente: cada ronda, un líder de grupo propondrá a quienes mandara a la expedición, (en la primera ronda serán 2 personas, en la segunda 3, en la tercera 4, en la cuarta 3 y en la quinta 4). Esta propuesta se pondrá en votación. Si no es aprobada, el cargo de líder pasa al siguiente jugador. Si por el contrario es aprobado, los enviados harán un voto oculto decidiendo si se completa o no la misión. Con que haya sólo un voto negativo, la misión se considera como fallida.

El juego lo gana el equipo que obtenga la victoria en 3 rondas, las cuales dependen del éxito o fracaso de la misión. Entre ronda y ronda se permite a los jugadores discutir el por qué confían o desconfían de la gente y de sus decisiones como líderes.

Aunque en nuestro proyecto no existe el factor del engaño, esta idea de conversación y diálogo sobre las decisiones tomadas entre rondas han influenciado el diseño del mismo.

1.4.2. JackBox Party Pack

Esta colección de juegos multijugador cómicos [12] tienen varios campos. Juegos de engaño, trivial, dibujo, etc. Para jugar una partida, debes unirte a una sala a través de un código alfabético de 4 caracteres. Las salas son privadas. Cada juego tiene su sistema de puntos que recompensa las decisiones bien tomadas. Como se muestra en la Figura 1.1, tanto el juego te otorga puntos por hacer respuestas correctas (columna de la derecha de la imagen) como los jugadores votan las mejores respuestas de los demás participantes (*Thumbs up* a la izquierda de las respuestas).

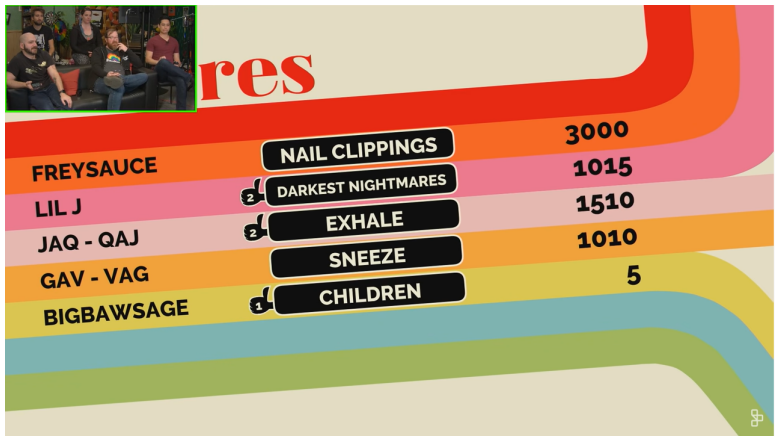

Figura 1.1: Captura de pantalla de una partida de Fibagge.

En uno de los juegos, *Patently Stupid*, en el tiempo de espera mientras los jugadores completan sus respuestas, la aplicación proporciona datos sobre la partida, ya sean reales o cómicos, incluso estadísticas de los propios jugadores como se puede ver en la Figura 1.2. El formato de salas de estos juegos entre cargas son un claro ejemplo de lo que se pretende hacer con este proyecto.

1.5. Objetivos del TFG

Los objetivos a cumplir en este proyecto son los siguientes:

- Definir y analizar los requisitos del juego, definiendo sus casos de uso y realizando los diagramas que permiten documentar los resultados de este análisis.
- Realizar el diseño de la interacción
- Diseñar la arquitectura del proyecto.
- Obtener los datos de las simulaciones de Vensim y traducirlos a un formato CSV necesario para poder visualizar los resultados.

Figura 1.2: Captura de pantalla del tiempo entre rondas de Patently Stupid.

- Redactar un manual de uso para la generación de datos y su posterior parseo.
- Aprender las tecnologías de SpringBoot, MySQL Workbench, MongoDB Compass.
- Desarrollar el backend de la primera version de Crossroads 2.0 en Java.
- Desplegar el backend en una máquina de la escuela.

1.6. Estructura de la memoria

A continuación se presentarán cada uno de los capítulos en los que se ha dividido este proyecto, con sus respectivas descripciones sobre qué se trata en cada uno:

Capítulo 1: Introducción. En este capítulo se presenta el proyecto, contexto, motivación, influencias, objetivos del TFG y estructura de la memoria.

Capítulo 2: Requisitos. Se expondrán los conceptos básicos, requisitos de la aplicación y casos de uso, junto con un diccionario de datos para su mayor comprensión.

Capítulo 3: Planificación y seguimiento. Se incluye la planificación inicial del proyecto, presupuesto, cambios que hayan surgido y el desarrollo que ha seguido el trabajo a lo largo de los meses.

Capítulo 4: Análisis. En este capítulo se explicará el modelo de interacción de los usuarios con la aplicación junto con el modelado conceptual.

Capítulo 5: Tecnologías utilizadas. Se presentarán las tecnologías utilizadas a lo largo del proyecto tanto para la gestión del mismo como para el desarrollo e implementación.

Capítulo 6: Diseño. Se documenta la fase de diseño de la aplicación. Se procede a definir la arquitectura, tanto la arquitectura lógica de la aplicación y el diseño lógico de las bases de datos, como el despliegue de la arquitectura lógica sobre la arquitectura física, junto con el diseño de las llamadas API REST que se han desarrollado y los mockups del frontend.

Capítulo 7: Implementación y pruebas. Se detalla el proceso de implementación del backend y las pruebas realizadas.

Capítulo 8: Conclusiones y trabajo futuro. Se reúnen las conclusiones finales, objetivos cumplidos a lo largo del proyecto y posibles futuras mejoras.

Anexos. Contiene un manual de instalación y de programador.

Bibliografía. Referencias bibliográficas consultadas para la realización del proyecto.

Capítulo 2

Requisitos

En este capítulo se presentarán los requisitos funcionales, no funcionales y de información que tendrá la aplicación. También se expondrán los casos de uso, sus requisitos relacionados y el diagrama correspondiente.

Antes de explicar todo, será necesario establecer los conceptos con los que se tratará, así como sus definiciones y tipos.

2.1. Definiciones necesarias

2.1.1. Conceptos y roles en el juego

Se distinguen dos tipos de roles en el juego, los jugadores participantes y el moderador.

Los jugadores participantes siempre juegan en grupos. Una partida se realiza en una sala con un moderador y jugadores que se distribuyen en grupos. La distribución en grupos puede ser por selección o automáticamente.

El moderador debe crear la sala donde se desarrollará la partida

La partida se dividirá en rondas.

2.1.2. Iteración de las rondas

Las fases del juego en el antiguo diseño se podrían dividir en tres:

- Fase de explicación y primera ronda (Figura 2.1): El moderador explica a los participantes el funcionamiento del juego y les reparte los formularios (1). Los grupos discuten las

respuestas (2) y entregan el formulario al moderador (3). Este introduce las respuestas en el simulador (4). Esta fase se realiza solo una vez.

- Fase intermedia (Figura 2.2): El moderador obtiene los resultados (5), y se los da a cada grupo (6). Se discute entre todos los resultados obtenidos (7) y los grupos responden a las preguntas otra vez para la siguiente iteración (8). Entregan los resultados al moderador (9) que vuelve a simularlos (10). Esta fase se repite tantas veces como rondas se realicen.
- Fase final (Figura 2.3): El moderador obtiene los resultados de la última simulación (11) y se los entrega a los grupos (12). Entre todos se discuten los resultados (13). Comúnmente, el moderador muestra un escenario con resultados positivos (14). Esta fase sólo se realiza una vez.

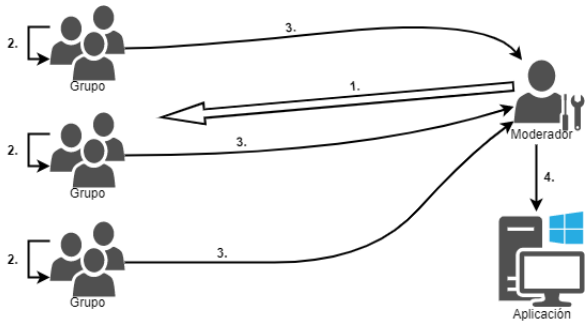

Figura 2.1: Primera fase del diseño antiguo.

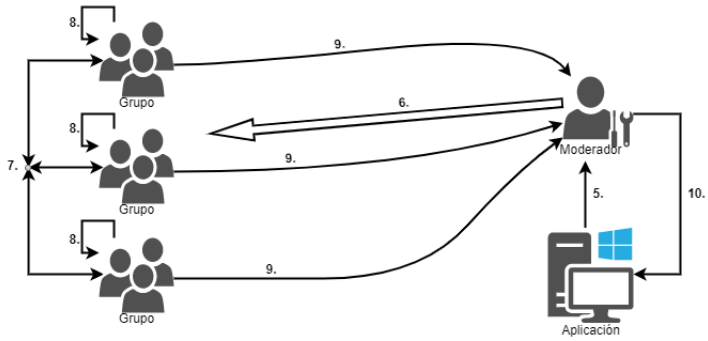

Figura 2.2: Segunda fase del diseño antiguo.

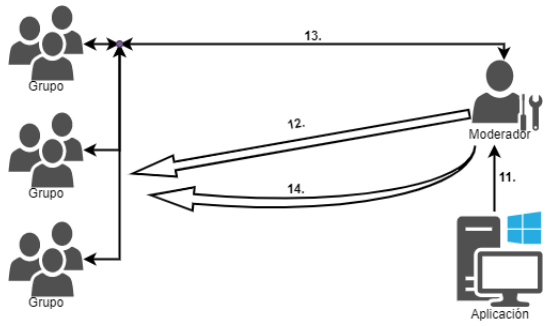

Figura 2.3: Tercera fase del diseño antiguo.

Con el nuevo modelo que se presenta, las fases serían ligeramente diferentes, pero a fin de cuentas, la iteración del juego sería la misma. Las tres nuevas fases serían de la siguiente forma:

- Fase de explicación y primera ronda (Figura 2.4) : El moderador explica a los participantes el funcionamiento del juego (1). Los grupos discuten las respuestas (2) y responden a las preguntas en la propia aplicación (3). El moderador también tendría acceso a la aplicación controlando la Partida. Esta fase se realiza solo una vez.
- Fase intermedia (Figura 2.5) : Los grupos reciben los resultados de la simulación (4). Se discute entre todos los resultados obtenidos (5) y los grupos responden a las preguntas otra vez para la siguiente iteración (6 y 7). Esta fase se repite tantas veces como rondas se realicen.
- Fase final (Figura 2.6) : Los grupos reciben los resultados de la última simulación (8). Entre todos se discuten los resultados (9). Comúnmente, el moderador muestra un escenario con resultados positivos (10). Esta fase sólo se realiza una vez.

La mayor diferencia entre el antiguo modelo y el nuevo es que ya no es necesario utilizar al moderador como intermediario, agilizando el juego, y permitiendo la inserción múltiple de respuestas, en vez de depender de una única persona.

Otras diferencias reseñables son: a) que la argumentación en el grupo se debe realizar en un chat entre los jugadores del grupo, b) de la misma manera para llegar a un consenso entre los miembros del grupo se realiza a través de la aplicación, y c) que se introducen elementos de juego para reforzar positivamente el trabajo del grupo y para competir con el resto de los grupos jugando en la sala.

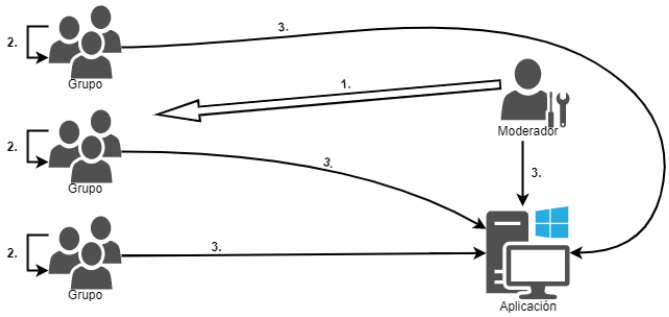

Figura 2.4: Primera fase del nuevo diseño.

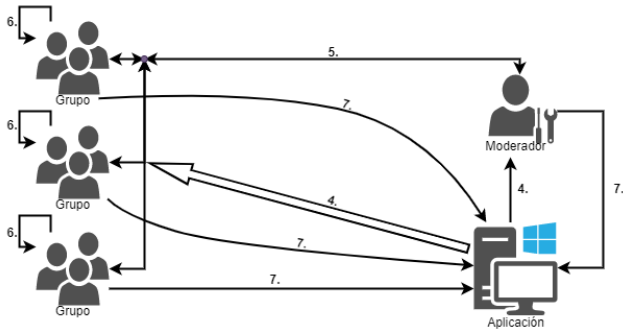

Figura 2.5: Segunda fase del nuevo diseño.

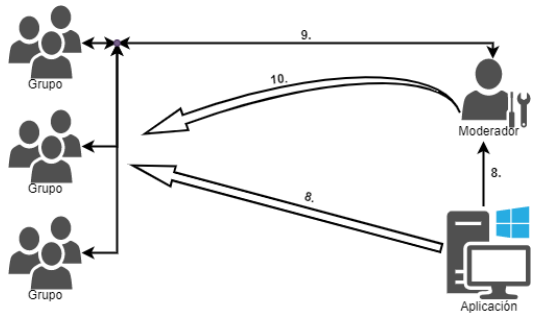

Figura 2.6: Tercera fase del nuevo diseño.

2.2. Especificación de requisitos

La descripción de la versión inicial de Crossroads, entrevistas con los autores de la misma y con investigadores del proyecto LOCOMOTION, así como con profesores del grupo de interés ECO-profes, han sido las principales fuentes de la elicitación de requisitos.

2.2.1. Requisitos funcionales

Tras analizar el juego en su versión inicial, las mejoras y modificaciones que se desean introducir y las nuevas necesidades que surgen, se han identificado una serie de requisitos. Éstos se dividen entre los que serán implementados es una primera versión y en requisitos propuestos para futuras versiones. En ellos se hará referencia a conceptos definidos previamente. Para mayor información sobre los conceptos en cuestión, se puede consultar el diccionario de datos (en el apartado 2.2.4).

En la Tabla 2.1 se presentan los requisitos funcionales que se tendrán en cuenta en la primera versión. En la Tabla 2.2 se presentan los requisitos funcionales que se han definido pero no se tendrán en cuenta en la primera versión, quedando para futuras versiones.

Tabla 2.1: Requisitos funcionales de la primera versión

Nº	**Requisito**
RF01	El sistema permitirá al usuario no registrado registrarse en la aplicación
RF02	El sistema permitirá al usuario registrado identificarse en la aplicación
RF03	El sistema permitirá al usuario registrado recuperar sus credenciales
RF04	El sistema permitirá al usuario registrado ver su Información personal
RF05	El sistema permitirá al usuario registrado cerrar su sesión
RF06	El sistema permitirá al usuario registrado crear una Sala
RF07	El sistema permitirá al usuario registrado buscar Sala
RF08	El sistema permitirá al usuario no registrado unirse a una Sala a través de un código alfanumérico
RF09	El sistema permitirá al usuario con Rol 'Moderador' iniciar una Partida
RF10	El sistema permitirá al usuario con Rol 'Jugador' abandonar una Sala
RF11	El sistema permitirá al usuario registrado modificar su Información personal
RF12	El sistema permitirá al usuario con Rol 'Jugador' cambiar de Chat en una Sala o Grupo
RF13	El sistema permitirá al usuario con Rol 'Jugador' enviar mensajes en el Chat de una Sala o Grupo
RF14	El sistema permitirá al usuario con Rol 'Jugador' moverse en el Formulario
RF15	El sistema permitirá al usuario con Rol 'Jugador' responder a las preguntas del Formulario
RF16	El sistema permitirá al usuario con Rol 'Jugador' responder a la trama narrativa.
RF17	El sistema permitirá al grupo resolver Conflictos en la Propuesta
RF18	El sistema permitirá al usuario con Rol 'Moderador' generar Informe
RF19	El sistema permitirá al usuario con Rol 'Moderador' terminar la Partida.
RF20	El sistema resolverá automáticamente conflictos cuando se da por terminada la Ronda.
RF21	El sistema deberá mostrar al usuario con Rol 'Jugador' los resultados obtenidos por el grupo al final de la Ronda.
RF22	El sistema deberá mostrar al usuario con Rol 'Jugador' los Medallas obtenidos por el grupo al final de la Ronda.
RF23	El sistema deberá mostrar al usuario con Rol 'Jugador' los resultados obtenidos por el grupo al final de la partida así como la posición del grupo respecto a los demás grupos en la Sala en cuanto a los diferentes Medallas.

Tabla 2.2: Requisitos funcionales para futuras versiones

Nº	Requisito
RF24	El sistema permitirá al usuario elegir el Modo de juego
RF25	El sistema permitirá al usuario cambiar el idioma del juego
RF26	El sistema permitirá al usuario unirse a una sala de diferentes formas
RF27	El sistema permitirá al usuario elegir la Dificultad de una Partida
RF28	El sistema permitirá al usuario reconectarse a una Sala
RF29	El sistema permitirá al usuario con Rol 'Moderador' monitorizar a los grupos.
RF30	El sistema permitirá al usuario con Rol 'Moderador' expulsar a un jugador.
RF31	El sistema permitirá al usuario con Rol 'Moderador' expulsar a un Grupo.
RF32	El sistema permitirá al usuario con Rol 'Moderador' eliminar mensajes de un Chat.
RF33	El sistema permitirá al usuario agregar a otros usuarios.

2.2.2. Requisitos no funcionales

En la Tabla 2.3 se presentan los requisitos **no** funcionales que se tendrán en cuenta en la primera versión. En la Tabla 2.4 se presentan los requisitos **no** funcionales que se han definido pero no se tendrán en cuenta en la primera versión, quedando para futuras versiones.

Tabla 2.3: Requisitos no funcionales de la primera versión

Nº	Requisito
RNF01	El sistema deberá garantizar que el 90 % de los usuarios utilizarán la aplicación sin necesidad de volver a consultar la ayuda en el 95 % de las veces.
RNF02	La aplicación devolverá un resultado en menos de 1'5 segundos.
RNF03	La aplicación contará con manuales de usuario bien estructurados.
RNF04	La aplicación deberá utilizar la codificación de caracteres UTF-8.

Tabla 2.4: Requisitos no funcionales para futuras versiones

Nº	Requisito
RNF05	La aplicación debe poder realizar llamadas asíncronas.
RNF06	La aplicación será una aplicación web multiplataforma.
RNF07	La aplicación soportará varios idiomas, inicialmente castellano e inglés.
RNF08	La aplicación web debe poseer un diseño 'Responsive' con el fin de garantizar la adecuada visualización en cualquier plataforma.

2.2.3. Requisitos de información

En la Tabla 2.5 se muestra la especificación de los requisitos de información elicitados.

Tabla 2.5: Requisitos de información de la aplicación

Nº	Requisito
RI01	El sistema deberá almacenar información sobre los usuarios, en particular: nombre, apodo (nick), password, email, fecha de nacimiento y país.
RI02	El sistema deberá almacenar información sobre las salas, en particular: nombre, tamaño, nº de rondas que se jugarán y código de la sala.
RI03	El sistema deberá almacenar información sobre los grupos, en particular: tamaño, integrantes y resultados de las rondas.
RI04	El sistema deberá almacenar información sobre las preguntas, en particular: el identificador, el texto de la pregunta, el tipo de la pregunta (hipótesis, objetivo, medida), sus opciones y el texto de las opciones.
RI05	El sistema deberá almacenar información sobre mensajes, en particular: quien lo envió, tipo de chat en el que se envió, pregunta y respuesta sobre la que se habla en el mensaje y momento de envío y el sentido del mensaje (a favor, en contra o neutral).
RI06	El sistema deberá almacenar información sobre el tipo de chat, en particular: si es chat de grupo o chat de sala.
RI07	El sistema deberá almacenar información sobre las acciones de los usuarios, en particular: qué jugador realizó la acción, cuándo la realizó y tipo de acción (elegir respuesta, enviar mensaje, enviar propuesta, resolver conflicto, entrar en un grupo o pasar una pregunta).
RI08	El sistema deberá almacenar información sobre los resultados, en particular: relación entre las respuestas y los resultados con sus respectivas gráficas.
RI09	El sistema deberá almacenar las argumentaciones finales de los jugadores: explicación de sus respuestas, independiente de cada jugador y realizado al final del formulario.

2.2.4. Diccionario de datos

En esta sección se explican detalladamente todos los conceptos que aparecen de forma corta en la especificación de requisitos. Las Tablas de la 2.2.4 a la 2.23 muestran las diferentes entradas del Diccionario de Datos.

Tabla 2.6: Sala

Sala	
Descripción	Espacio digital formado por los grupos que participan en la Partida. Al terminar la Partida, la sala desaparece.
Acceso	▪ Introduciendo un código de 4 caracteres alfanuméricos (No requiere inicio de sesión, se introduce en la página de Login).

Tabla 2.7: Partida

Partida	
Descripción	Actividad que tiene lugar en la sala. Las personas con Rol 'Jugador' son los que responden a las preguntas y obtienen puntuacion y el 'Moderador' es quien va guiando el juego.

Tabla 2.8: Grupo

Grupo	
Descripción	Conforman una sala. Conjunto de usuarios que comparten propuesta. Deben llegar a un consenso en todas las respuestas a las diferentes preguntas.
Características	▪ Pueden ser formados por los jugadores (uniéndose a los grupos) o de forma aleatoria. ▪ Hasta que todos los usuarios del grupo no estén de acuerdo en la respuesta final, no se permitirá enviar la respuesta. ▪ Posee un canal común para que sus integrantes chateen.

Tabla 2.9: Información personal

Información personal	
Descripción	Datos relevantes del usuario de la aplicación.
Listado	▪ Apodo: Nombre que verán el resto de usuarios de la aplicación. Modificable. ▪ Contraseña: Cadena de caracteres alfanuméricos para identificar al usuario. Modificable. ▪ Correo electrónico: Email asociado a la cuenta. Modificable.

Tabla 2.10: Chat

Chat	
Descripción	Plataforma de mensajería instantánea para que los jugadores conversen.
Tipos	▪ Global: Chat común para todos los jugadores de la sala. ▪ Grupal: Chat privado para los jugadores pertenecientes a un grupo.

Tabla 2.11: Modo de juego

Modo de juego	
Descripción	Especifica cómo se jugará la Partida, cuántas personas y con qué tipo de interacción. Elegido en el menú principal.
Tipos	▪ Un jugador: No se crean grupos ni se une a una sala. Se abre el formulario para una persona, para que sea capaz de ejecutar pruebas al momento. No existe ni puntuación ni chat (futuras versiones). ▪ Local: Simula la ejecución online en un solo ordenador. El jugador tiene el rol de 'Moderador'. Es capaz de crear grupos y realizar pruebas asociadas a los grupos. No existe chat, pero sí puntuación (futuras versiones). ▪ Online: Se busca o crea una sala para jugar con más usuarios en grupos. Existe puntuación y chat tanto de grupo como global (en la versión inicial).

Tabla 2.12: Dificultad

Dificultad	
Descripción	Cantidad de información y preguntas que se realizarán al jugador.
Tipos	▪ Básico: Preguntas básicas, con mucha información para entender las preguntas y con resultados finales sencillos y fáciles de entender. En esta versión solo se tendrá en cuenta esta dificultad. ▪ Experto: Preguntas más técnicas, con menor información en las preguntas pero mayor en los resultados, siendo más específico. Preparado para personas con conocimientos previos.

Tabla 2.13: Formulario

Formulario	
Descripción	Documento de hipótesis, objetivos y medidas que se les propone a los grupos. Posee un campo de trama narrativa donde se pide una argumentación de las respuestas escogidas. Es personal y no tiene por qué coincidir con el de los compañeros. La complejidad de las preguntas depende de la dificultad seleccionada, aunque en esta versión solo existe la dificultad básica

Tabla 2.14: Propuesta

Propuesta	
Descripción	Respuesta realizada por el grupo al conjunto de hipótesis, objetivos y medidas (ver Tabla 2.13). Para que una propuesta pueda ser enviada se necesita consenso, es decir, que las respuestas de todos los integrantes del grupo sean iguales.

Tabla 2.15: Conflicto

Conflicto	
Descripción	Situación de desacuerdo al realizar una propuesta. Ocurre cuando las respuestas de los integrantes del grupo a alguna(s) pregunta(s) no coinciden. Los miembros del grupo deberán debatir en el chat privado del grupo qué propuesta desean enviar. Mientras, se permitirá a los jugadores responder de nuevo las preguntas del formulario que generen dicho problema.

Tabla 2.16: Rol

Rol	
Descripción	Nivel de privilegios y funcionalidades que posee un usuario.
Tipos	▪ Jugador: Puede unirse y salir de una sala, reconectarse, responder a las preguntas, usar el chat, argumentar, solucionar conflictos, enviar una propuesta y cambiar de idioma. ▪ Moderador: Coordinador de la sala. Puede crear y configurar las Salas, iniciar una Partida y finalizarla antes de tiempo, iniciar y terminar rondas y crear PDFs.

Tabla 2.17: Ronda

Ronda	
Descripción	Iteración de la Partida. Cada jugador debe hacer su turno en este periodo de tiempo. Cada ronda tiene una duración indeterminada, puesto que está en manos del 'Moderador' elegir cuando se pasará de ronda. También puede finalizar antes de que el 'Moderador' tome la decisión si todos los Grupos ya han llegado a un consenso y enviado sus propuestas.
Etapas	▪ Formulación de objetivos: Se preguntan los objetivos al jugador. ▪ Ronda de preguntas: Se responden a las preguntas del formulario. ▪ Trama narrativa: Se rellena una argumentación de la elección de las respuestas. ▪ Envío de las preguntas: Se espera a que todos envíen su respuesta. ▪ Resultados: Se devuelven los resultados a los jugadores.

Tabla 2.18: Resultado

Resultado	
Descripción	Gráficas y datos devueltos por la aplicación a los grupos, obtenidos según las propuestas realizadas. El número de gráficas y datos obtenidos dependen de la dificultad seleccionada.
Listado	▪ Gráficas de los objetivos: Se mostrará una guía del objetivo propuesto por el grupo en la ronda y cómo ha sido el resultado de la simulación aplicando las medidas propuestas. ▪ Puntuación de la ronda: Se concederá una puntuación sobre 100 que evalúe el resultado del grupo. ▪ Medallas: Pequeños incentivos que se otorgarán al cumplir ciertos requisitos (ver posibles medallas en la Tabla de Medallas).

Tabla 2.19: Medallas

Medallas	
Descripción	Logros o incentivo que se le otorga a un grupo tras cumplir ciertos requisitos en los resultados de una ronda.
Listado	▪ Trabajo eficiente: Se ha enviado la propuesta en menos de 7 minutos. ▪ Perfeccionista: Se ha conseguido un resultado con menos de 0'2 de diferencia frente a los objetivos ▪ Trabajo en equipo: Se han realizado múltiples argumentos y no ha habido conflicto. ▪ Ecológico: El resultado final ha cumplido los objetivos base del modelo.

Tabla 2.20: Informe

Informe	
Descripción	Documento autogenerado con la información de los resultados obtenidos en la Partida. Aparecerán tantas rondas como se hayan jugado, y cada ronda contendrá los elementos listados a continuación.
Elementos	▪ Trama narrativa donde los jugadores explican el por qué de sus medidas. ▪ Gráficas obtenidas a partir de la simulación. ▪ Mensajes enviados a través del chat. ▪ Elecciones realizadas por cada uno de los jugadores del grupo. ▪ Puntuación obtenida.

Tabla 2.21: Errores de datos

Errores de datos	
Descripción	Tipo de entradas de datos no admitidos por ser incorrectos o por poner en peligro la seguridad del sistema.
Tipos	▪ Entrada vacía: El campo de datos no ha sido rellenado. ▪ Tipo de dato: El tipo de dato esperado y el recibido no coinciden. ▪ Inyección de código: La entrada presenta un peligro para la seguridad al intentar insertar código ajeno en el sistema.

Tabla 2.22: Desconexión

Desconexión	
Descripción	Pérdida de la comunicación con el usuario.
Tipos	▪ Cierre de la aplicación: El usuario cierra la aplicación. ▪ Caída de internet: El usuario pierde la conexión con Internet.

Tabla 2.23: Requisitos de las contraseñas

Requisitos de las contraseñas	
Descripción	Mínimos de seguridad que deben cumplir las contraseñas de los usuarios.
Tipos	▪ Mínimo de 6 caracteres. ▪ Mínimo una letra y un dígito. ▪ Los caracteres especiales (paréntesis, tildes, puntos, comas, símbolos) no son admitidos.

2.3. Casos de uso

En esta sección, se detallan los casos de uso para la aplicación. Estos representan los requisitos de la interacción y especifican las acciones que pueden realizar los usuarios del sistema.

2.3.1. Diagrama de casos de uso

En la Figura 2.7 se muestra el Diagrama de Casos de Uso. El actor Usuario Registrado podrá jugar el Rol de 'Moderador' en el juego. Para los actores Jugador y Usuario Registrado se define una actor generalización que permite indicar los casos de uso que ambos tipos de actores pueden realizar en la aplicación. Se indica que un usuario no registrado, a través del caso de uso Registrarse puede pasar a ser Usuario

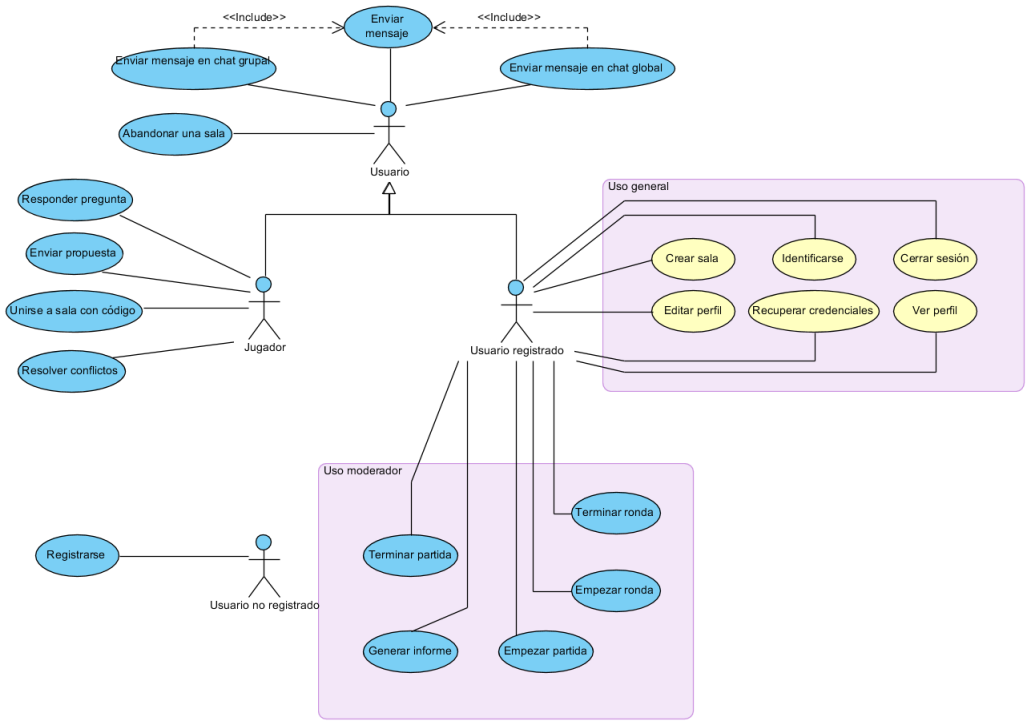

Figura 2.7: Diagrama de casos de uso.

2.3.2. Descripción de los casos de uso

Las Tablas de la 2.24 a la 2.43 contiene la descripción de los casos de uso presentados en el diagrama de la Figura 2.7.

Tabla 2.24: CU01 - Registrarse

ITEM	VALUE
UseCase	Registrarse
Actor	Usuario no registrado
Precondition	
Postcondition	El actor posee una cuenta y pasa a ser usuario registrado
Base Sequence	1. El usuario elige crear una cuenta 2. El sistema pide al usuario los datos necesarios (nombre, apellido, fecha de nacimiento, correo electrónico, país, nickname y contraseña). 3. El usuario rellena los campos de registro y los envía. 4. El sistema comprueba los datos y registra e inicia sesión al usuario.
Exception Sequence	4.1 Los datos son erróneos (ver tabla de Errores de datos), el sistema vuelve al paso 3.

Tabla 2.25: CU02 - Unirse a una sala con código

ITEM	VALUE
UseCase	Unirse a una sala con código
Actor	Jugador
Precondition	
Postcondition	El jugador entra en la Partida y cambia a estado 'sinGrupo'
Base Sequence	1. El usuario introduce el código de la sala. 2. El sistema confirma el código y muestra la pantalla de selección de nick. 3. El usuario escribe su nick. 4. El sistema comprueba el nick e introduce al jugador en la sala.
Exception Sequence	2.1 El código de la sala no corresponde con ninguna sala activa o es un dato erróneo (ver tabla de Errores de datos), se vuelve al paso 1. 4.1 El nick seleccionado esta repetido o es un dato erróneo (ver tabla Errores de datos), se vuelve al paso 3.

Tabla 2.26: CU03 - Responder pregunta

ITEM	VALUE
UseCase	Responder pregunta
Actor	Jugador
Precondition	El jugador esta en estado 'respondiendoPreguntas'
Postcondition	El jugador permanece en estado 'respondiendoPreguntas' salvo que haya sido la última pregunta
Base Sequence	1. El usuario selecciona una de las posibles respuestas. 2. El sistema guarda la respuesta y actualiza la pagina. 3. El usuario pasa a la siguiente pregunta. 4. El sistema comprueba el estado del formulario y muestra la siguiente pregunta.
Exception Sequence	4.1 El sistema detecta que ha sido la última pregunta y muestra el resumen.

Tabla 2.27: CU04 - Enviar mensaje

ITEM	VALUE
UseCase	Enviar mensaje
Actor	Usuario
Precondition	
Postcondition	El mensaje se muestra en chat
Base Sequence	1. El usuario escribe el mensaje 2. El sistema comprueba el mensaje, y actualiza el chat.
Exception Sequence	1.1 El mensaje es un dato erróneo (ver tabla de Errores de datos), se informa del error y se vuelve al paso 1.

Tabla 2.28: CU05 - Enviar mensaje en el chat grupal

ITEM	VALUE
UseCase	Enviar mensaje en chat grupal
Actor	Usuario
Precondition	
Postcondition	El mensaje se muestra en chat grupal con la información sobre la pregunta
Base Sequence	1. El usuario selecciona el chat grupal. 2. El sistema muestra el chat y solicita la pregunta sobre la que se va a comentar, y la posible respuesta así como el sentido de la opinión que se va a emitir (argumento positivo o negativo). 3. El usuario selecciona la pregunta, la respuesta y el sentido de la opinión (argumento positivo o negativo). 4. El sistema solicita el mensaje (la opinión o argumento del usuario sobre esa opción en la pregunta). 5. Se realiza el caso de uso «CU04 - Enviar mensaje».
Exception Sequence	3.1 El usuario selecciona el chat de sala, el caso de uso queda sin efecto.

Tabla 2.29: CU06 - Enviar mensaje en chat global

ITEM	VALUE
UseCase	Enviar mensaje en chat global
Actor	Usuario
Precondition	
Postcondition	El mensaje se muestra en chat global
Base Sequence	1. El usuario selecciona el chat grupal. 2. El sistema muestra el chat y solicita el mensaje. 3. Se realiza el caso de uso «CU04 - Enviar mensaje».
Exception Sequence	1.1 El usuario selecciona el chat de sala, el caso de uso queda sin efecto.

Tabla 2.30: CU07 - Enviar propuesta

ITEM	VALUE
UseCase	Enviar propuesta
Actor	Jugador
Precondition	Se han respondido todas las preguntas y no hay conflictos. El jugador está en estado 'rellenandoTramaNarrativa'
Postcondition	El jugador pasa a estado 'pendienteEnvio'
Base Sequence	1. El usuario rellena la trama narrativa. 2. El sistema comprueba el mensaje, cambia el estado del jugador a 'pendienteEnvio'.
Exception Sequence	2.1 La trama narrativa es un dato erróneo (ver tabla de Errores de datos), se informa del error y se vuelve al paso 1.

Tabla 2.31: CU08 - Resolver conflictos

ITEM	VALUE
UseCase	Resolver conflictos
Actor	Jugador
Precondition	Se han respondido todas las preguntas y hay conflictos. El jugador está en estado 'rellenandoTramaNarrativa'
Postcondition	Se resuelven todos los conflictos. El jugador vuelve al estado 'rellenandoTramaNarrativa'
Base Sequence	1. El usuario selecciona la pregunta a resolver. 2. El sistema cambia el estado del jugador a 'revisandoPreguntas', muestra la pregunta y pide una respuesta. 3. El usuario selecciona respuesta. 4. El sistema guarda la respuesta y pide un argumento. 5. El usuario introduce un argumento. 6. El sistema comprueba el mensaje y lo muestra en el chat grupal. 7. El usuario selecciona la siguiente pregunta. 8. El sistema comprueba que no hay más preguntas y pasa a la página de resumen.
Exception Sequence	6.1 El argumento es un dato erróneo (ver tabla de Errores de datos), se informa del error y se vuelve al paso 4. 8.1 El conflicto no esta solucionado, el sistema vuelve al paso 2. 8.2 Existen más preguntas con conflicto, el sistema vuelve al paso 2.

Tabla 2.32: CU09 - Abandonar una sala

ITEM	VALUE
UseCase	Abandonar una sala
Actor	Usuario
Precondition	El usuario forma parte de una sala en estado de espera.
Postcondition	El usuario se elimina.
Base Sequence	1. El usuario sale de la sala. 2. El sistema elimina al usuario de la base de datos.
Exception Sequence	

Tabla 2.33: CU10 - Identificarse

ITEM	VALUE
UseCase	Identificarse
Actor	Usuario registrado
Precondition	
Postcondition	
Base Sequence	1. El usuario selecciona la opción de inicio de sesión 2. El sistema solicita el correo y la contraseña. 3. El usuario introduce el correo y la contraseña. 4. El sistema comprueba los datos y muestra el menú principal.
Exception Sequence	4.1 Los datos son erróneos (ver tabla Errores de datos), el sistema vuelve al paso 3.

Tabla 2.34: CU11 - Cerrar sesión

ITEM	VALUE
UseCase	Cerrar sesión
Actor	Usuario registrado
Precondition	El usuario estaba identificado
Postcondition	
Base Sequence	1. El usuario solicita el cierre de sesión 2. El sistema pide confirmación de la acción. 3. El usuario confirma el cierre. 4. El sistema cierra la sesión y vuelve al inicio.
Exception Sequence	3.1 El usuario cancela el cierre, el caso de uso queda sin efecto.

Tabla 2.35: CU12 - Ver perfil

ITEM	VALUE
UseCase	Ver perfil
Actor	Usuario registrado
Precondition	El usuario esta identificado
Postcondition	
Base Sequence	1. El usuario selecciona el perfil. 2. El sistema muestra los datos del usuario.
Exception Sequence	

Tabla 2.36: CU13 - Editar perfil

ITEM	VALUE
UseCase	Editar perfil
Actor	Usuario registrado
Precondition	El usuario esta identificado
Postcondition	
Base Sequence	1. Se realiza el caso de uso «CU12 - Ver perfil». 2. El usuario indica que desea modificar los datos. 3. El sistema solicita los datos a modificar (apodo, contraseña y/o correo electrónico). 4. El usuario rellena los datos. 5. El sistema comprueba los datos, actualiza la información personal del usuario y vuelve al perfil.
Exception Sequence	5.1 Los datos son erróneos (ver tabla de Errores de datos), el sistema vuelve al paso 3.

Tabla 2.37: CU14 - Recuperar credenciales

ITEM	VALUE
UseCase	Recuperar credenciales
Actor	Usuario registrado
Precondition	El usuario posee una cuenta en el sistema
Postcondition	El usuario ha iniciado sesión
Base Sequence	1. El usuario indica que ya tiene una cuenta. 2. El sistema solicita las credenciales de inicio de sesión 3. El usuario indica que ha olvidado las credenciales. 4. El sistema solicita el correo electrónico. 5. El usuario introduce su correo. 6. El sistema comprueba el correo y envia un email a dicho correo con el enlace a la página de modificación de contraseña. 7. El usuario entra en enlace. 8. El sistema pide la nueva contraseña 2 veces. 9. El usuario rellena los campos. 10. El sistema comprueba la contraseña y pasa al menú principal.
Exception Sequence	6.1 El correo es erróneo o no existe, el sistema vuelve al paso 4. 10.1 Las contraseñas no cumplen los requisitos mínimos de seguridad (ver tabla de Requisitos de las contraseñas) o no son iguales, el sistema vuelve al paso 8.

Tabla 2.38: CU15 - Crear sala

ITEM	VALUE
UseCase	Crear sala
Actor	Usuario registrado
Precondition	El usuario ha iniciado sesión
Postcondition	El usuario ha creado una sala y esta en espera de jugadores
Base Sequence	1. El usuario selecciona la creación de una sala 2. El sistema solicita los parámetros de la sala. 3. El usuario rellena los campos de la sala. 4. El sistema comprueba los datos y cambia a la sala de espera.
Exception Sequence	3.1 El usuario cancela la creación de sala, el caso de uso queda sin efecto.
	4.1 Los datos son erróneos (ver tabla de Errores de datos), el sistema vuelve al paso 2.

Tabla 2.39: CU16 - Empezar partida

ITEM	VALUE
UseCase	Empezar partida
Actor	Usuario registrado
Precondition	El usuario ha creado una sala y hay mínimo un jugador por grupo
Postcondition	Se inicia la Partida para todos los jugadores, los jugadores pasan a estado 'respondiendoPreguntas'
Base Sequence	1. El usuario indica que quiere empezar la Partida 2. El sistema cambia al usuario registrado a la sala de control y a los jugadores a las preguntas, cambiando el estado de éstos a 'respondiendo-Preguntas'.
Exception Sequence	1.1 El usuario cancela la Partida, el caso de uso queda sin efecto.

Tabla 2.40: CU17 - Empezar una ronda

ITEM	VALUE
UseCase	Empezar una ronda
Actor	Usuario registrado
Precondition	El usuario se encuentra en una Partida empezada
Postcondition	
Base Sequence	1. El usuario selecciona el inicio de ronda 2. El sistema genera una nueva ronda y cambia el estado de todos los jugadores activos a 'respondiendoPreguntas'.
Exception Sequence	

Tabla 2.41: CU18 - Terminar una ronda

ITEM	VALUE
UseCase	Terminar una ronda
Actor	Usuario registrada
Precondition	El usuario se encuentra en una Partida empezada y en ronda activa
Postcondition	
Base Sequence	1. El usuario selecciona el fin de ronda 2. El sistema popula el tiempo de fin de ronda, comprueba si los grupos han terminado, genera los resultados de los grupos y cambia el estado de todos los jugadores activos a 'viendoResultados'.
Exception Sequence	

Tabla 2.42: CU19 - Terminar partida

ITEM	VALUE
UseCase	Terminar partida
Actor	Usuario registrado
Precondition	El usuario se encuentra en una Partida empezada
Postcondition	La Partida finaliza y todos los jugadores pasan al estado 'viendoResultadosFinales'
Base Sequence	1. El usuario indica que desea finalizar la Partida 2. El sistema solicita confirmación. 3. El usuario confirma la acción. 4. El sistema cierra la sala, cambia el estado de los jugadores a 'viendoResultadosFinales' muestra los resultados globales a los jugadores y al usuario.
Exception Sequence	3.1 El usuario cancela la acción, el caso de uso queda sin efecto.

Tabla 2.43: CU20 - Generar informe

ITEM	VALUE
UseCase	Generar informe
Actor	Usuario registrado
Precondition	El usuario acaba de terminar una Partida
Postcondition	Se genera y descarga en la máquina del usuario un informe
Base Sequence	1. El usuario indica que desea imprimir los resultados. 2. El sistema genera un informe con los detalles de la Partida e inicia una descarga del documento en la máquina del usuario.
Exception Sequence	

Capítulo 3

Planificación y seguimiento

3.1. Planificación inicial

Para la planificación del proyecto, se ha escogido el Proceso Unificado de Desarrollo Software (UP) [17]. Esta es una metodología estándar basada en componentes e interfaces, que junto con el Lenguaje Unificado de Modelado (UML) [11] conforman la técnica más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Una de las características que define UP es su desarrollo iterativo e incremental. Estos incrementos pueden ser en diferentes fases como por ejemplo en las fases de definición de los requisitos del sistema, o en fases en las que se desarrolla funcionalidad en el código. Las fases que componen el ciclo de vida de un proyecto en esta metodología son:

Fase de concepción o inicio: Definir visión, objetivos y alcance del proyecto. En esta etapa se pretende obtener una lista de casos de uso y otra de factores de riesgo del proyecto. Esta fase esta prevista que termine a principios de febrero.

Fase de elaboración: Completar el análisis de los casos de uso, definición de arquitectura y generación de una aplicación ejecutable. En este proyecto también se contempla en esta fase la obtención de los resultados de las simulaciones de Vensim junto con la transformación de sus datos a un formato legible. La fecha prevista para finalizar esta fase es finales de marzo.

Fase de construcción: Compuesta por varias iteraciones, en esta fase se irán implementando los casos de uso definidos previamente, aumentando la funcionalidad de la aplicación. El fin de esta fase esta previsto para mediados de mayo.

Fase de transición: Despliegue de la aplicación y sistema en fase de producción. En el caso de este proyecto, sería el despliegue en una maquina del servidor, pudiendo realizar consultas a la misma. Esta fase esta prevista que termine a finales de mayo.

La aplicación de estas fases en la calendarización del proyecto puede verse en el diagrama de Gantt de la figura 3.1. En cada fase se diferenciado dos tareas a realizar para el desarrollo de la etapa.

Para asegurar un correcto desarrollo del proyecto, se han establecido una serie de reuniones semanales con los tutores para la revisión del trabajo y correcciones en caso necesario. Por otro lado, se ha acordado reunirse con los miembros de ECO-Profes en febrero para una presentación de la aplicación, aclaración de conceptos y obtención de feed-back.

3.1.1. Diagrama de Gantt inicial

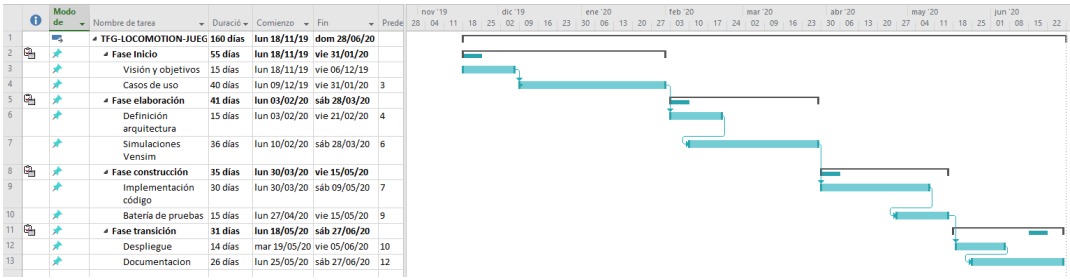

Figura 3.1: Diagrama de Gantt del proyecto inicial (Parte 1).

3.2. Modificación de la planificación inicial

Teniendo en cuenta los problemas surgidos a lo largo del proyecto, descritos en la sección 3.6, se ha realizado un nuevo diagrama de Gantt (figura 3.2) para la nueva planificación del proyecto. En esta nueva planificación se ha tenido en cuenta lo que faltaba por desarrollar del proyecto y se ha dividido por fases de implementación.

3.2.1. Diagrama de Gantt modificado

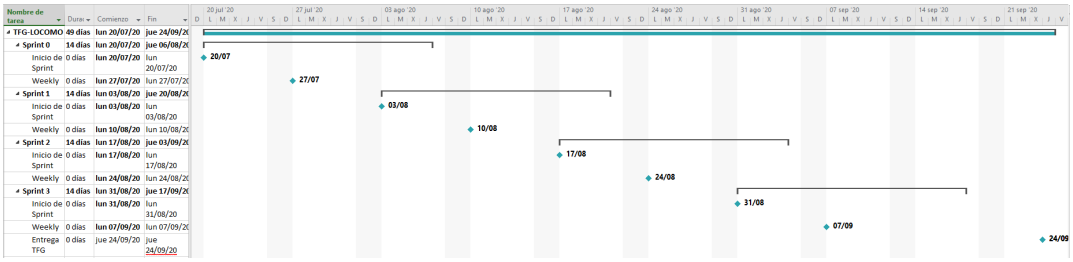

Figura 3.2: Diagrama de Gantt modificado.

3.3. Presupuesto inicial

A continuación se presentará el presupuesto inicial del proyecto, por tipos de gastos.

Concepto	Cantidad	Precio por unidad	Total
Trabajo humano en horas	338	15,00 €	5070 €
Dominio	0	-	0 €
Alojamiento	0	-	0 €
Licencia Visual Paradigm	1	86,67 €	86,67 €
Licencia Mockplus	1	174,23 €	174,23 €
Licencia Vensim	1	77,92 €	77,92 €
Licencia MS Project	1	849,00 €	849,00 €
			6.257,82 €

Tabla 3.1: Presupuesto inicial

El coste en licencias Software corresponde a la licencia permanente básica de cada uno de los especificados. Visual Paradigm, MS Project y Vensim son gratuitos para alumnos adheridos a la UVa por lo que finalmente no ha repercutido en el coste del proyecto. Por otro lado, se ha utilizado el mes de prueba para la licencia de Mockplus, ahorrándose también su coste. El proyecto se ha realizado sin gastos añadidos por software. El único software de pago utilizado ha sido amortizado con anterioridad y su pago no ha ido destinado al proyecto en exclusiva. El resto de software utilizado es gratuito o bien ofrece algún tipo de licencia gratuita. En la Tabla 3.2 se listan los que han sido utilizados en el proyecto. Respecto al pago del trabajo humano, se ha contado con la beca ofertada por el proyecto LOCOMOTION, que otorgaba una financiación de 285,51 € mensuales, durante 8 meses, sumando un total de 2.284,08 €.

3.4. Software utilizado

Nombre	Descripción	Uso dado en el proyecto
Sublime Text 3	Editor de texto y código fuente	Se utilizará para la programación de los scripts utilizados.
IntelliJ IDEA	IDE	Se utilizará para el desarrollo del código.
MS Project	Software de gestión de proyectos	Se utilizará para la planificación del calendario, proyecto y la creación de los diagramas de Gantt.
Postman	Plataforma de desarrollo de APIs	Se utilizará para la prueba de la aplicación.
Visual Paradigm 15.1	Editor UML	Se utilizará para la creación de diagramas UML.
Vensim 7.3.1	Simulador de modelos	Se utilizará para la obtención de resultados del modelo MEDEAS.
Mockplus	Software de diseño de interfaces	Se utilizará para los mockups de la aplicación web, como prototipo de las interfaces.
MySQL Workbench	Software de administración de bases de datos	Se utilizará para diseñar y guardar la información de las partidas y los jugadores.
MongoDB Compass	Software de administración de bases de datos no relacionales	Se utilizará para guardar los resultados y patrones de las simulaciones.
Tomcat	Servidor de aplicaciones	Se utilizará como servidor que soporte la lógica de la aplicación web.

Tabla 3.2: Software utilizado

3.5. Análisis de riesgos

Identificador	R01 - Fallo en la planificación del proyecto
Descripción	Los tiempos especificados para las tareas a realizar en el proyecto son insuficientes.
Impacto	Crítico
Probabilidad	75 %
Plan de mitigación	Tratar de realizar las tareas en los tiempos establecidos, utilizando los weeklies para buscar las tareas que requieran mayor esfuerzo.
Plan de contingencia	Priorización de las tareas, reducción del campo de trabajo y replanificación del proyecto.

Tabla 3.3: Riesgo de fallo en la planificación

Identificador	R02 - Indisponibilidad del desarrollador
Descripción	El desarrollador no puede dedicar el tiempo suficiente al proyecto ya sea por enfermedad, otras tareas o asuntos personales, no llegando a las fechas de entrega.
Impacto	Catastrófico
Probabilidad	50 %
Plan de mitigación	Establecer un colchón de tiempo para suplir los posibles días de baja.
Plan de contingencia	Replanificación de los sprints, del proyecto y atrasar la fecha de entrega.

Tabla 3.4: Riesgo de indisponibilidad del desarrollador

Identificador	R03 - Curva de aprendizaje demasiado larga
Descripción	El tiempo necesario para controlar las tecnologías a utilizar es demasiado alto, o su complejidad escapa las capacidades del desarrollador, suponiendo una demora en el proyecto.
Impacto	Crítico
Probabilidad	80 %
Plan de mitigación	Realización de cursos diversos, e investigación en las tecnologías de forma previa, haciendo el aprendizaje desde el inicio del proyecto.
Plan de contingencia	Replanificación de los sprints, solapando tareas.

Tabla 3.5: Riesgo de curva de aprendizaje demasiado larga

Identificador	R04 - Tecnología no válida
Descripción	Las tecnologías escogidas para la realización del proyecto no son capaces de cumplir con todos los requisitos especificados.
Impacto	Catastrófico
Probabilidad	15 %
Plan de mitigación	Investigación de cada tecnología y del proyecto, asegurandose de que son capaces de cubrir todos los objetivos especificados.
Plan de contingencia	Búsqueda de tecnologías de respaldo que si puedan realizar dichos objetivos, y reorganizar el proyecto.

Tabla 3.6: Riesgo de tecnología no válida

Identificador	R05 - Ausencia de comunicación
Descripción	Mala comunicación entre el alumno y los tutores, provocando malentendidos y retrasos en el proyecto
Impacto	Crítico
Probabilidad	20 %
Plan de mitigación	Reuniones semanales (sprints y weeklies) donde se especifiquen las tareas de la semana y se hagan revisiones de lo ya realizado.
Plan de contingencia	Replanificación de las tareas, y aumentando el número de reuniones, realizando un seguimiento más cercano.

Tabla 3.7: Riesgo de ausencia de comunicación

3.6. Seguimiento del Proyecto y dificultades encontradas

Estos han sido las tareas realizadas a lo largo de los cinco primeros meses del proyecto:

- **Diciembre:** Reunión con David Álvárez, informarse del proyecto, del funcionamiento de la herramienta Vensim, y de la versión actual de Crossroads. Primer borrador de los requisitos del sistema y búsqueda de posibles tecnologías para el proyecto.
- **Enero:** Segundo borrador de los requisitos, definiciones del diccionario de datos y búsqueda de referencias.
- **Febrero:** Reunión con el grupo de GEEDS para probar la versión actual del juego, realización de pruebas con scripts Python para la automatización de simulaciones de Vensim. Creación de los Mockups de las páginas web de la aplicación para la reunión con el grupo GEEDS. Presentación a miembros del grupo GEEDS y de ECO-Profes.
- **Marzo:** Preparación de las simulaciones de Vensim y redimensión del proyecto. Definición de los casos de uso, y transformación de las pruebas.
- **Abril:** Arreglos a los casos de uso, requisitos, y creación de los diagramas de arquitectura y despliegue. Formación en SpringBoot.

El trabajo realizado durante el primer mes de diciembre fue productivo. La familiarización con las herramientas, la búsqueda de tecnologías y las preparaciones de las simulaciones iban en el tiempo esperado. A mayores, la presentación del proyecto al grupo GEEDS y a los miembros de ECO-Profes con los mockups marcó la línea que se debía seguir. Aunque se empezaban a presentar problemas en el diseño de la aplicación, sobre todo en la parte de requisitos funcionales.

El proceso de aprendizaje de las librerías necesarias para la creación de los script Python, utilizados para la obtención de las simulaciones y su posterior parseo, fue lenta. La necesidad de hacer múltiples pruebas y el tiempo que requería realizarlas, y teniendo en cuenta que el tiempo para el proyecto estaba limitado por otras asignaturas, se retrasó la obtención de los datos de las simulaciones.

A partir del 10 de marzo, todo el trabajo empezó a sufrir retrasos, falta de horas y de calidad de trabajo. Debido a la pandemia mundial, el COVID-19, que ha obligado a la población a realizar cuarentena en sus domicilios, haciendo necesario el trabajo desde casa. La enseñanza pasó de ser presencial a ser completamente online, y requiriendo un mayor número de horas de trabajo, teniendo que priorizar asignaturas frente al proyecto. A mayores, por temas personales y de salud, la productividad cayó en picado a mediados de abril, no habiendo recuperado el trabajo hasta inicios del mes de Julio. Teniendo en cuenta todos los problemas y riesgos, se decidió hacer una modificación a la planificación inicial del proyecto, aplazando la entrega al mes de septiembre.

Posteriormente, en septiembre, debido a problemas en la economía de la familia del estudiante, las horas se vieron reducidas por tener que incorporarse al mundo laboral a media

jornada. Se propuso pedir un aplazamiento de la entrega a la primera semana de diciembre, junto con una nueva planificación por hitos:

- Entrega de los datos de las simulaciones. Para el 2 de septiembre.
- Creación de las APIs del usuario registrado. Para el 1 de noviembre.
- Creación de las APIs del jugador. Para el 10 de noviembre.
- Creación de las APIs de obtención de datos. Para el 20 de noviembre.
- Documentación del proyecto. Para el 23 de noviembre

En cuanto a los problemas técnicos, el desarrollo sufrió grandes bloqueos a la hora de implementar. Por un lado debido al desconocimiento del framework SpringBoot y de la base de datos no relacional MongoDB. Para entender el funcionamiento de estas herramientas fue necesaria mucha formación, por ejemplo la proporcionada en el foro de ProgrammingTechie [31], en el cual se explica como funciona SpringBoot, la conexión con MySQL y MongoDB.

Tras este aprendizaje, la implantación del código se realizó de manera casi mecánica: se diseñaba una API, se programaba, se probaba en Postman, corregían posibles fallos y se pasaba a la siguiente. Para finales de noviembre ya se tenían todas las llamadas API REST terminadas y probadas.

Todos los cambios de planificación y problemas surgidos a lo largo del proyecto han supuesto un aumento en las horas dedicadas al proyecto, estimándose cerca de las 450 horas. Este aumento de horas supondría un aumento del presupuesto del proyecto de 1680 €.

Capítulo 4

Análisis

4.1. Modelado de la interacción

En esta sección se detalla la interacción que realizarán los usuarios de la aplicación, diferenciando entre jugadores y el moderador. El modelado de la interacción se realiza mediante diagramas de máquinas de estado y se muestran en las Figuras 4.1, 4.2 y 4.3.

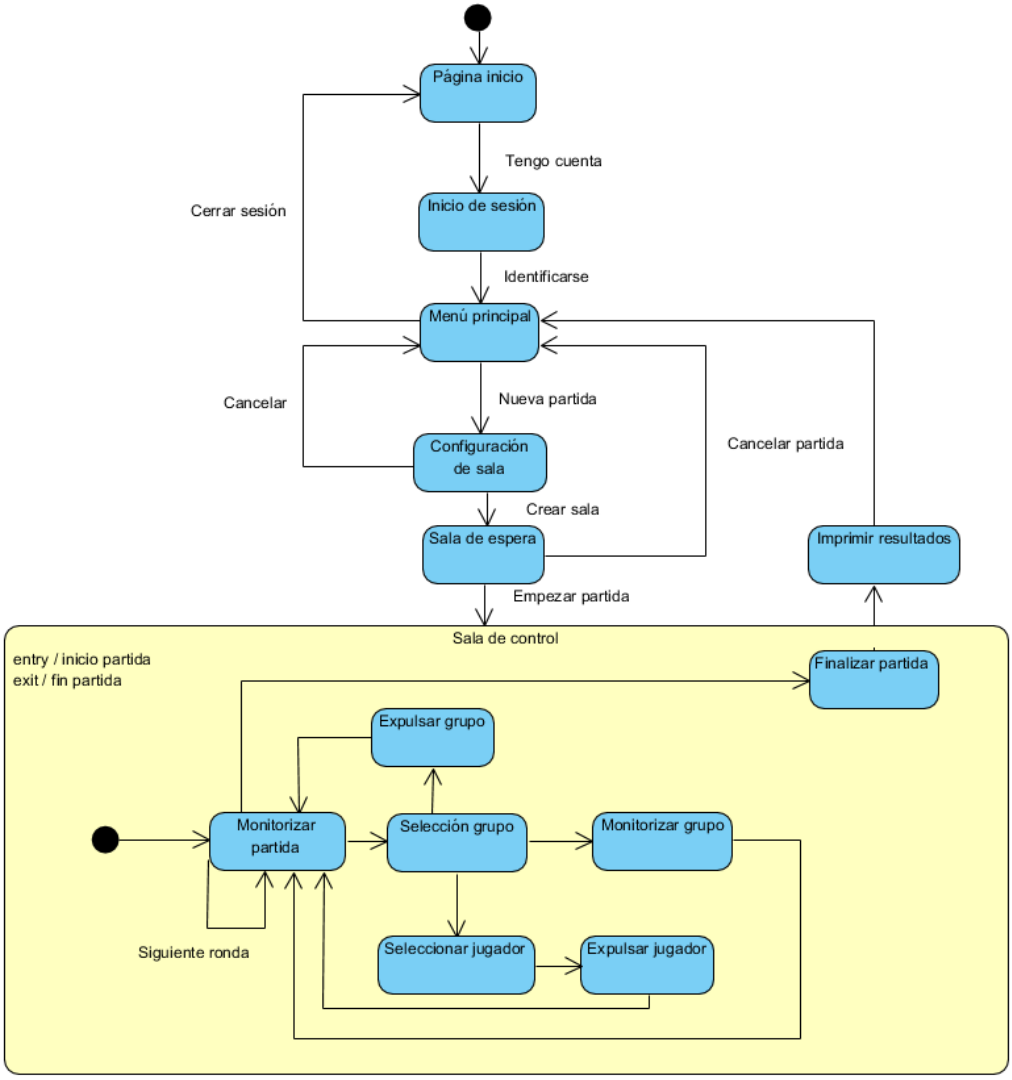

Figura 4.1: Modelado de la interacción de un usuario con el rol 'Moderador'

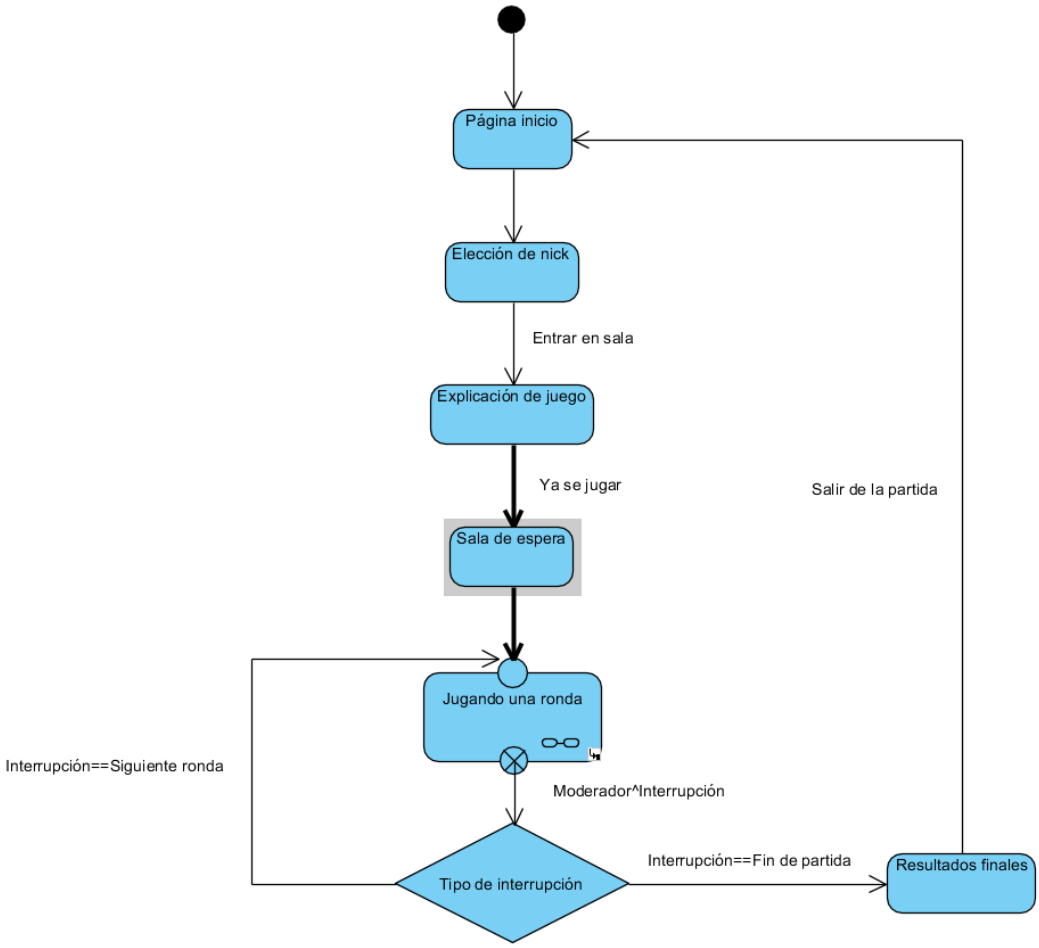

Figura 4.2: Modelado de la interacción de un usuario con el rol 'Jugador'

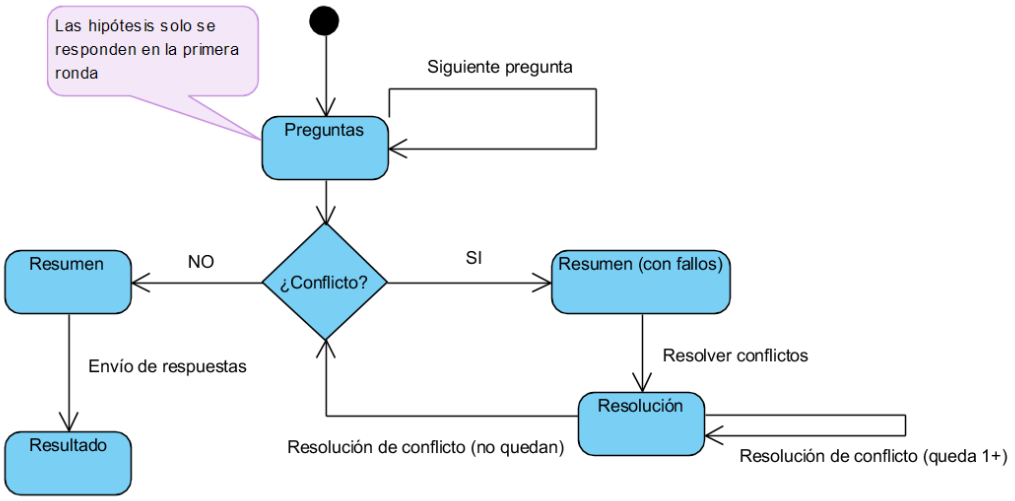

Figura 4.3: Modelado de la interacción de un usuario con el rol de 'Jugador' mientras está jugando una ronda en una partida.

4.2. Modelado conceptual

A partir del análisis de los requisitos de información, funcionales, y de la interacción, así como de otras necesidades que se prevén en el futuro se realiza un modelado conceptual del dominio que se refleja en el diagrama de clases que muestra la Figura 4.4.

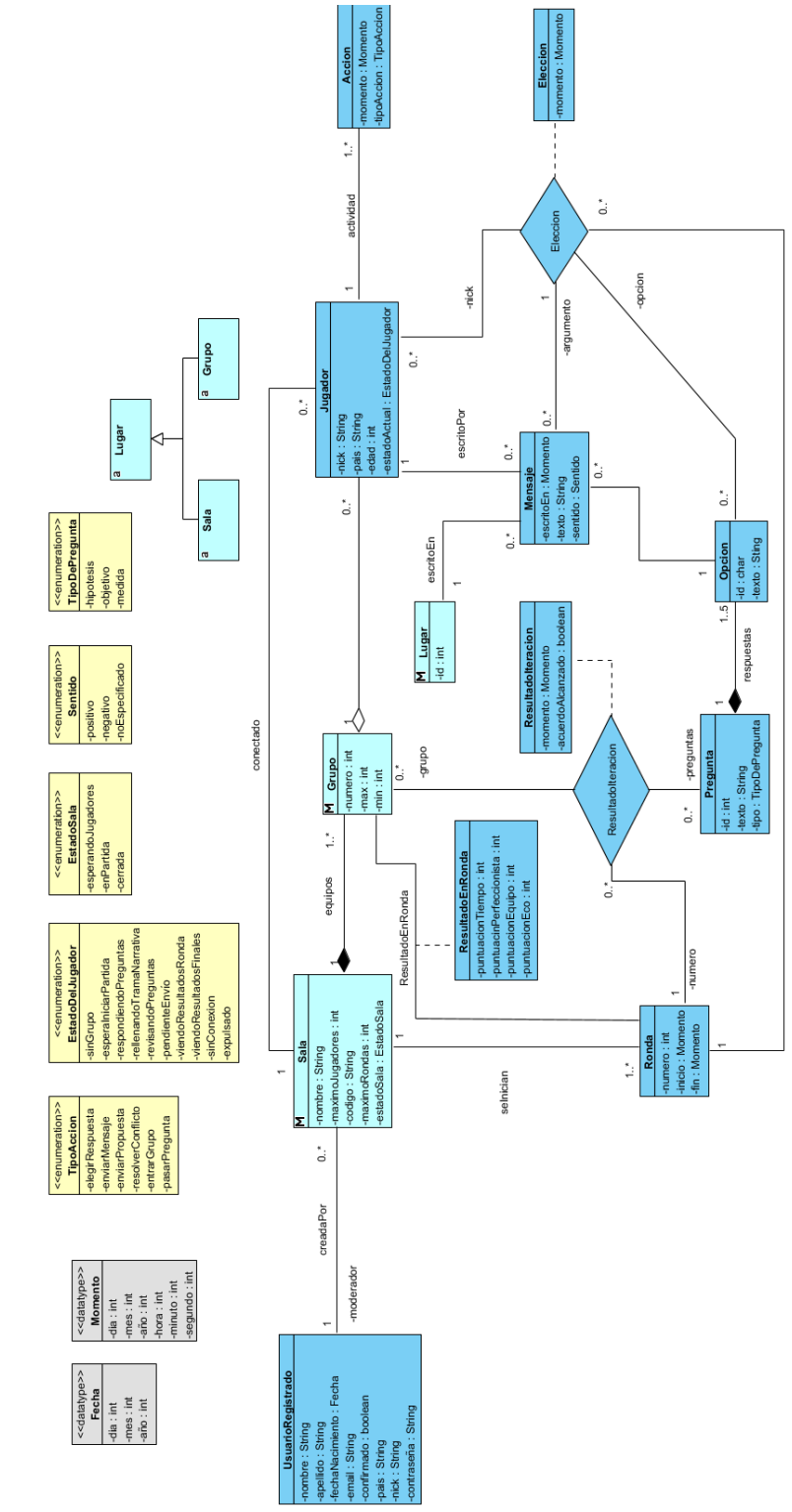

Figura 4.4: Diagrama de clases que representa el modelo conceptual del dominio

La clase Jugador describe objetos que pasan por diferentes estados. Estos cambios son generados a lo largo de la partida. Este objeto se modela como una máquina de estados que se muestra en la Figura 4.5.

El estudio conjunto del jugador como máquina de estados, del diseño de la interacción basado en máquinas de estado y de los tipos enumerados definidos en el modelado del dominio que describen el estado del jugador y los tipos de acciones nos permite concluir sobre la consistencia del análisis.

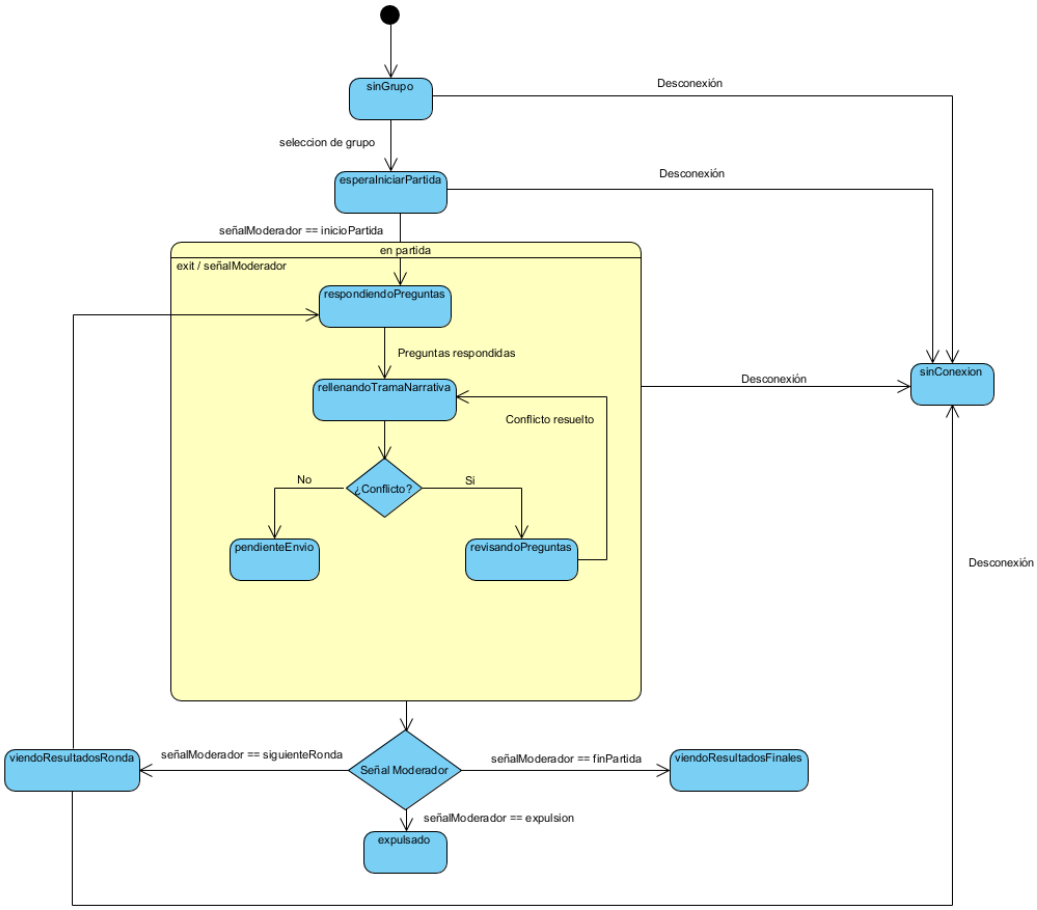

Figura 4.5: Análisis de los objetos de la clase Jugador como una máquina de estados

Capítulo 5

Tecnologías utilizadas

En este capítulo se repasarán las tecnologías utilizadas durante el desarrollo de este proyecto, tanto para gestionarlo como para diseñarlo e implementarlo.

5.1. Gestión del proyecto

5.1.1. Memoria

Para la documentación del proyecto, se ha utilizado *Overleaf* [19], editor de texto colaborativo de *LaTeX* [25] en la nube. Es utilizado comúnmente para la redacción de documentos científicos y de proyectos universitarios, gracias a su facilidad de uso, la posibilidad de editar el documento simultáneamente entre sus integrantes y por la versatilidad que ofrece *LaTeX*. Respecto a los documentos, se ha utilizado tanto el propio *Overleaf* como una carpeta compartida del OneDrive de la Universidad.

5.1.2. Comunicación

Para la comunicación entre las distintas partes del proyecto se ha utilizado principalmente el correo electrónico, la aplicación móvil Whatsapp [10], el software empresarial gratuito RocketChat [7], y la plataforma de videoconferencia Skype [22]. Sobre todo desde el inicio de la pandemia, ha sido más importante utilizar medios como éstos, debido a la necesidad de mantener las distancias de seguridad y evitar los espacios cerrados.

5.1.3. Planificación

Para la planificación del proyecto se ha hecho uso de la herramienta proporcionada por Microsoft Office, MS Project [21]. Éste software permite gestionar proyectos desde su línea temporal hasta el presupuesto necesario que requerirá. Principalmente utilizada para la ilustración de las figuras 3.1 y 3.2, también se ha utilizado para marcar las reuniones y fechas relevantes que han surgido, como las reuniones mencionadas en la Sección 3.6.

5.2. Desarrollo

5.2.1. SpringBoot

Para la implementación de esta aplicación en el lado del servidor se ha decidido utilizar SpringBoot [24]. Es un framework Open Source pensado para facilitar la creación de aplicaciones en Java, Kotlin y Groovy. En nuestro caso, Java como lenguaje. Spring cuenta con una modularización modificable, pudiendo utilizar sólo aquellas partes que sean necesarias para nuestra aplicación, enfocando esta tecnología a la Programación Orientada a Aspectos o *AOP*. En la figura 5.1 se muestra un esquema con los módulos que proporciona Spring por defecto.

La inyección de dependencias que proporciona Spring, facilita la programación, permitiendo a los componentes que conforman la aplicación depender de interfaces, ahorrando código y eliminando el acoplamiento. Gracias a las múltiples librerías que posee, se ha facilitado las conexiones con las bases de datos y la creación de documentos del proyecto. Aún teniendo todas las comodidades que ofrece, Springboot nos permite si queremos granular más el código, pudiendo hacer, por ejemplo, consultas mucho más complejas con una simple etiqueta.

SpringBoot es una versión mas simplificada de Spring, gracias a sus dos principales mecanismos

- Contenedor de aplicaciones integrado: SpringBoot permite compilar las aplicaciones Web en un ".jar". En esta aplicación no se ha hecho uso de este mecanismo.
- Starters: Proporciona una serie de dependencias llamadas *starters*, que se pueden añadir al proyecto. Al añadirlo, nos proporcionará las dependencias necesarias tanto de Spring como de software de terceros que necesitemos. Los más relevantes en este proyecto a utilizar son los necesarios para los controladores REST, documentación en Swagger y para la creación de documentos PDF.

5.2.2. Tomcat

Tomcat [1] es un contenedor de servlets desarrollado con java utilizado principalmente como servidor web. Para este proyecto, se ha utilizado un servidor embebido para el despliegue

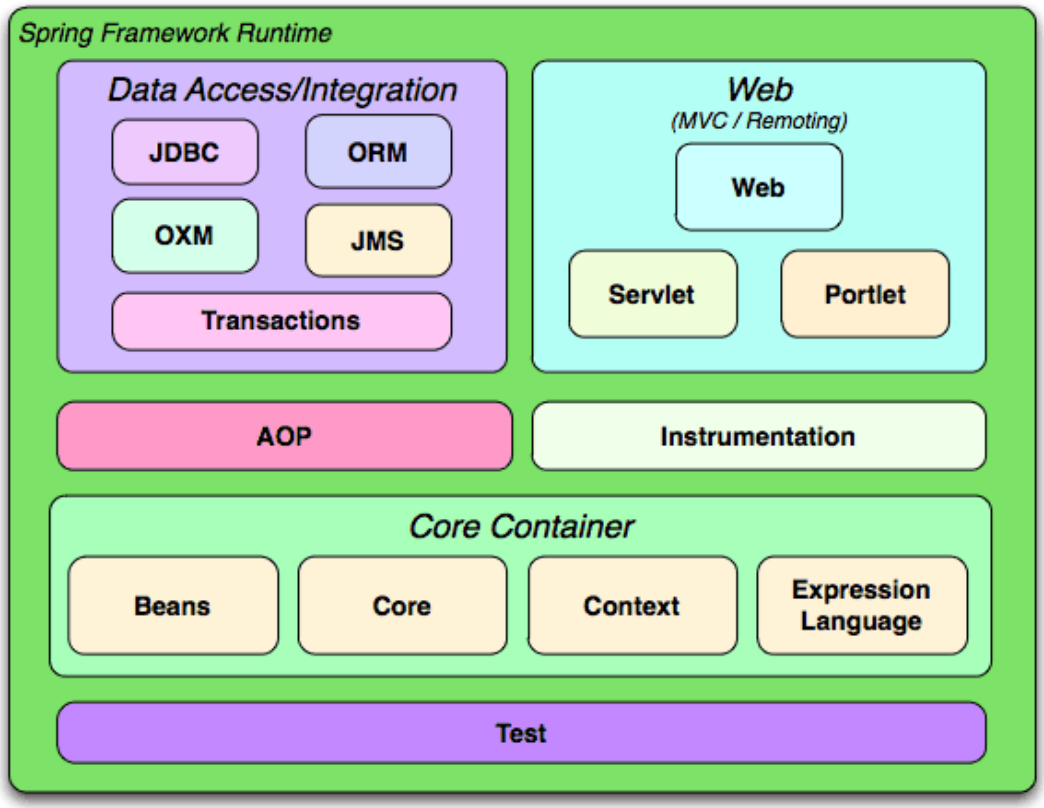

Figura 5.1: Ilustración de los diferentes módulos de Spring

de la aplicación. Es el servidor más común para el desarrollo con Springboot.

5.2.3. MySQL Workbench

MySQL Workbench [8] es una herramienta de visualización, diseño, desarrollo y administración de bases de datos MySQL. Esta herramienta proporciona herramientas de modelado de datos, consultas y control sobre las tablas de las bases de datos, pudiendo gestionar cómodamente la información contenida, y facilitando el uso de la misma. Proporciona instrumentos para mejorar el rendimiento de las aplicaciones MySQL.

Su uso principal en este proyecto ha sido la monitorización de los datos, y el uso de sus editores de tablas para la administración de las mismas.

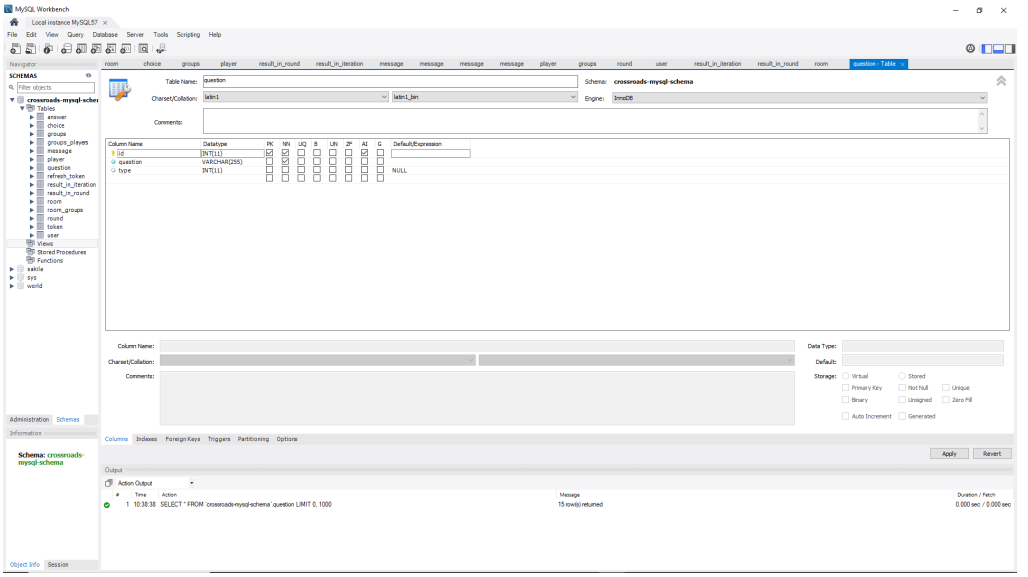

Figura 5.2: Interfaz de MySQL Workbench

5.2.4. MongoDB Compass

MongoDB Compass [16] es una interfaz gráfica para la visualización y gestión de una base de datos MongoDB. Al igual que MySQL Workbench, MongoDB Compass permite el modelado de datos, consultas y control sobre las colecciones de datos. La necesidad de guardar gran cantidad de valores para las gráficas de los resultados de las simulaciones hizo que se utilizara una base de datos no relacional como MongoDB.

5.2.5. Postman

Postman [9] es una plataforma de desarrollo de APIs basado en el modelo de desarollo API First [33]. Esta aplicación permite la creación, monitorización y envío de peticiones HTTP a servicios REST mediante una interfaz gráfica. Éstas pueden ser guardadas y almacenadas en colecciones. Además, permite compartir las APIs y peticiones entre equipos de trabajo, creación de mockups, gestión de la documentación y generación de pruebas.

El uso que se le ha dado a esta potente herramienta en el proyecto ha sido principalmente para la prueba de las API RESTs, creando las peticiones HTTP pertinentes. En la figura 5.4 se muestra como se realizan las peticiones en la aplicación.

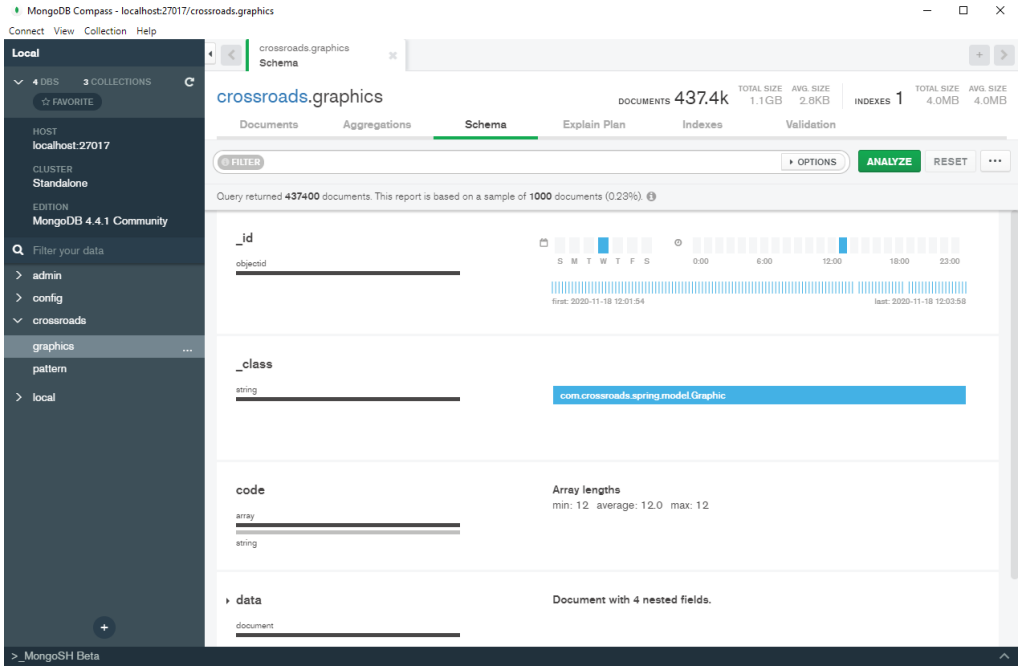

Figura 5.3: Interfaz de MongoDB Compass

5.2.6. IntelliJ IDEA

IntelliJ IDEA [18] es un IDE gratuito para programas informáticos. Soporta múltiples lenguajes de programación y tiene integración con Git, simplificando el control de versiones. Con funcionalidades como el control de versiones, de calidad asegurando que no exista código duplicado y ofreciendo la opción de refactorización en caso necesario, soporte de múltiples frameworks, servidores, hacen de este IDE una herramienta idónea para este proyecto.

5.2.7. SourceTree

Cliente GUI para manejar repositorios git y mercurial. Desarrollado por Atlassian, SourceTree [2] permite gestionar los repositorios de los proyectos de una manera muy sencilla y amigable, facilitando el control de versiones de nuestra aplicación. Permite un uso de las opciones del repositorio (realizar commits, pushear/pullear los cambios, crear y moverse entre ramas) facilmente, sin interferir en el IDE utilizado.

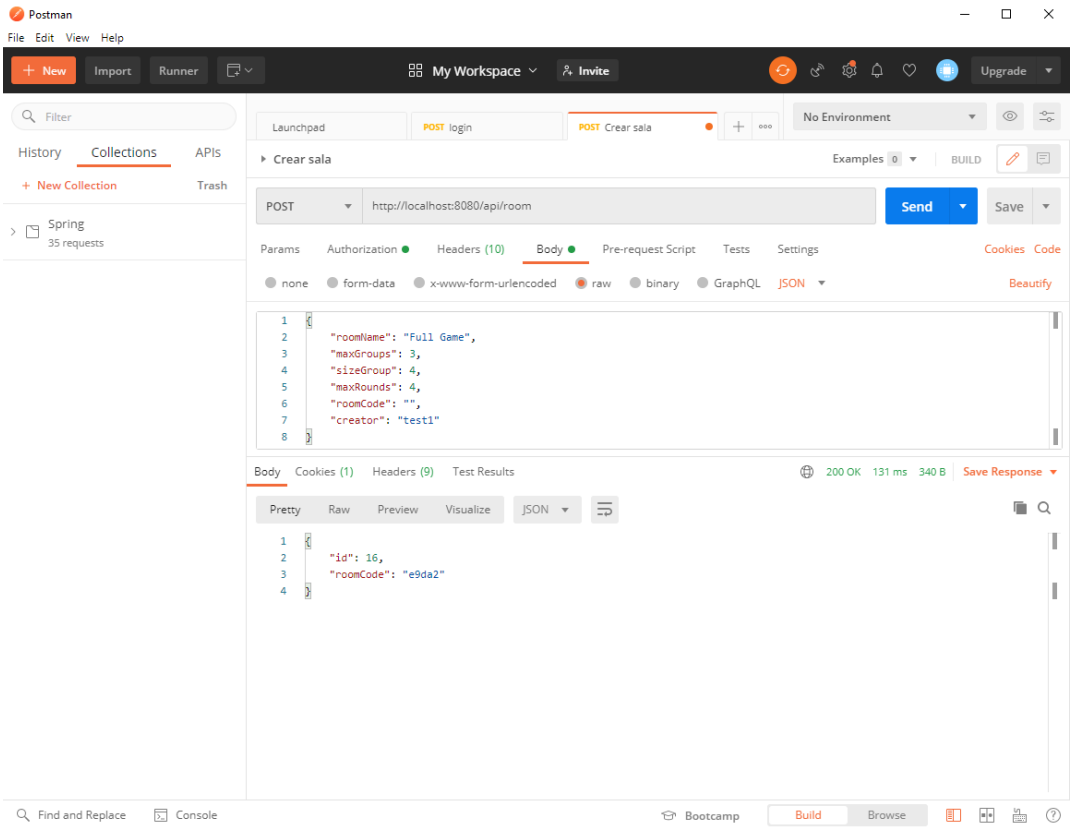

Figura 5.4: Interfaz de Postman

5.2.8. Sublime Text 3

Sublime Text 3 [14] es un editor de texto y de código fuente gratuito. Con un soporte para el reconocimiento de una amplia variedad de lenguajes, y posibilidad de instalar paquetes que aumenten su funcionalidad, es la herramienta utilizada para la programación de los scripts utilizados en Vensim, tanto para la obtención de los datos como para su posterior transformación.

Capítulo 6

Diseño

Tras haber analizado el proyecto, el siguiente paso es diseñar la aplicación. Teniendo en cuenta las tecnologías que se han definido en el capítulo anterior, se procede a definir la arquitectura, despliegue y APIs que compondrán el backend.

6.1. Mockups

Para la presentación del proyecto a los profesores y para la planificación del futuro front-end que dispondrá, se diseñaron los siguientes mockups.

Las figuras 6.1 y 6.3 son las salas de conexión e introducción de datos del jugador para entrar a la partida. En la siguiente ventana (Figura 6.3), se mostrará la información acerca de Crossroads II: quiénes lo conforman, objetivos, descripción del juego, funcionamiento del mismo y como se desarrollará.

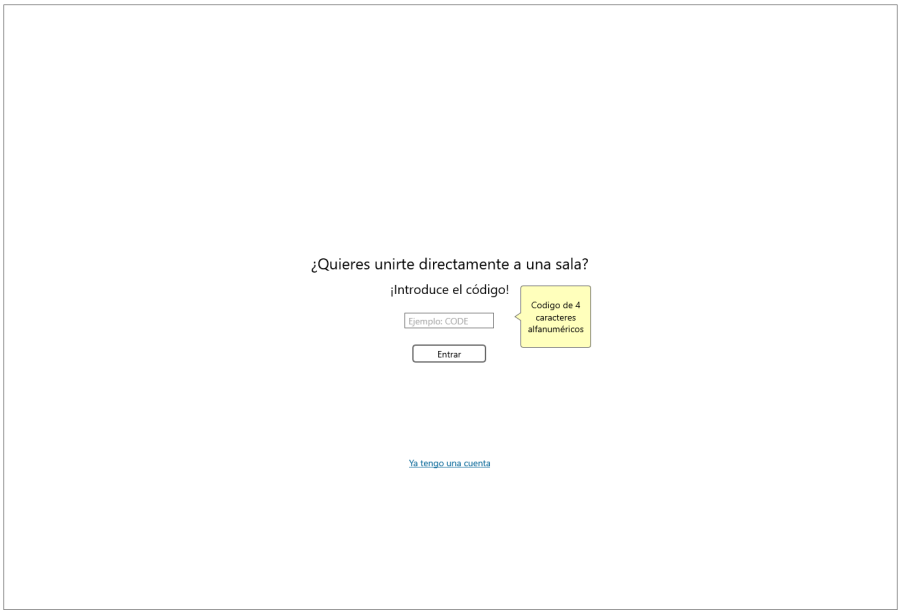

Figura 6.1: Ventana de Página de Inicio

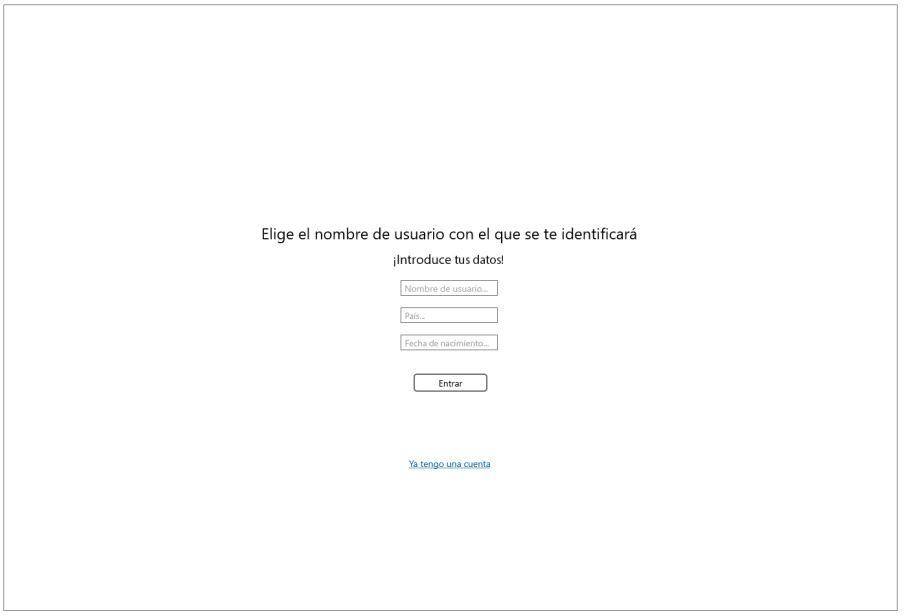

Figura 6.2: Ventana de Introducción de datos

Figura 6.3: Ventana de información para el jugador

Una vez dentro de la sala, los jugadores podrán escoger el grupo al que desean unirse en una ventana como la que se ve en la figura 6.4. Aquí esperarán hasta que el moderador de la partida inicie el juego.

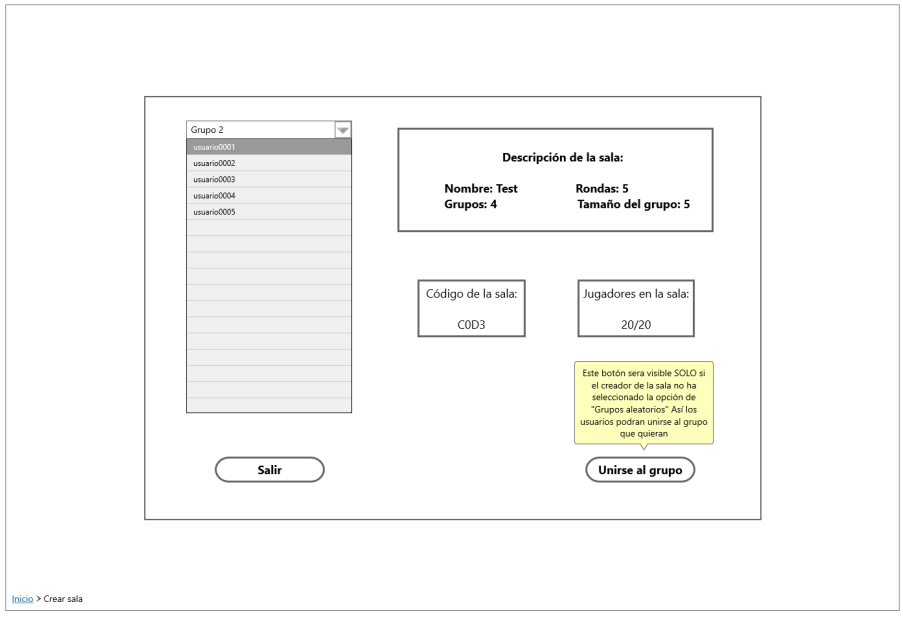

Figura 6.4: Ventana de espera del jugador

En cuanto comience la partida, los jugadores responderán a las preguntas (figura 6.5), argumentando sus elecciones. Una vez hayan respondido a todo, pasaran a ver la tabla de estado de la ronda (figura 6.6), donde podrán comprobar en que estado se encuentran sus compañeros y donde deberán redactar una pequeña trama narrativa explicando el por qué de sus respuestas. En caso de que exista conflicto en alguna pregunta, se pasaría a la ventana de la figura 6.5, donde se poblará una matriz con los argumentos a favor y en contra de cada respuesta. Estos argumentos son los mensajes que han enviado los jugadores a lo largo de la partida sobre cada una de las posibles elecciones. Además se podrá ver que opción está votando cada jugador.

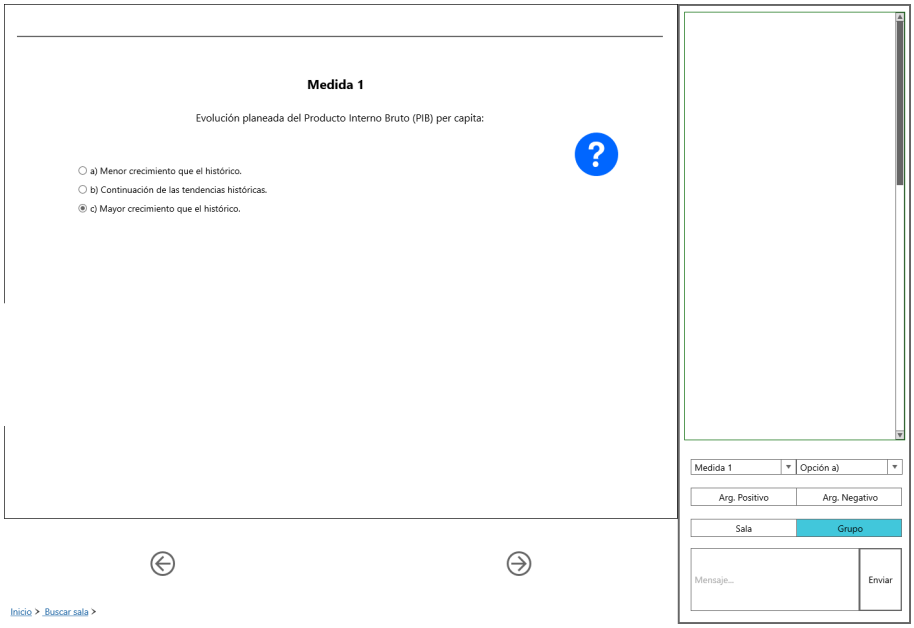

Figura 6.5: Ventana de pregunta

Figura 6.6: Ventana de resumen

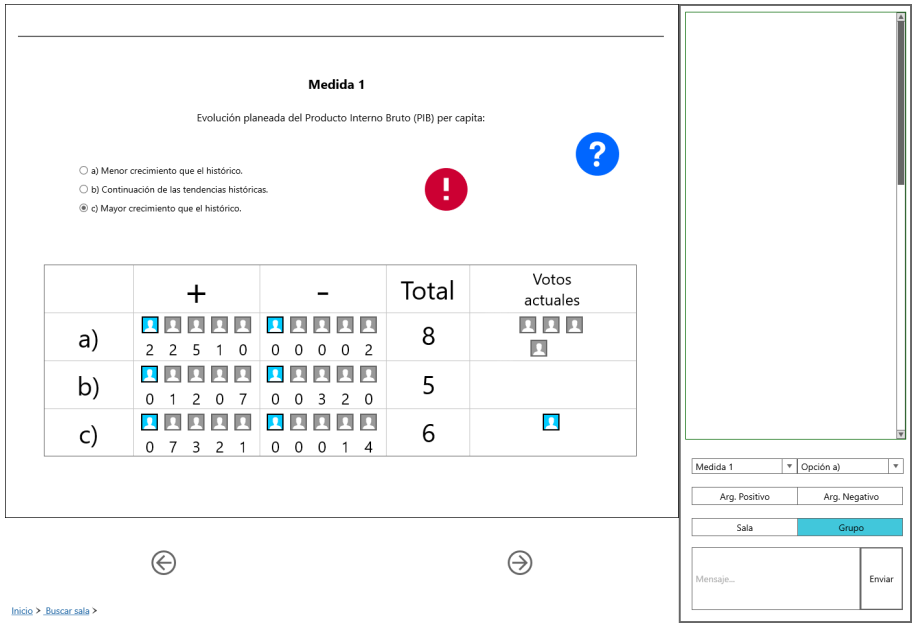

Figura 6.7: Ventana de conflicto

Una vez terminada la ronda, los jugadores podrán ver sus resultados (en caso de tenerlos) con la puntuación obtenida (figura 6.8). Cuando terminen la partida, podrán ver la ventana de la figura 6.9, con las estadísticas de todos los grupos a lo largo de las rondas.

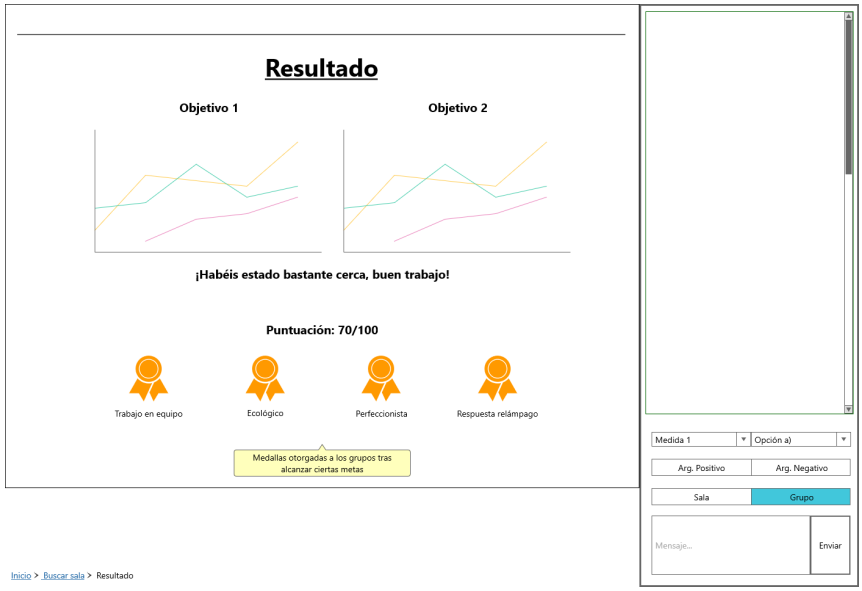

Figura 6.8: Ventana de resultado

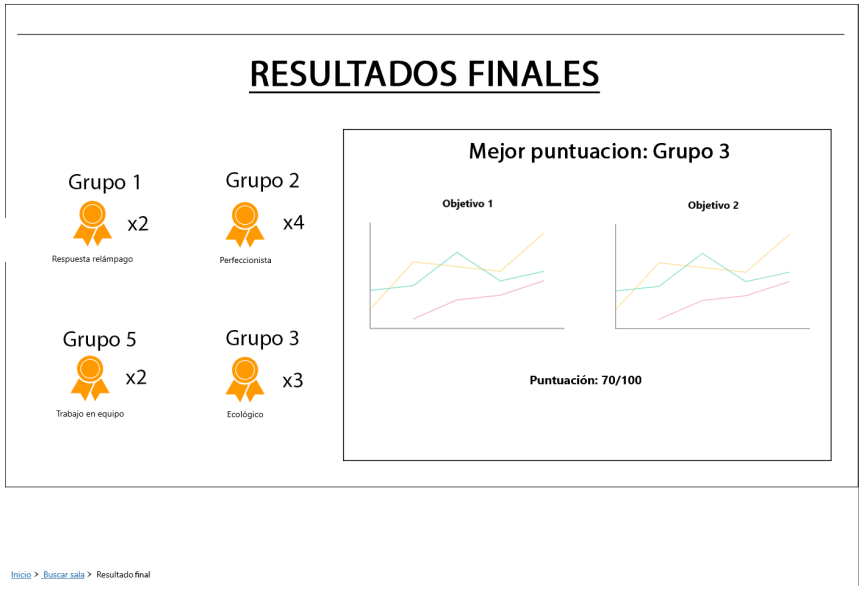

Figura 6.9: Ventana de resultado final

Las vistas de las figuras 6.10 y 6.11 indican los campos necesarios para la creación de una cuenta del usuario y su posterior inicio de sesión. En caso de que el usuario no recuerde sus credenciales, tendra una ventana de recuperación de cuenta como la que se ve en la figura 6.12. Una vez dentro el usuario verá el menú (figura 6.13), donde podrá acceder a su perfil (figura 6.14), a las salas que ha creado (figura 6.15) o crear una nueva sala (figura 6.16).

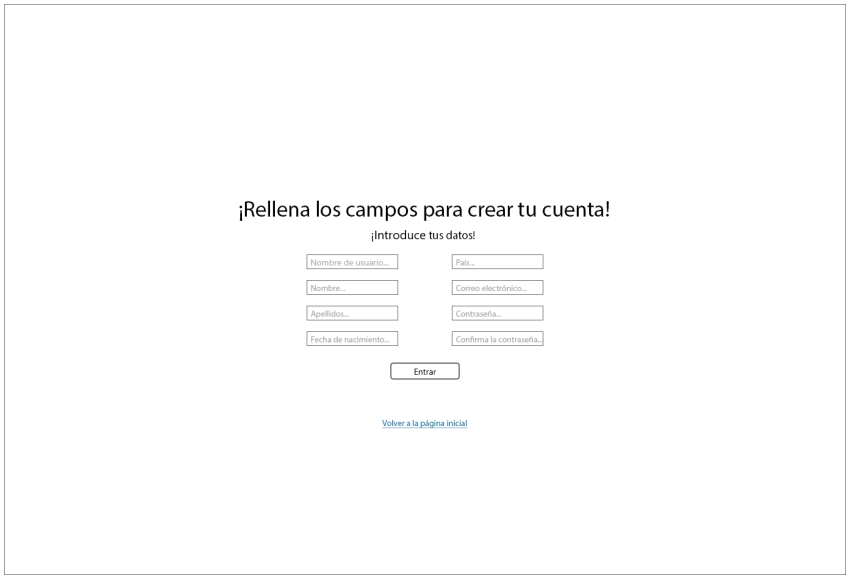

Figura 6.10: Ventana de registro

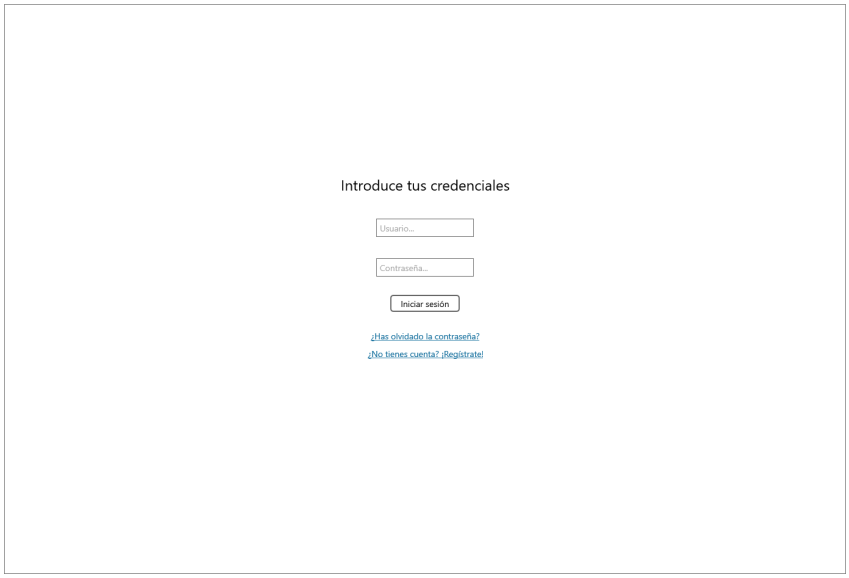

Figura 6.11: Ventana de inicio de sesión

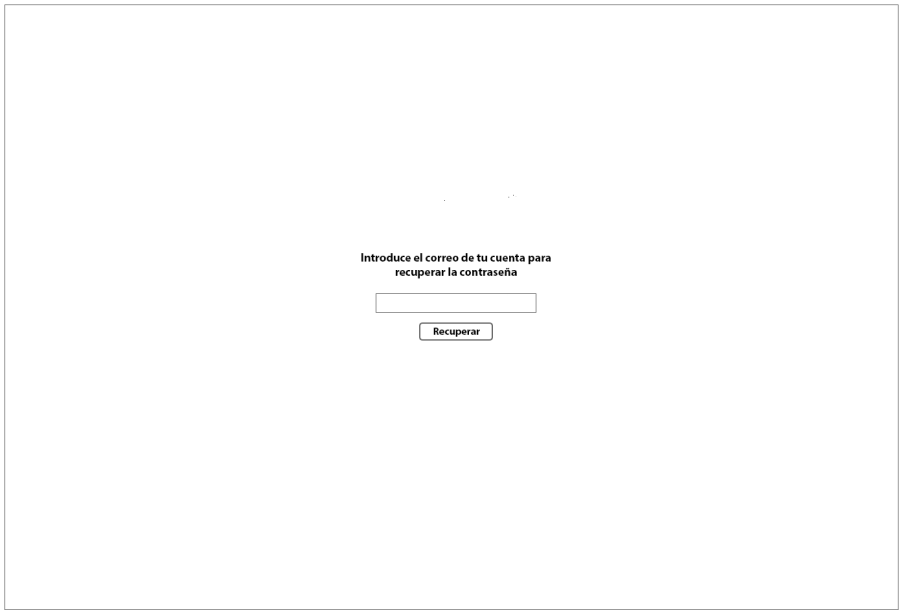

Figura 6.12: Ventana de recuperación de cuenta

Figura 6.13: Ventana de menú de usuario

Figura 6.14: Ventana de perfil del usuario

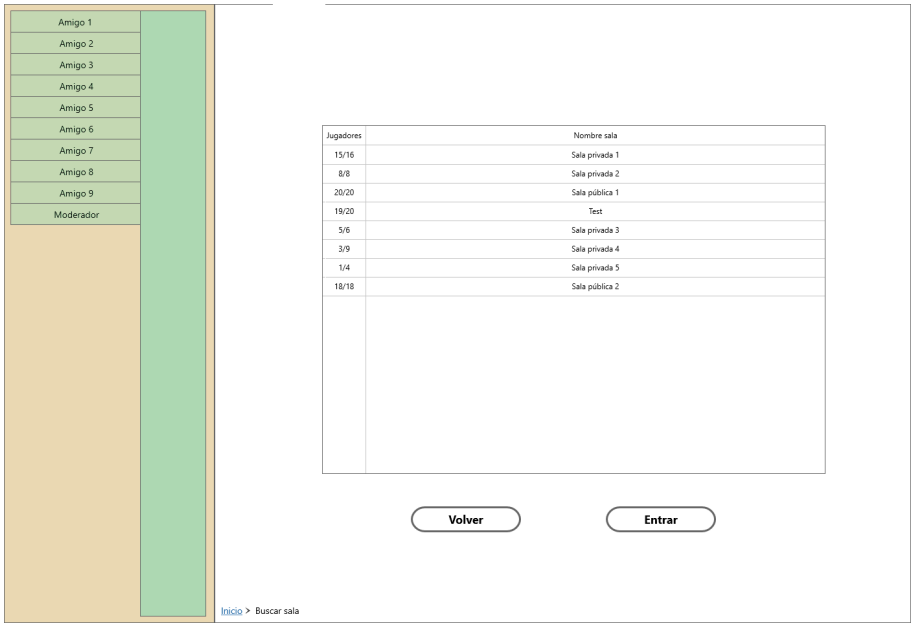

Figura 6.15: Ventana de búsqueda de salas

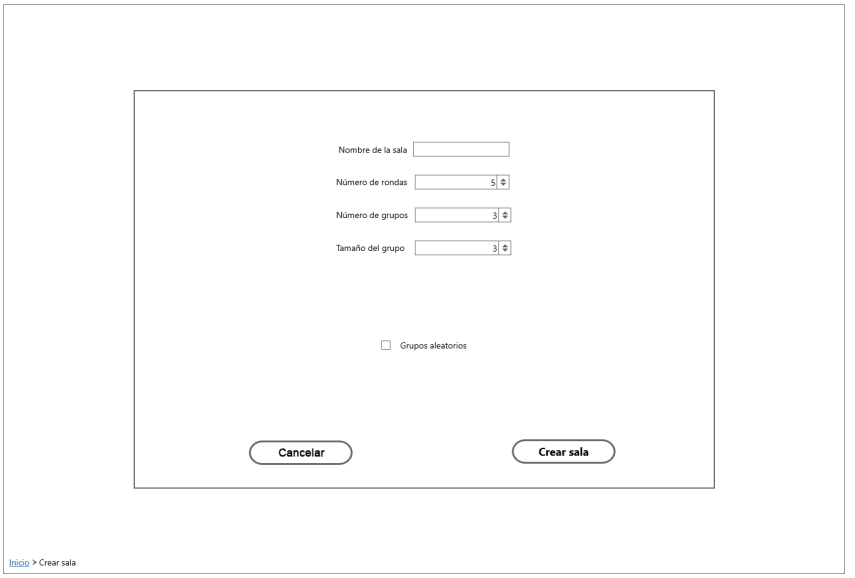

Figura 6.16: Ventana de crear sala

Tras crearla, tendrá una ventana similar a la de los jugadores (figura 6.17) en esperará a que los jugadores entren e iniciará la partida cuando considere. Mientras la partida este en curso, el moderador podrá coordinar el juego con las opciones que se pueden ver en la ventana de la figura 6.18.

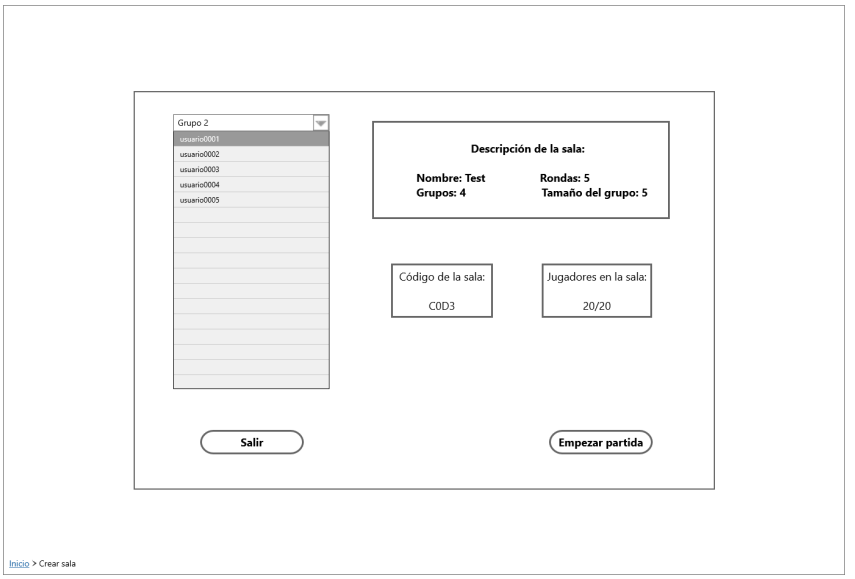

Figura 6.17: Ventana de espera de moderador

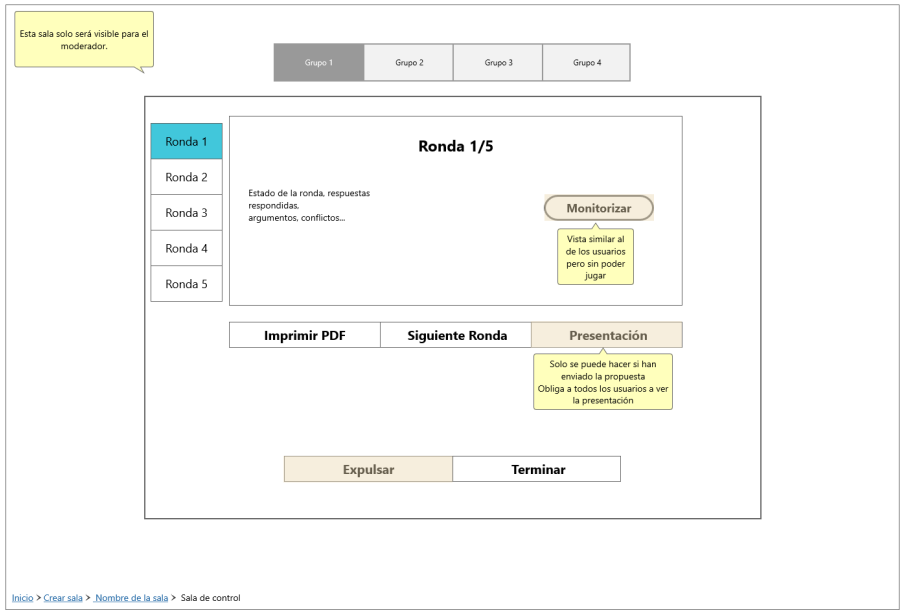

Figura 6.18: Ventana de control del moderador

6.2. Diseño de datos

En esta sección se ilustrará el modelado de datos de las bases de datos de MySQL y MongoDB.

6.2.1. MySQL

Se ha utilizado una base de datos relacional para el almacenado del modelo de la aplicación

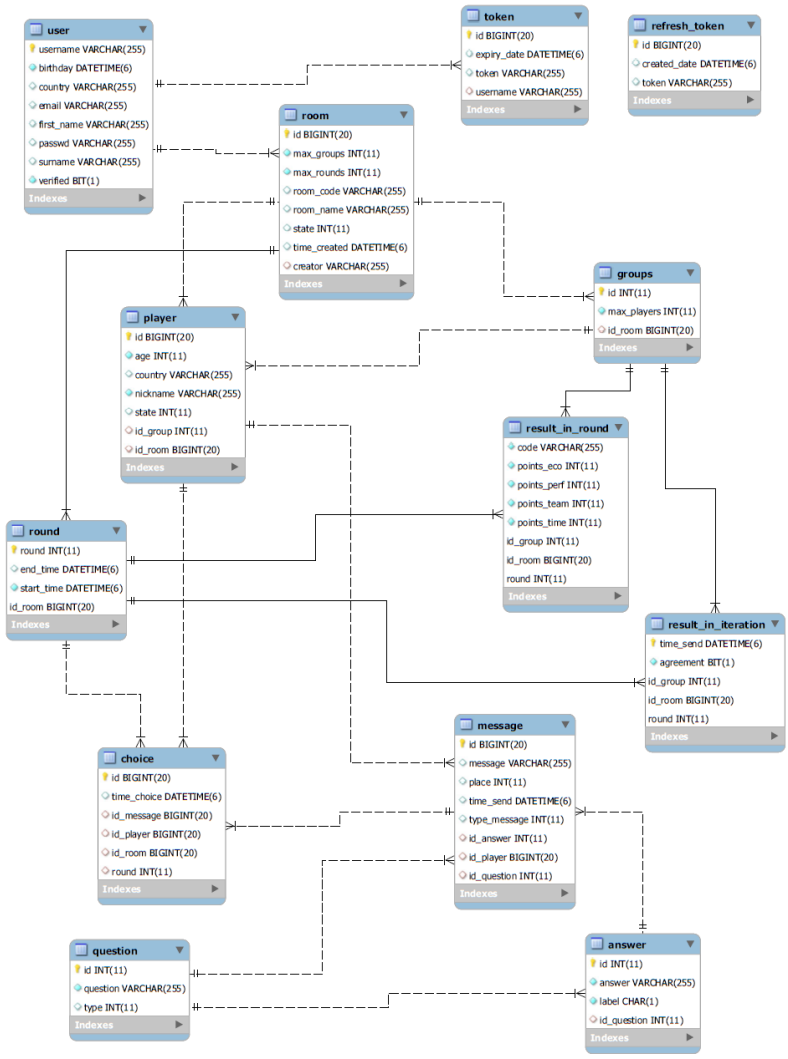

Figura 6.19: Diagrama de la base de datos de MySQL

6.2.2. MongoDB

Se ha utilizado una base de datos no relacional para el almacenado de los resultados de las simulaciones de *Vensim* y de los patrones que se producen en los resultados.

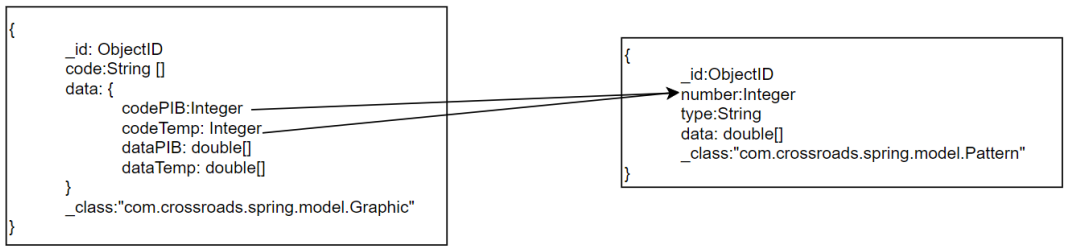

Figura 6.20: Diagrama de la base de datos de MongoDB

6.3. Arquitectura

6.3.1. Diseño de la arquitectura lógica

En la Figura 6.21, encontramos la arquitectura lógica definida para la aplicación, en este caso se ha optado por Cliente-Servidor. Se presenta el cliente, un navegador web, y por otro lado el servidor que contiene toda la lógica de negocio y las bases de datos. La comunicación se realiza a través de peticiones HTTP a una API REST. En este proyecto se diseñará e implementará toda la parte del servidor.

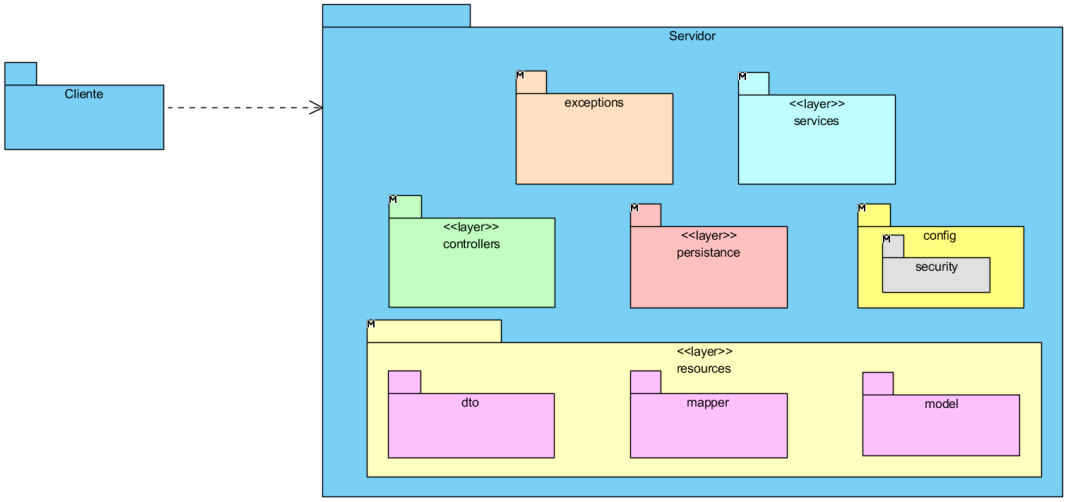

Figura 6.21: Diagrama de la arquitectura lógica de la aplicación

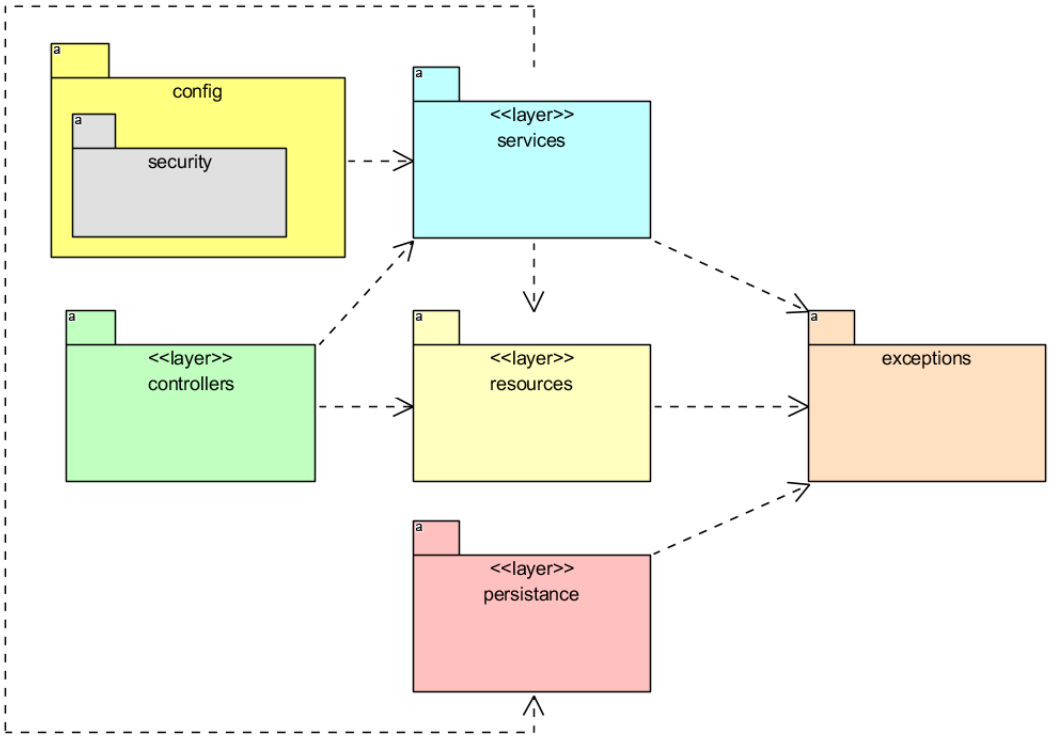

Figura 6.22: Diagrama de la arquitectura lógica del servidor

6.3.2. Diseño del despliegue

En la figura 6.23 se ilustra el diagrama de despliegue de la aplicación. La arquitectura del proyecto es cliente-servidor. En nuestro backend, podemos comprobar que tenemos el despliegue en un Apache Tomcat, conectado con las bases de datos MySQL y MongoDB a través de los drivers de SpringBoot. Las peticiones que se realizan al servidor son peticiones HTTP.

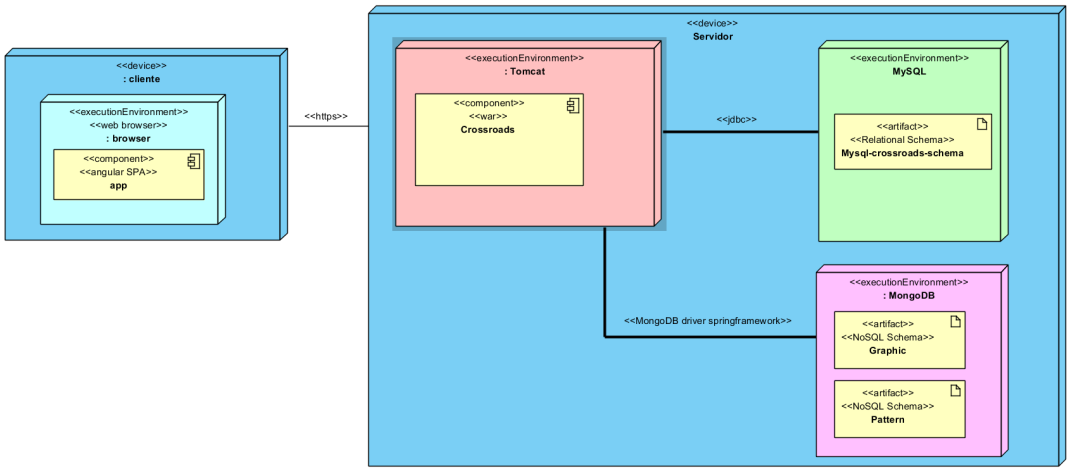

Figura 6.23: Diagrama de despliegue de la aplicación

6.4. Estructura de las API REST

En esta sección, utilizando la documentación generada por *Swagger* [26], se expondrán todas las llamadas API REST con su respectivos parámetros y resultados.

6.4.1. AuthController

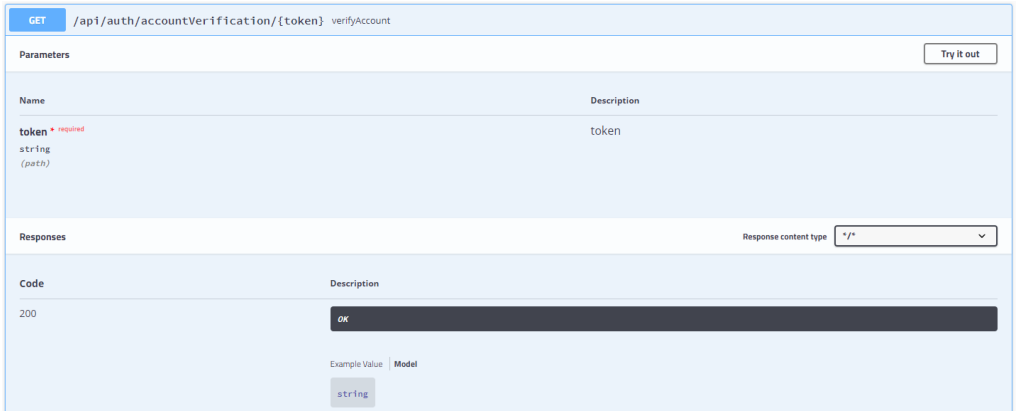

Figura 6.24: Esquema de la API verifyAccount

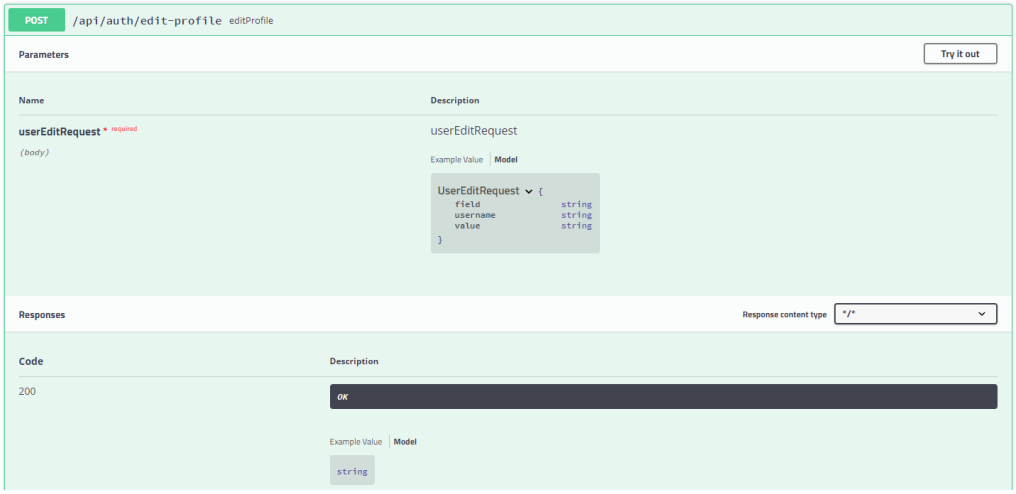

Figura 6.25: Esquema de la API editProfile

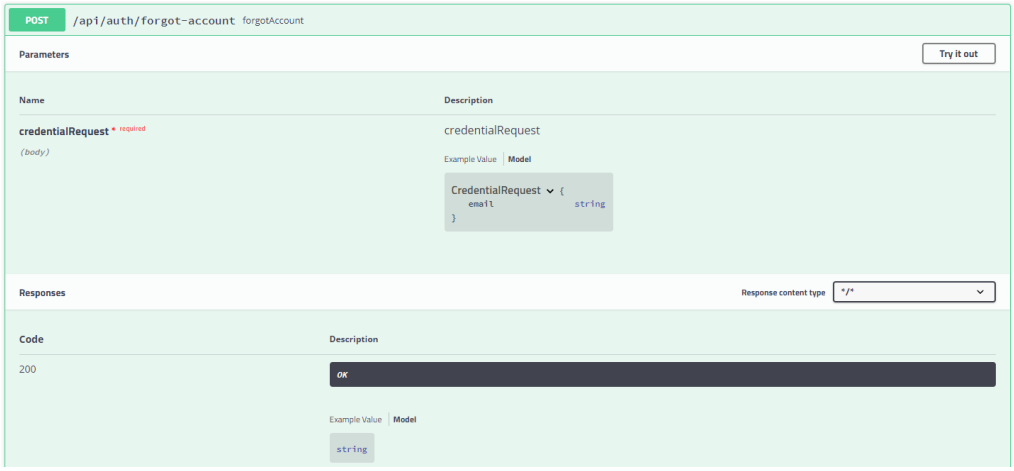

Figura 6.26: Esquema de la API forgotAccount

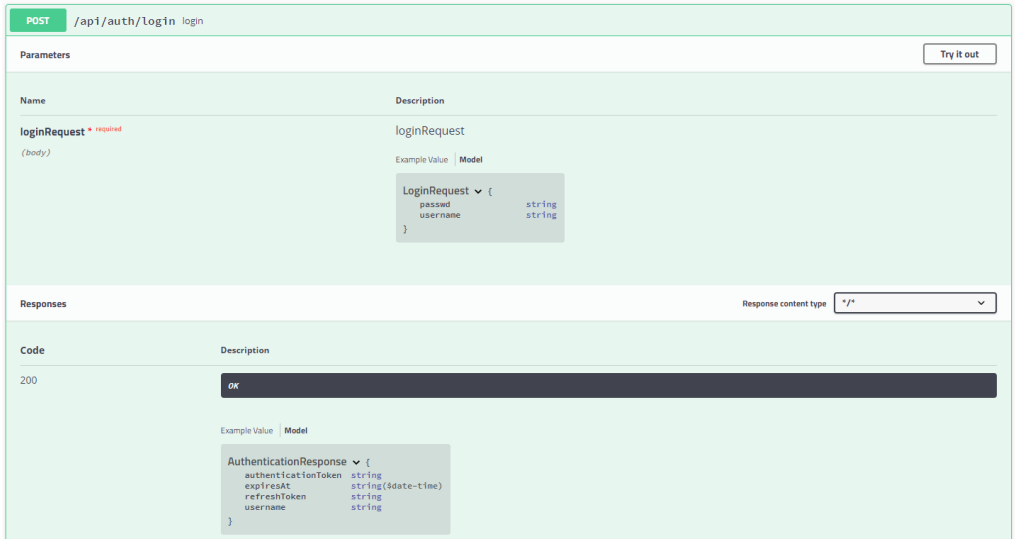

Figura 6.27: Esquema de la API login

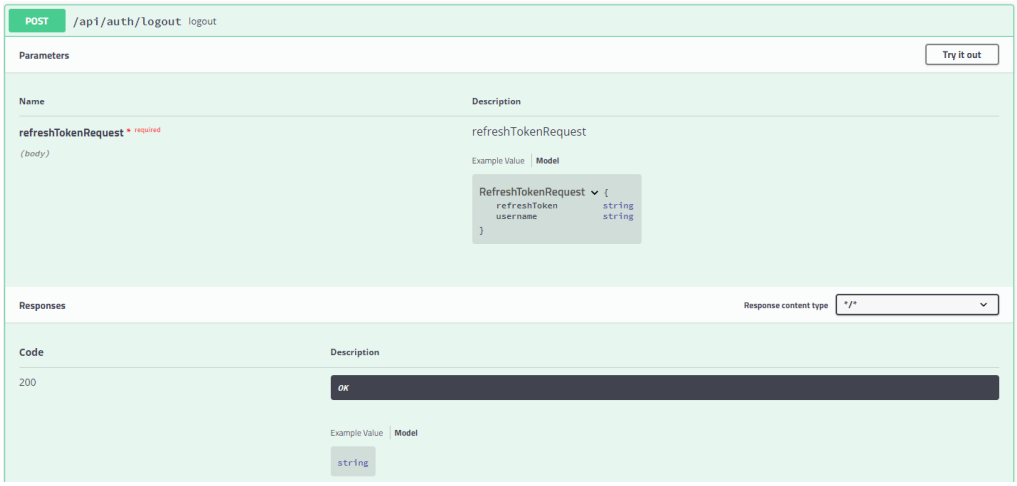

Figura 6.28: Esquema de la API logout

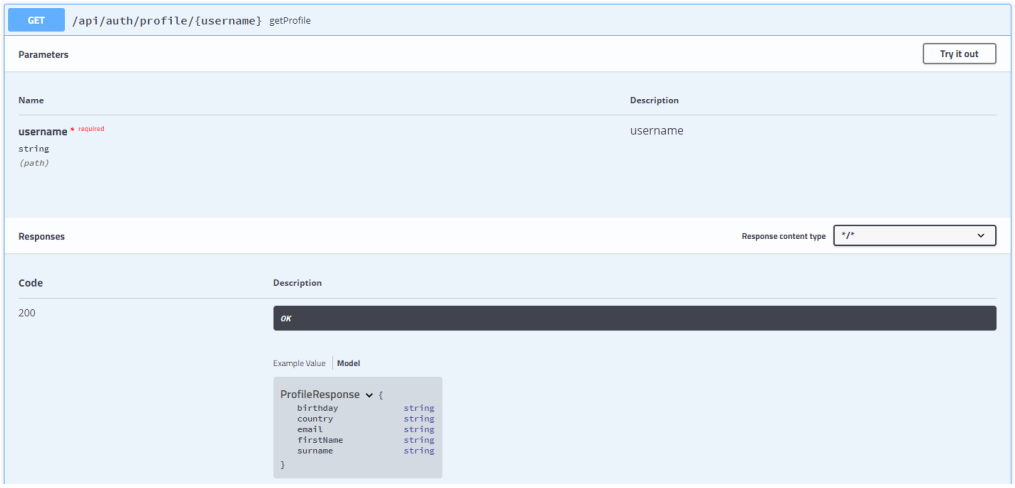

Figura 6.29: Esquema de la API getProfile

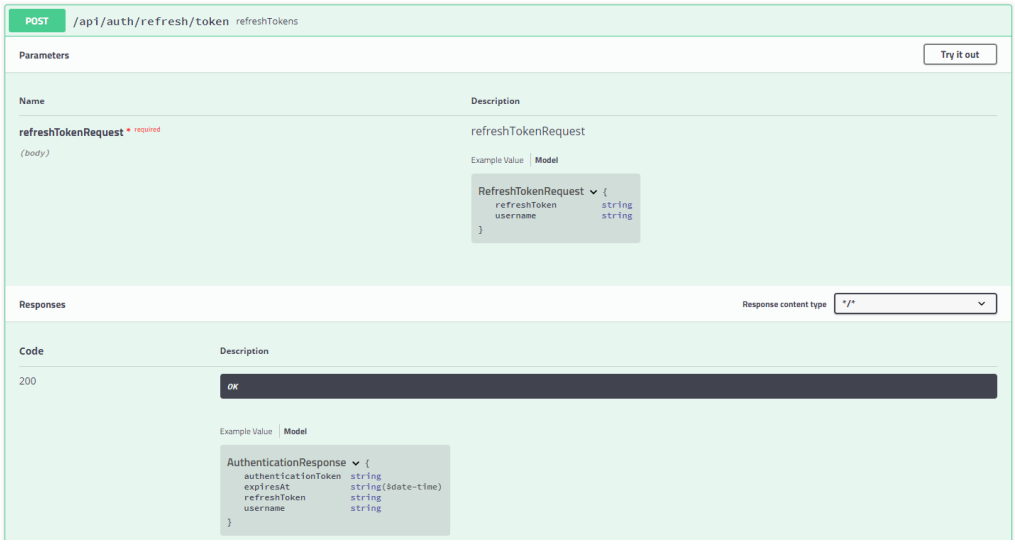

Figura 6.30: Esquema de la API refreshToken

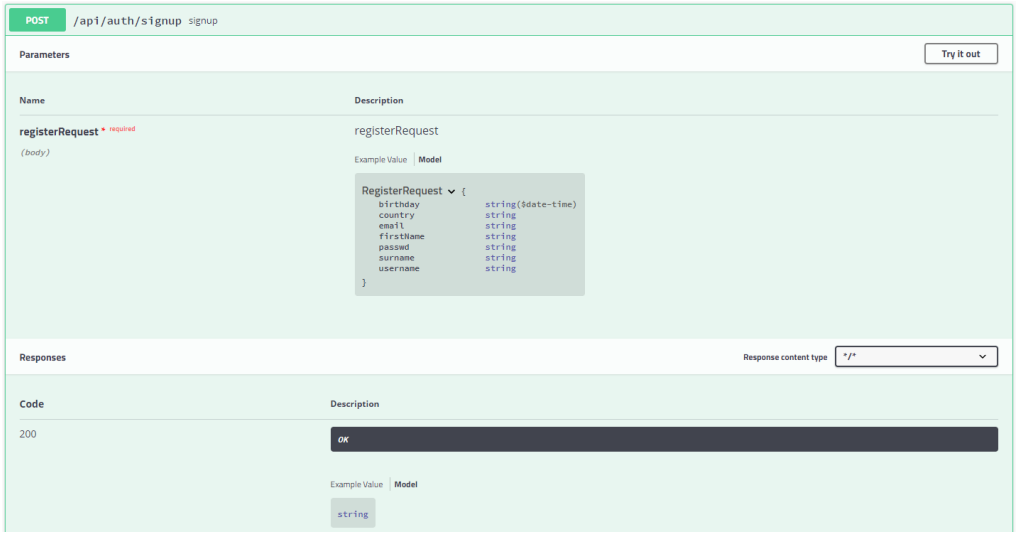

Figura 6.31: Esquema de la API signup

6.4.2. ChatController

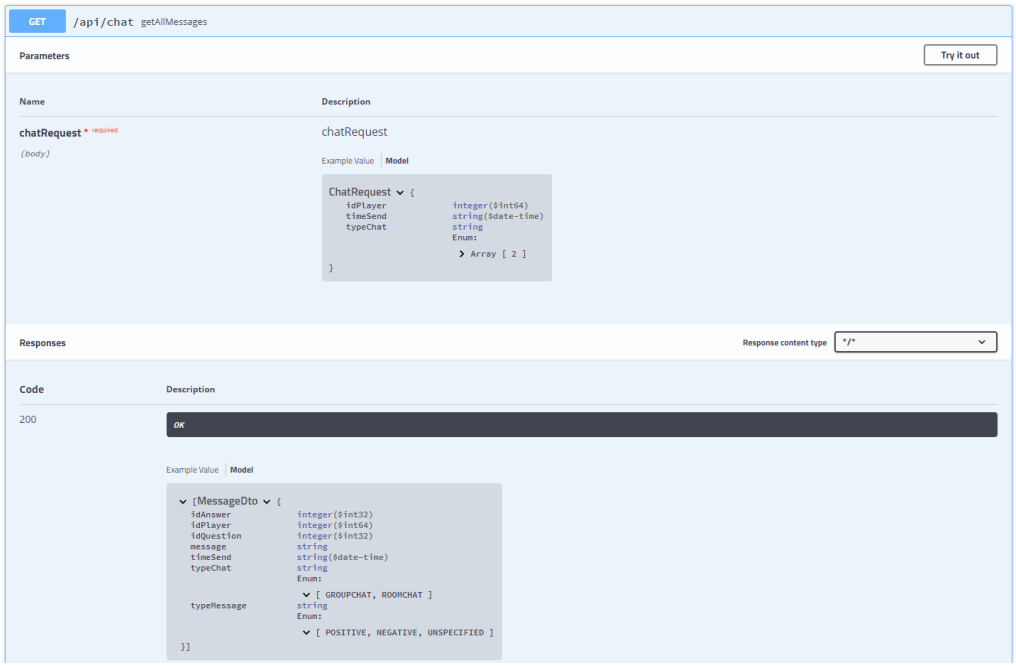

Figura 6.32: Esquema de la API getAllMessages

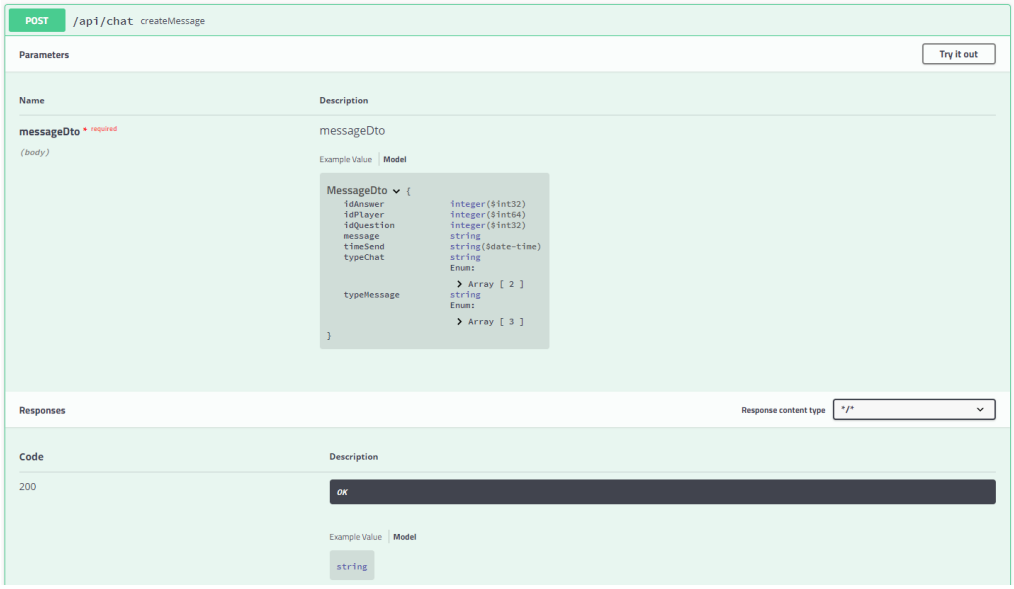

Figura 6.33: Esquema de la API createMessage

6.4.3. FormController

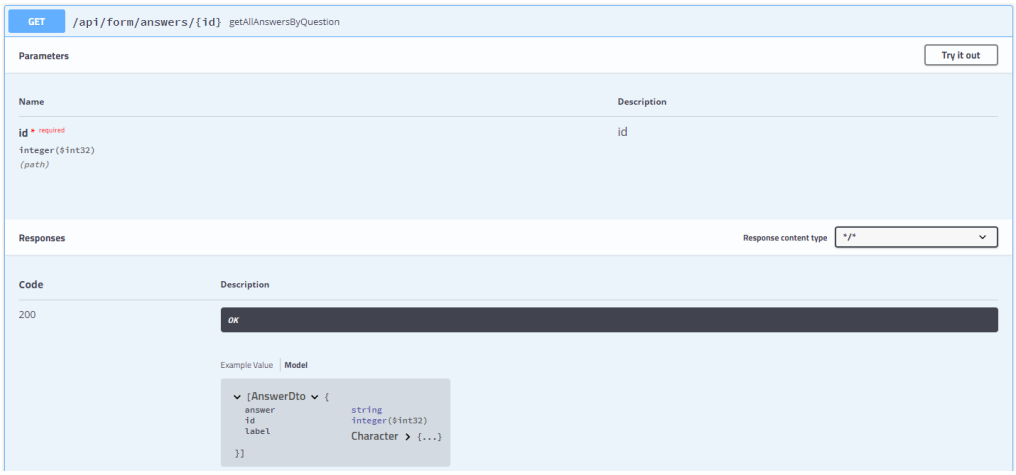

Figura 6.34: Esquema de la API getAllAnswersByQuestion

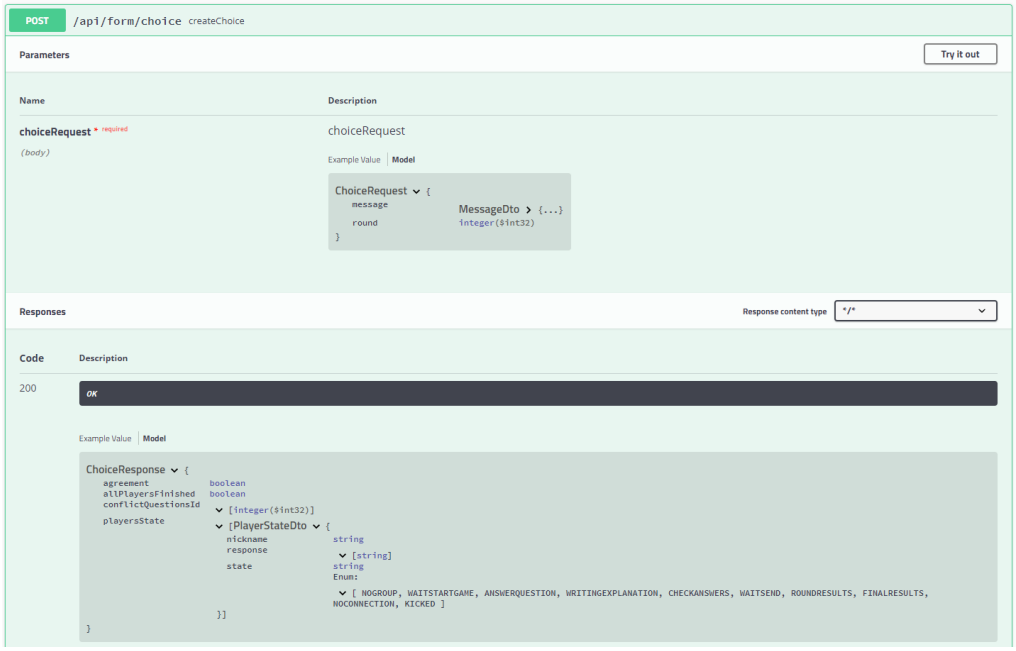

Figura 6.35: Esquema de la API createChoice

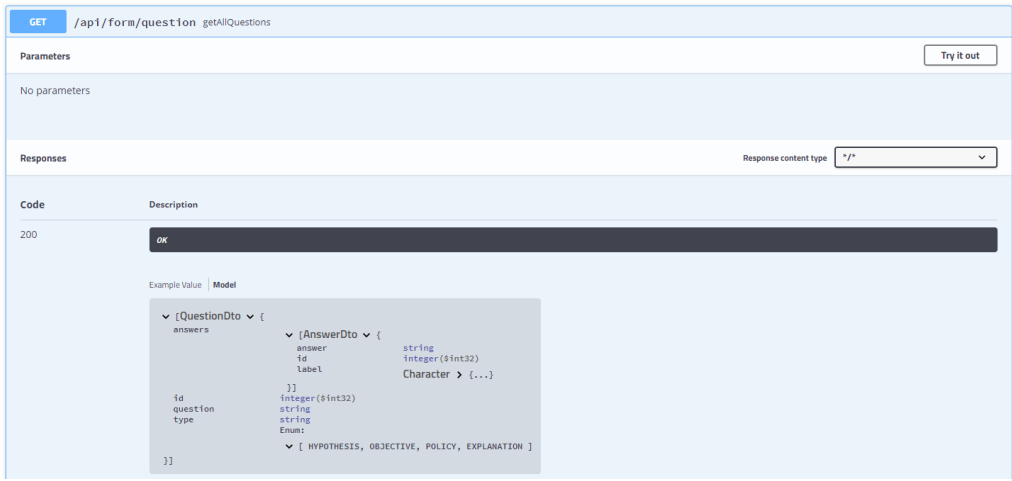

Figura 6.36: Esquema de la API getAllQuestions

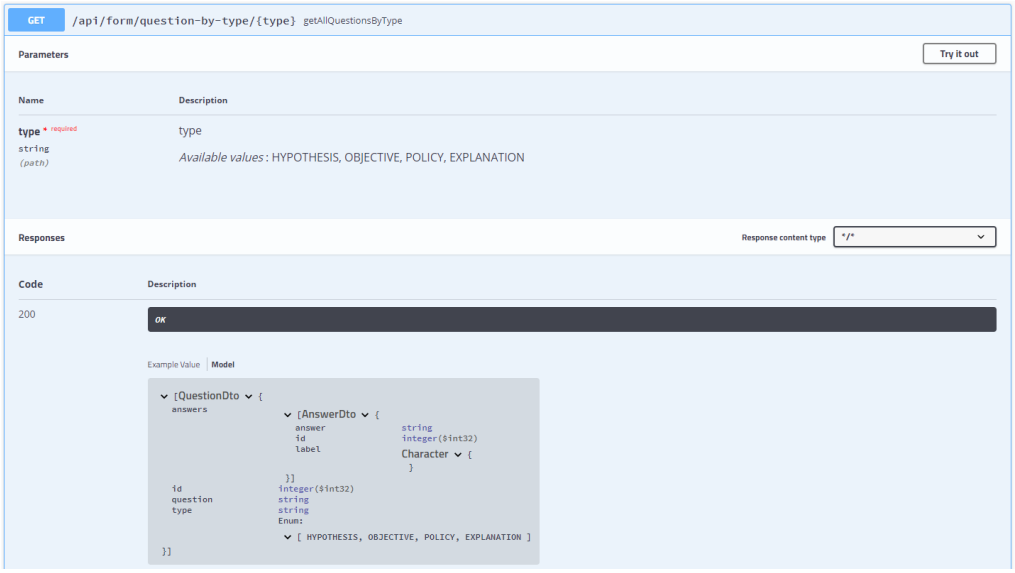

Figura 6.37: Esquema de la API getAllQuestionsByType

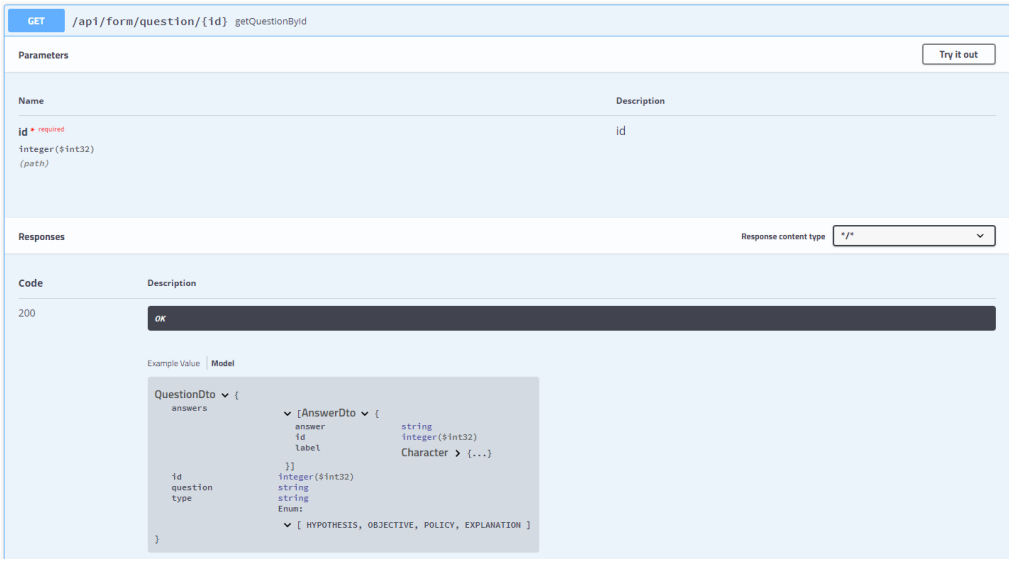

Figura 6.38: Esquema de la API getQuestionById

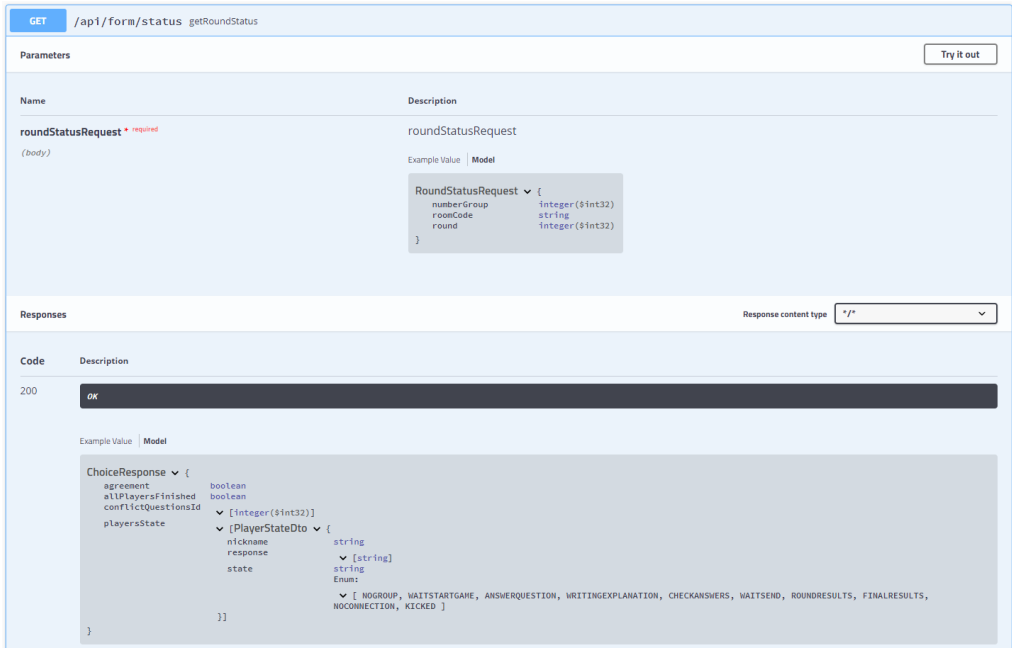

Figura 6.39: Esquema de la API getRoundStatus

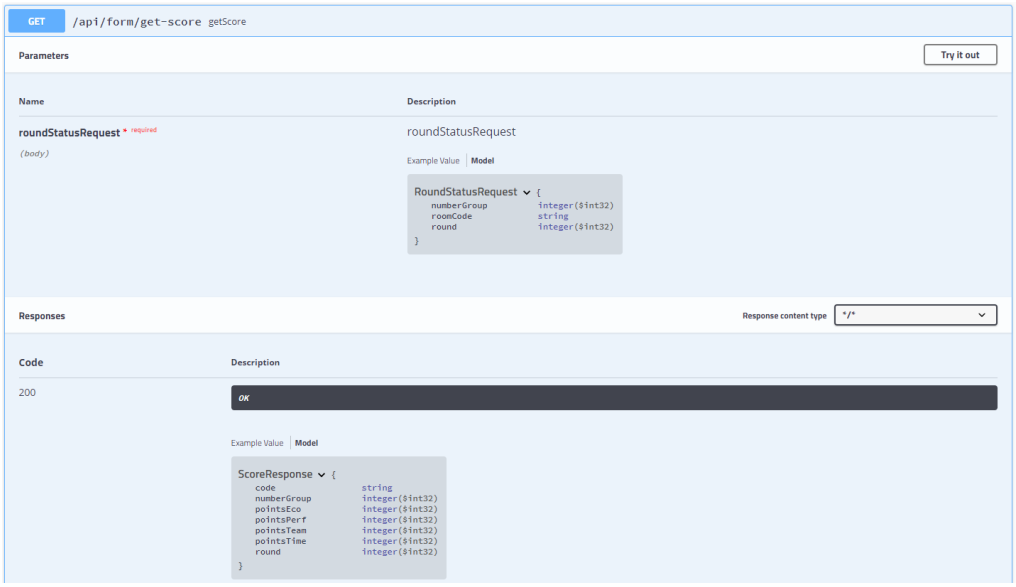

Figura 6.40: Esquema de la API getScore

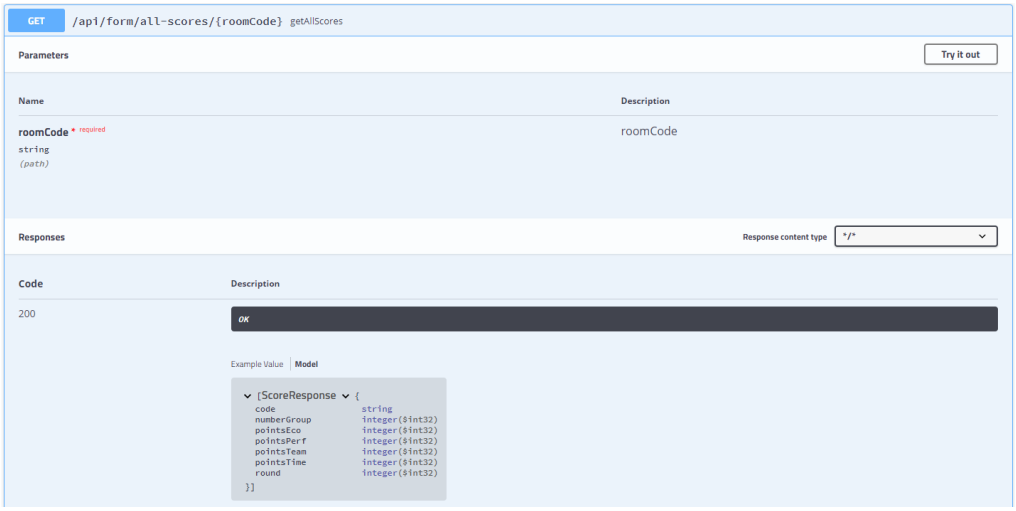

Figura 6.41: Esquema de la API getAllScores

6.4.4. GraphicController

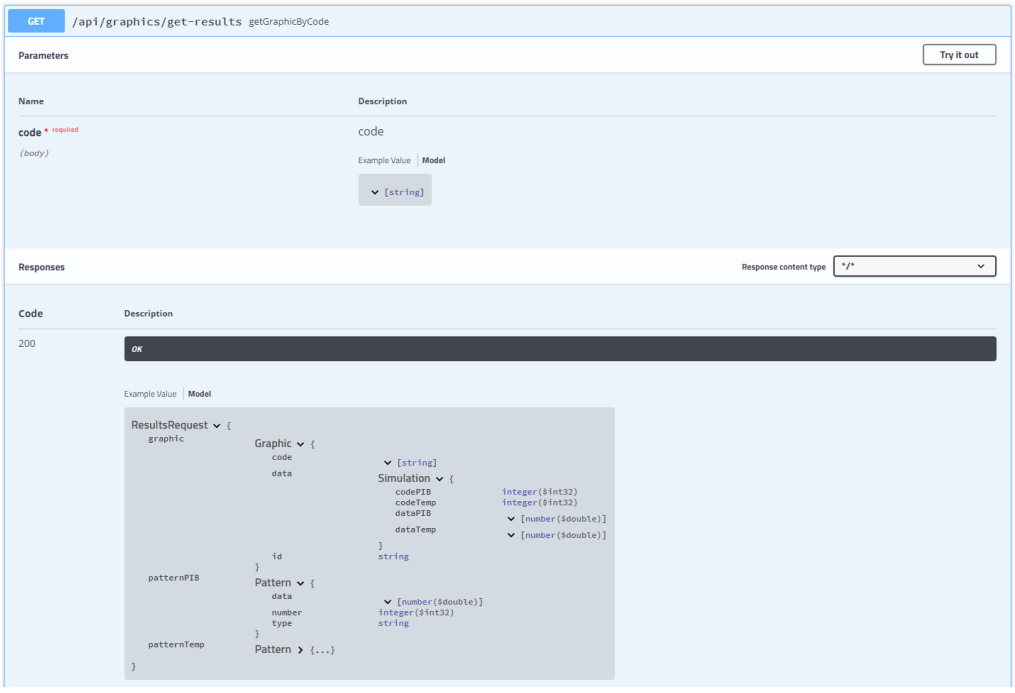

Figura 6.42: Esquema de la API getGraphicByCode

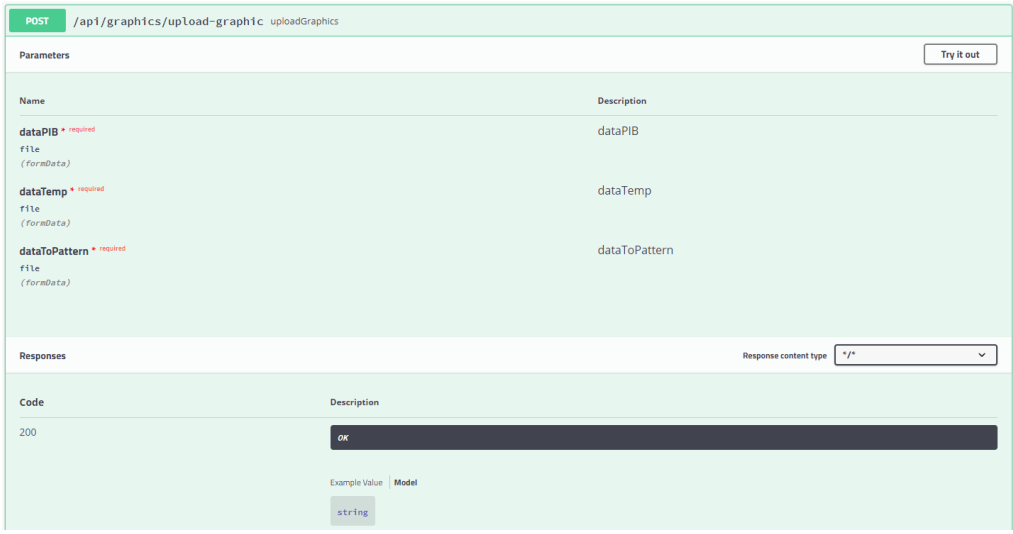

Figura 6.43: Esquema de la API uploadGraphics

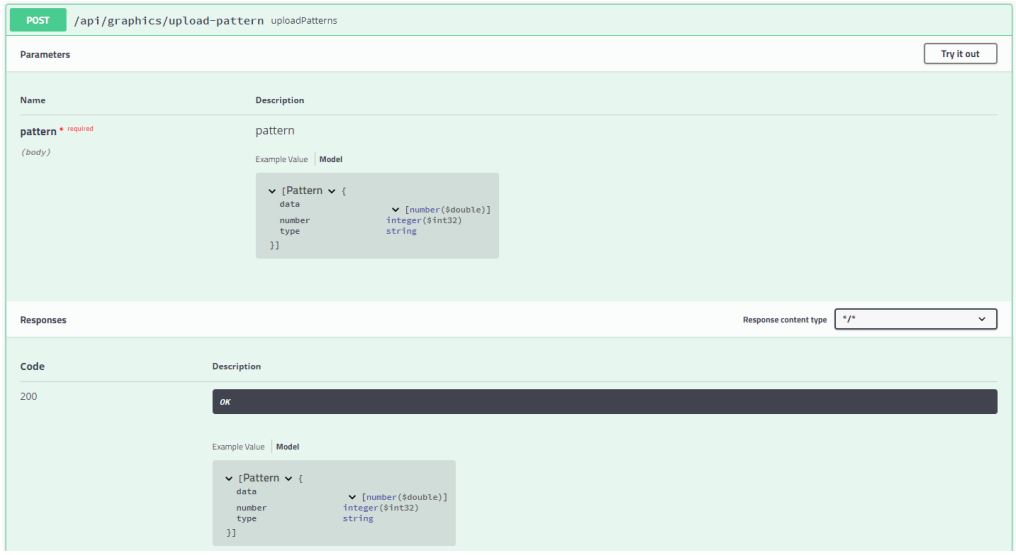

Figura 6.44: Esquema de la API uploadPatterns

6.4.5. PlayerController

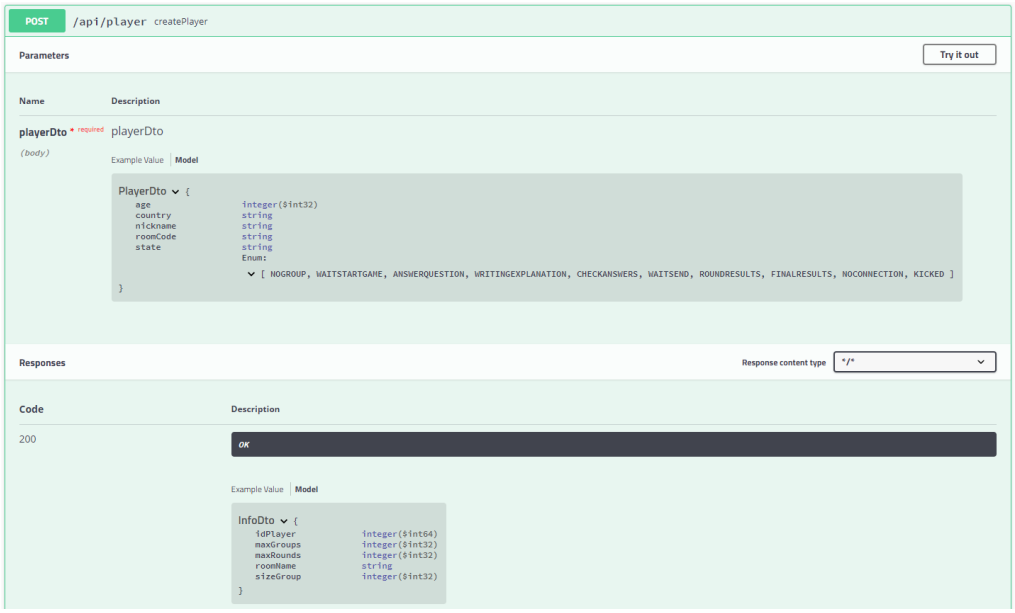

Figura 6.45: Esquema de la API createPlayer

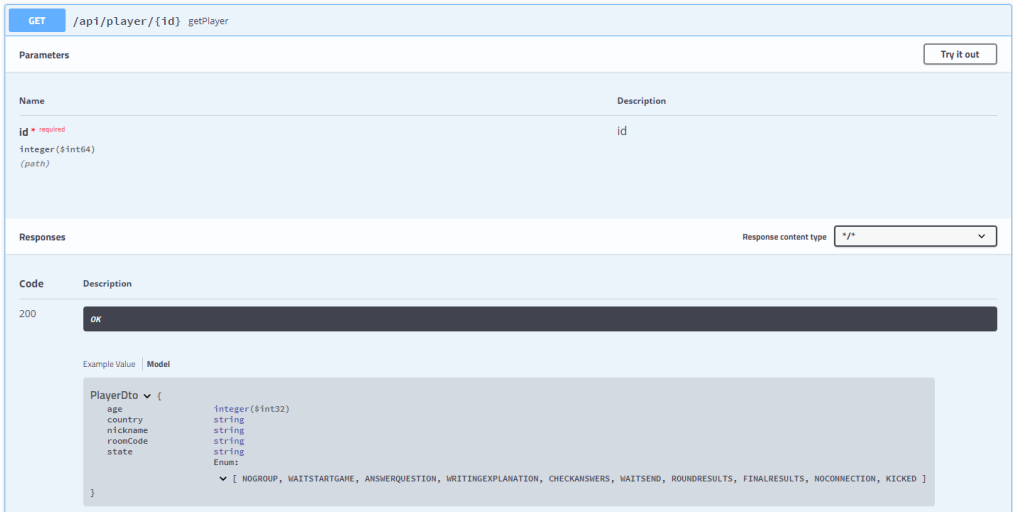

Figura 6.46: Esquema de la API getPlayer

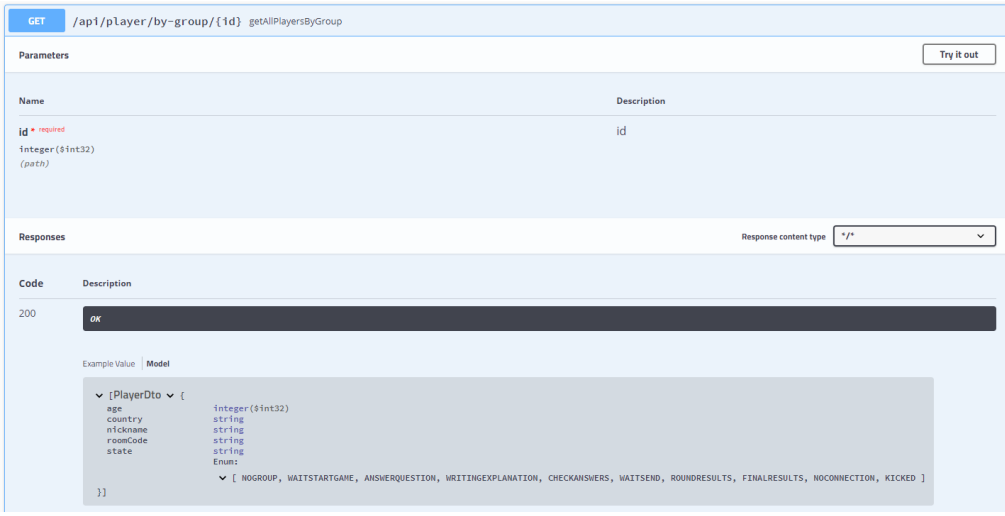

Figura 6.47: Esquema de la API getAllPlayersByGroup

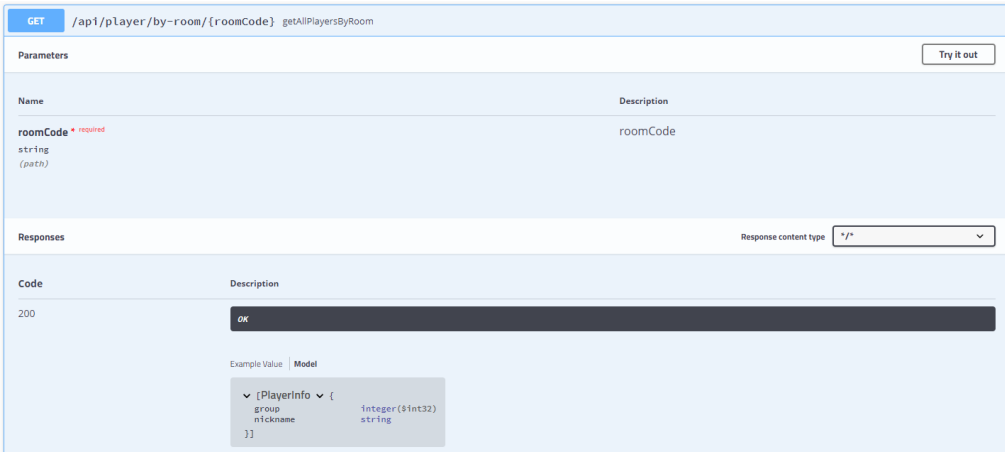

Figura 6.48: Esquema de la API getAllPlayersByRoom

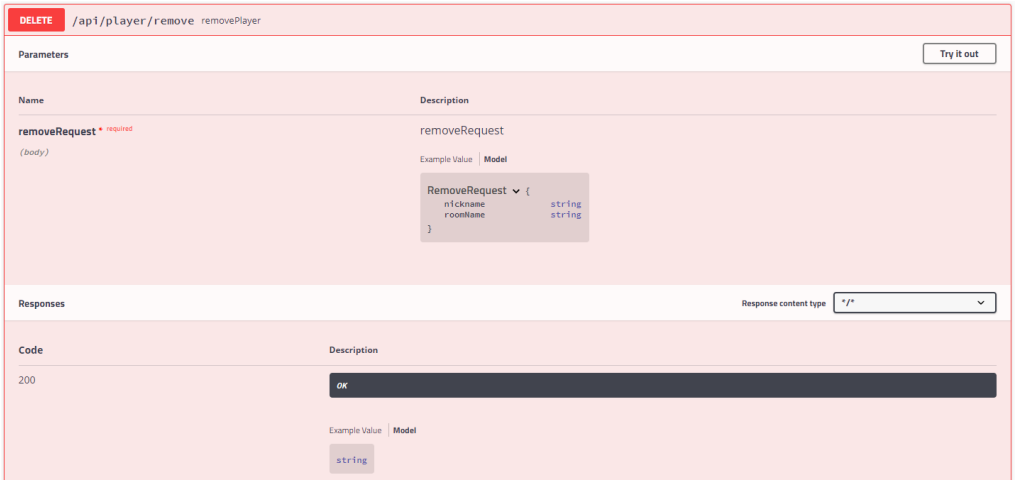

Figura 6.49: Esquema de la API removePlayer

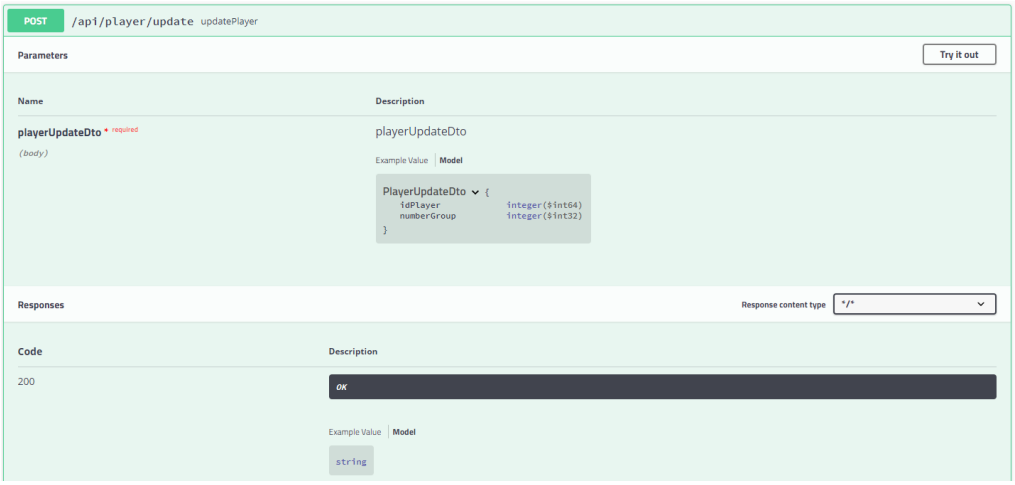

Figura 6.50: Esquema de la API updatePlayer

6.4.6. RoomControler

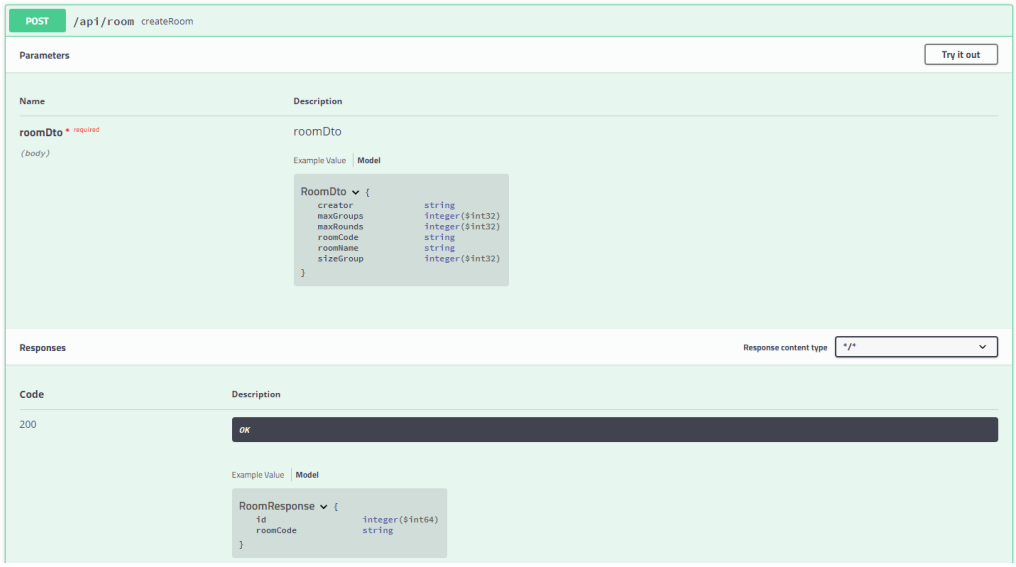

Figura 6.51: Esquema de la API createRoom

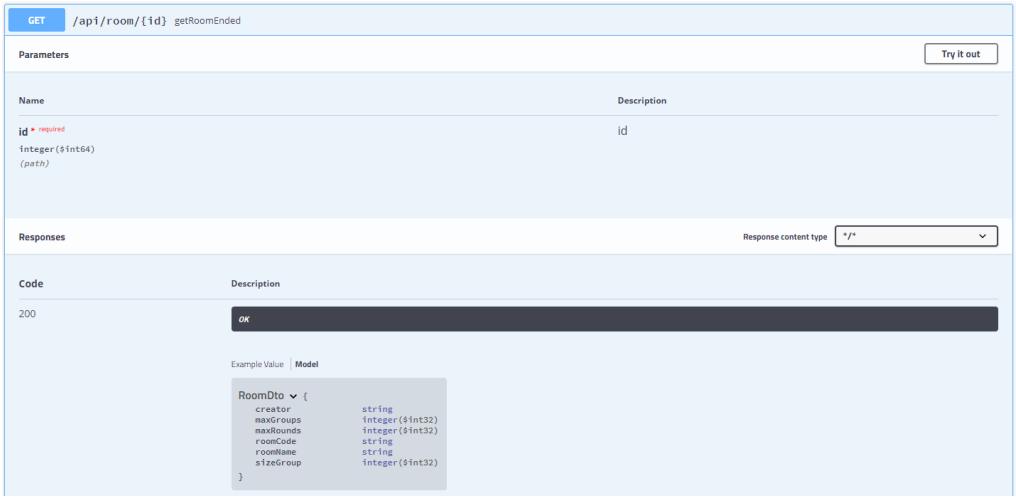

Figura 6.52: Esquema de la API getRoomEnded

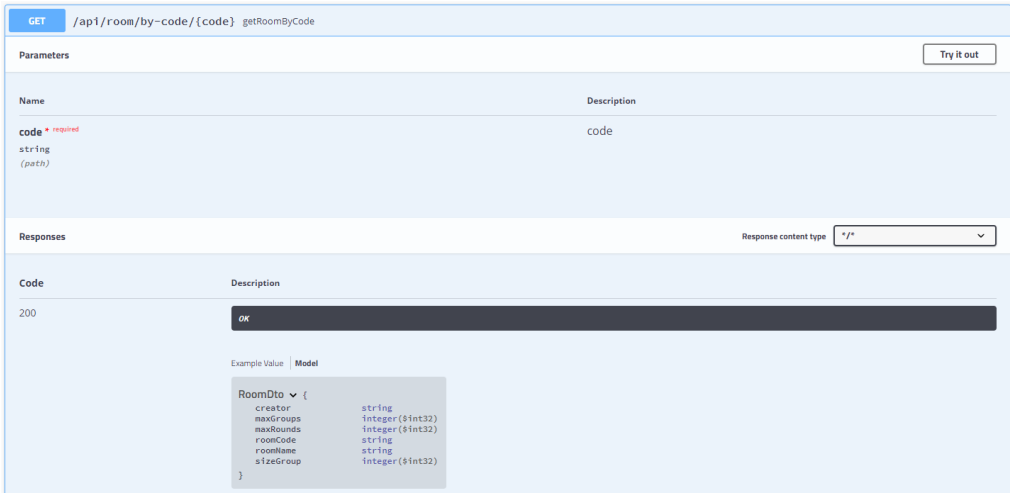

Figura 6.53: Esquema de la API getRoomByCode

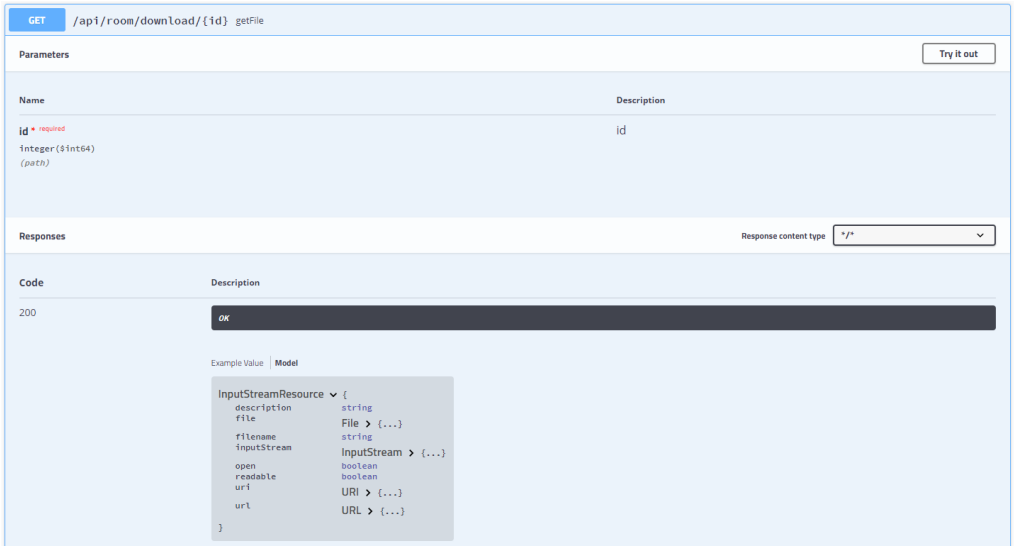

Figura 6.54: Esquema de la API getFile

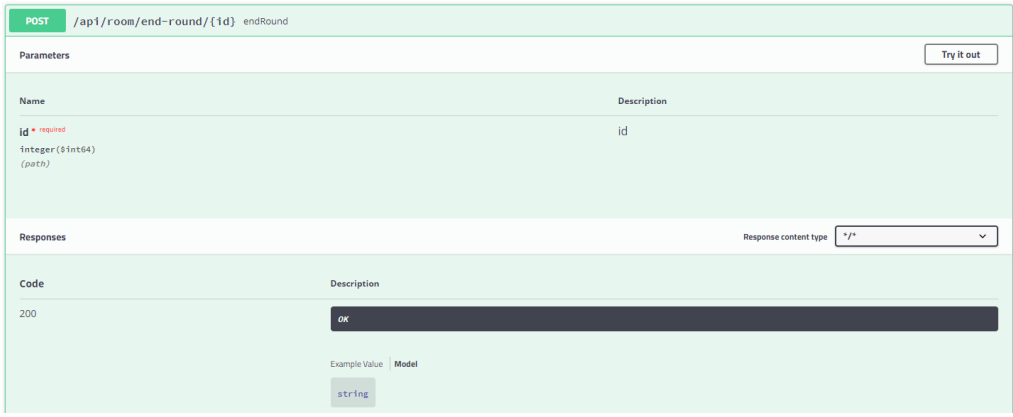

Figura 6.55: Esquema de la API endRound

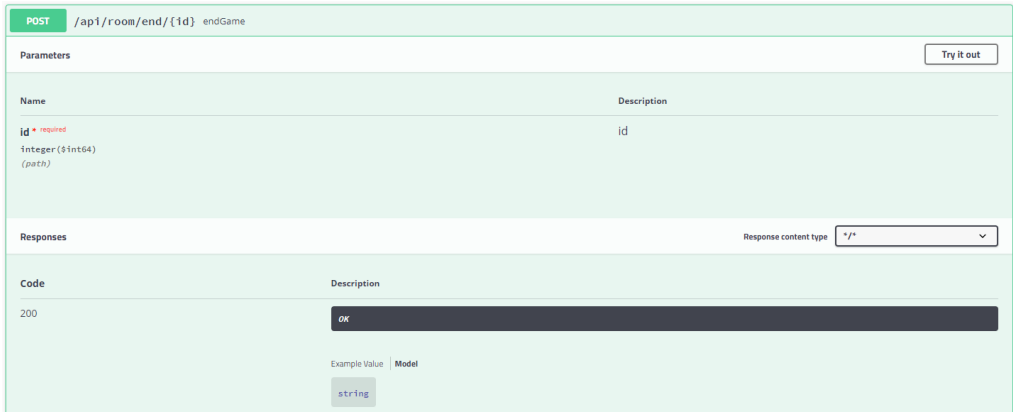

Figura 6.56: Esquema de la API endGame

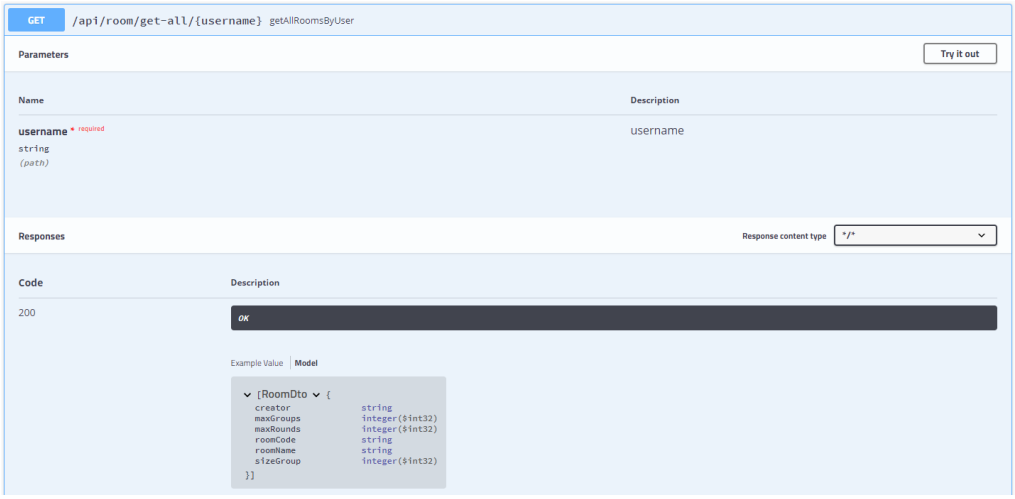

Figura 6.57: Esquema de la API getAllRoomsByUser

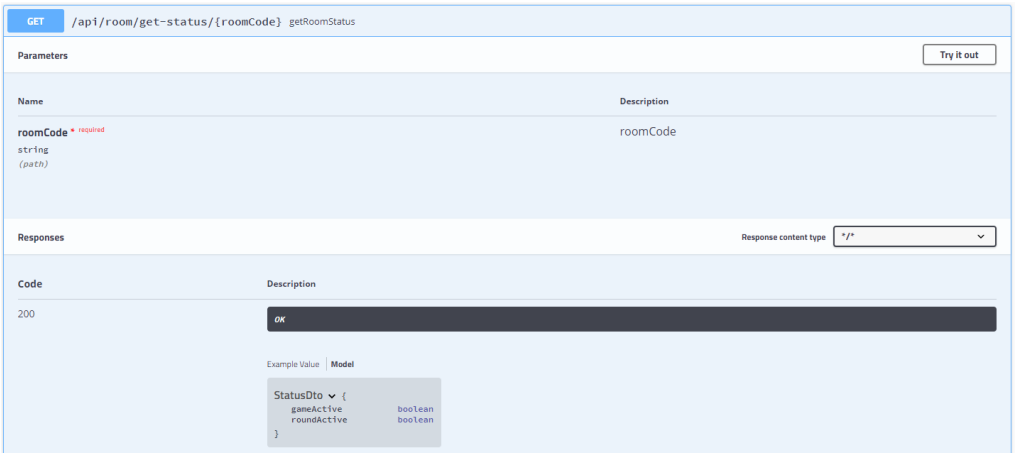

Figura 6.58: Esquema de la API getRoomStatus

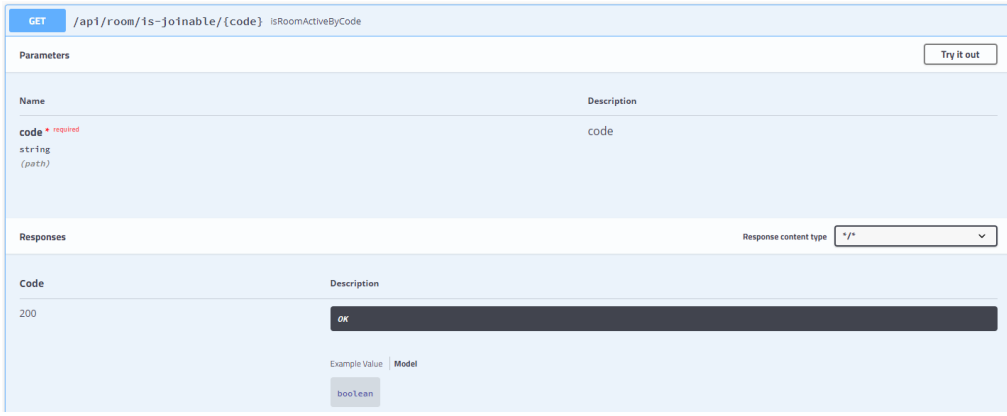

Figura 6.59: Esquema de la API isRoomActiveByCode

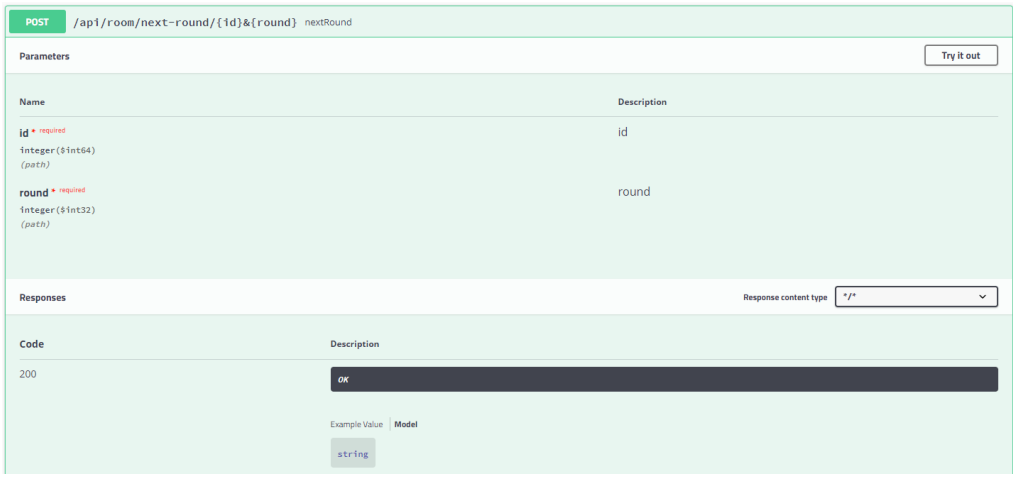

Figura 6.60: Esquema de la API nextRound

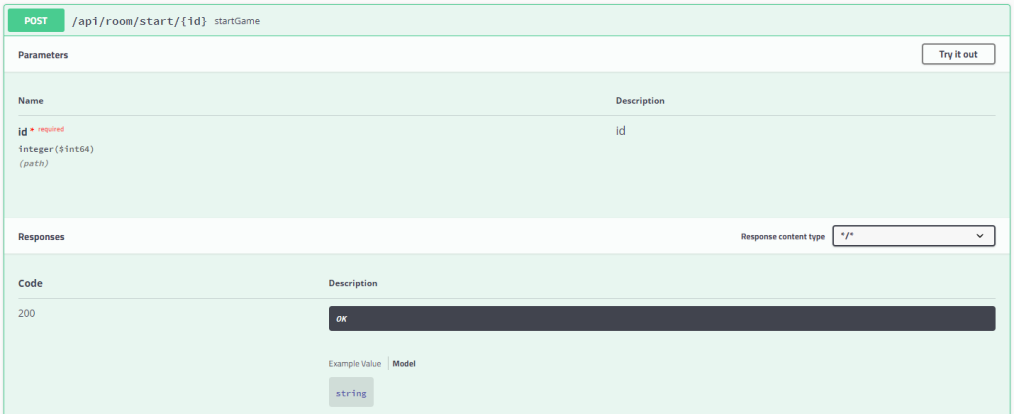

Figura 6.61: Esquema de la API startGame

6.5. Patrones

En esta sección se definirán patrones de diseño software que han sido implantados en la aplicación. De detallarán los patrones y dónde han sido aplicados.

6.5.1. Patrón Singleton

Patrón diseñado para asegurar una única instanciación de una clase, teniendo un único punto de acceso a la misma. La clase dispone de una función que crea la instancia de la clase y cada vez que es llamada devuelve esta instancia, evitando la creación de otras diferentes. En nuestra aplicación cada bean de Spring se encarga de crear una instancia de una clase concreta, de forma que cuando se necesite un objeto de esa clase, Spring inyecta dicho bean en la variable sin realizar una nueva instanciacion.

6.5.2. Patrón IoC

El patrón de inversión de control o IoC es también conocido como principio de Hollywood por la frase “No nos llame, nosotros le llamaremos”. Este patrón hace referencia a los casos en los que un componente del programa/sistema no dirige el flujo de la ejecución sino que el control del flujo queda relegado a un agente externo al componente que decide cómo y cuándo lo llama. En este proyecto, se aplica en la inyección de dependencias necesarias para el funcionamiento de la aplicación.

6.5.3. Patrón Inyección de Dependencias

Aplicación del patrón IoC. Se trata de suministrar a una clase de los objetos que necesite sin la necesidad de instanciarlos. En el proyecto, se aplica al inyectar en las variables que lo necesiten los beans creados por el core containter de Spring.

6.5.4. Patrón de mapeo de datos

El patrón de mapeo de datos se utiliza para la transferencia de información entre la base de datos y la representación de datos dentro de la aplicación. El objetivo es mantener ambas partes independientes la una de la otra.

6.6. Diagramas de clases de diseño

En esta sección se ilustrarán las diferentes capas del backend del proyecto.

6.6.1. Controladores

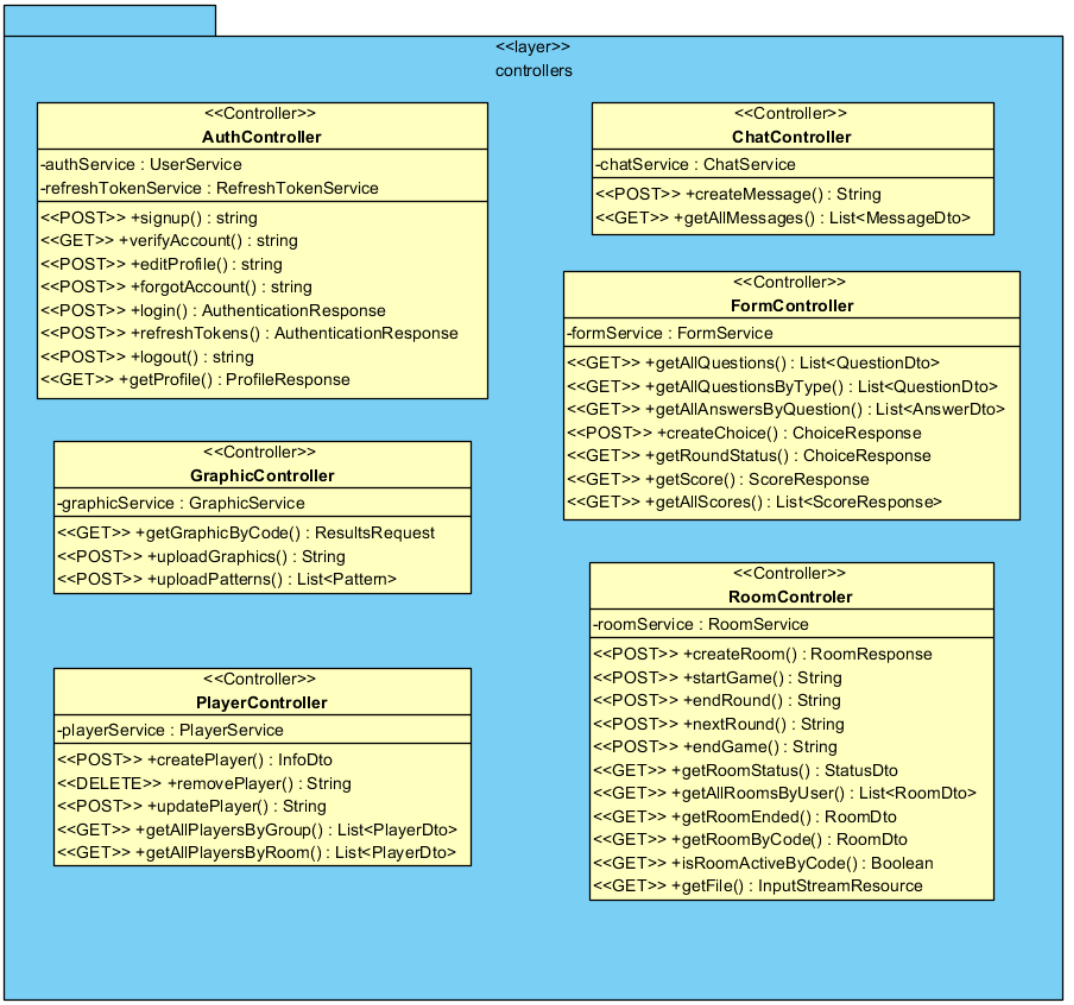

Figura 6.62: Diagrama de controladores en diseño

6.6.2. Recursos

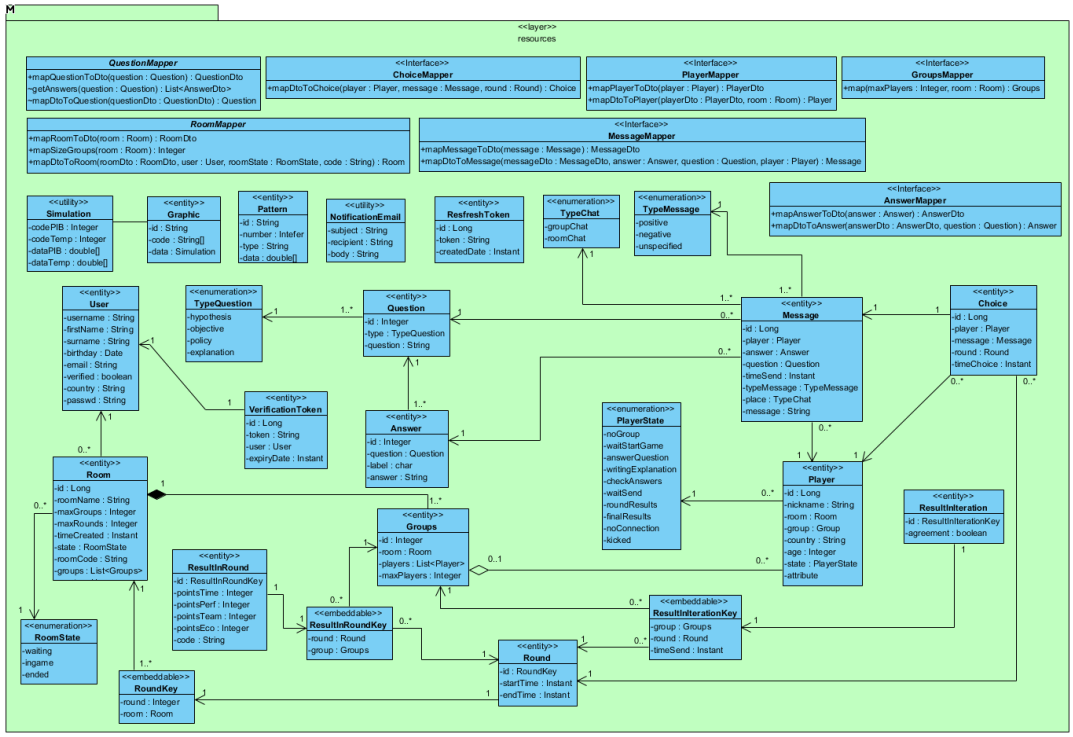

Figura 6.63: Diagrama de recursos en diseño 1

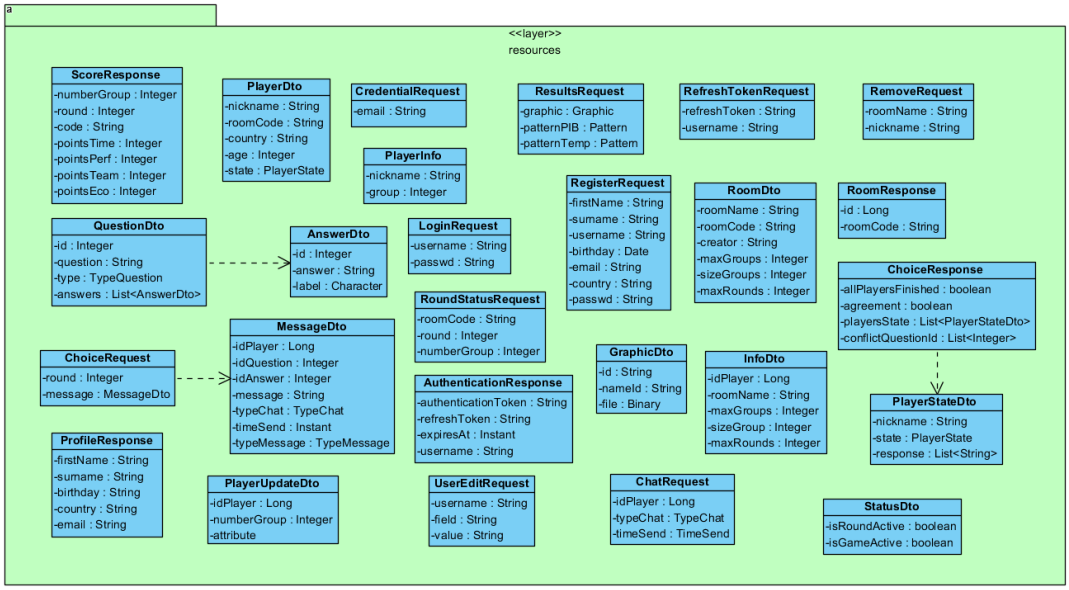

Figura 6.64: Diagrama de recursos en diseño 2

6.6.3. Servicios

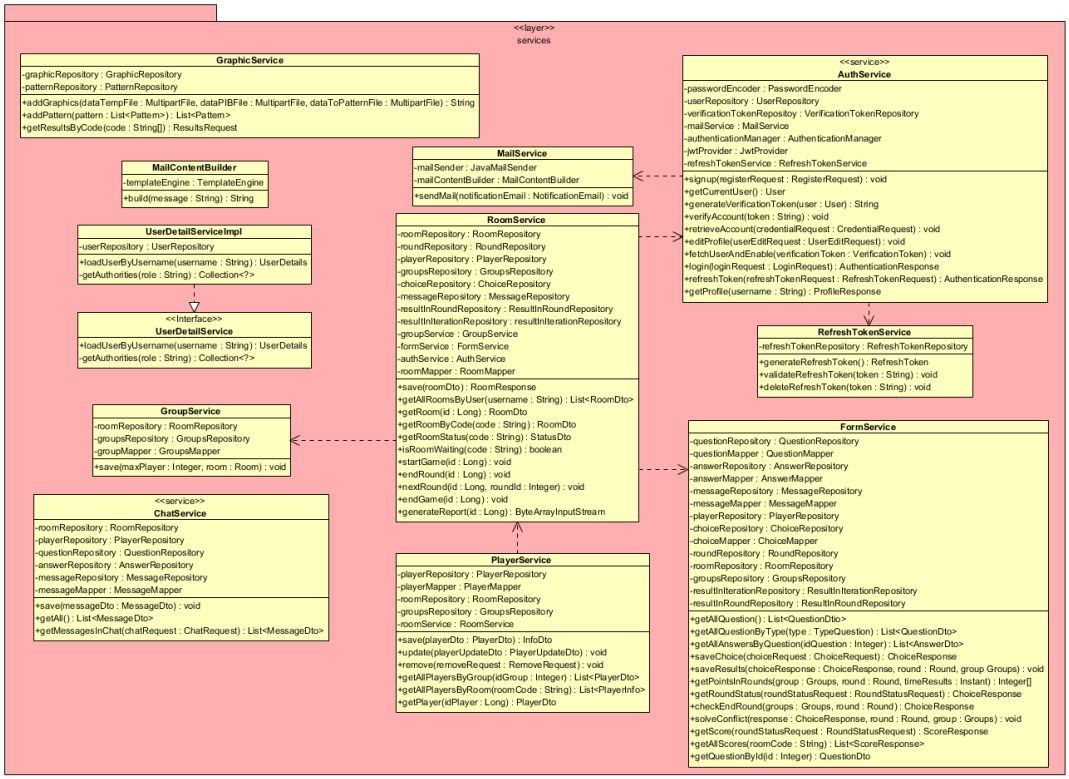

Figura 6.65: Diagrama de servicios en diseño

6.6.4. Persistencia

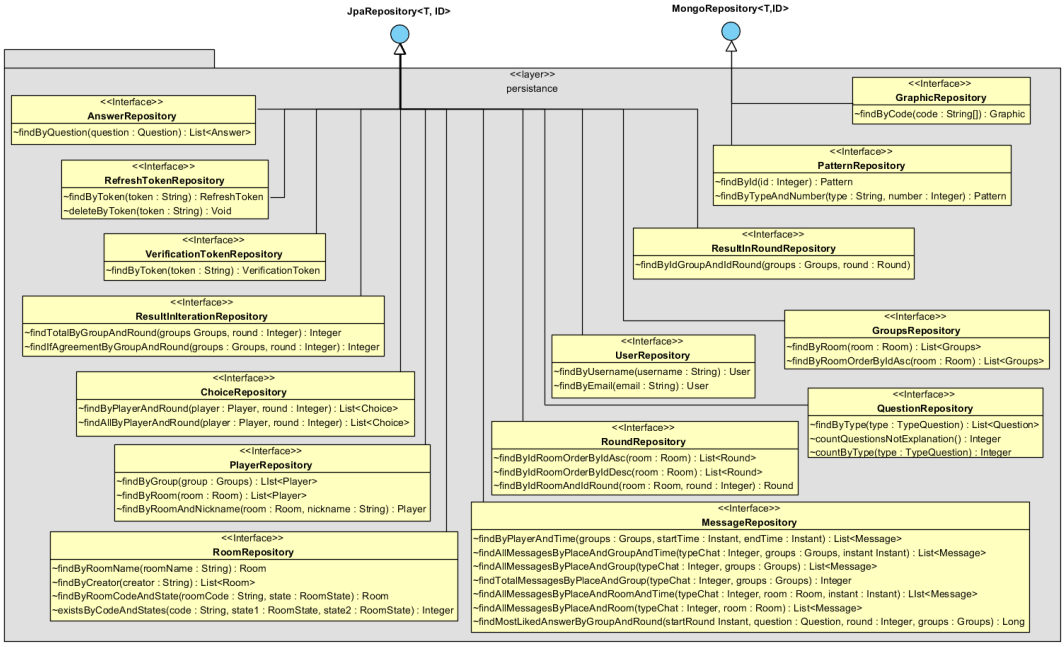

Figura 6.66: Diagrama persistencia en diseño

6.6.5. Modelo dinámico de casos de uso

En esta sección se presentarán algunos de los procesos de de la aplicación. Sólo se detallarán dos casos de uso debido a que el resto siguen la misma dinámica. Se han escogido dos de los procesos más importantes del proyecto: el fin de ronda, puesto que gestiona la resolución forzada de conflictos de cada grupo, y la obtención del documento pdf con los datos de la partida.

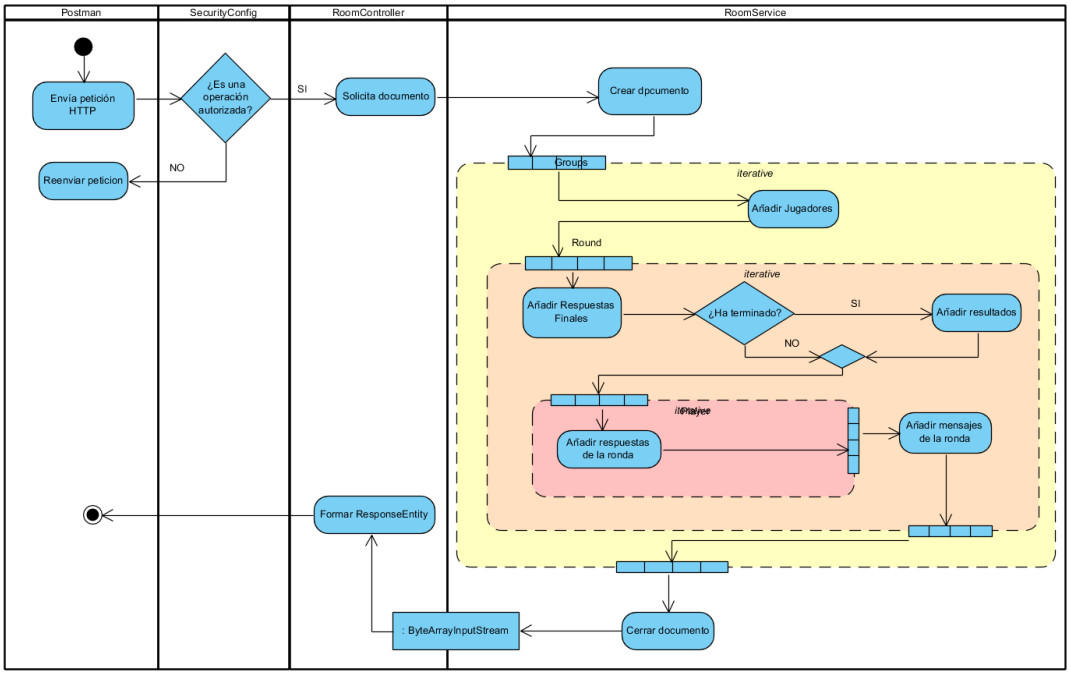

Figura 6.67: Diagrama de actividad con carriles de obtención del documento pdf

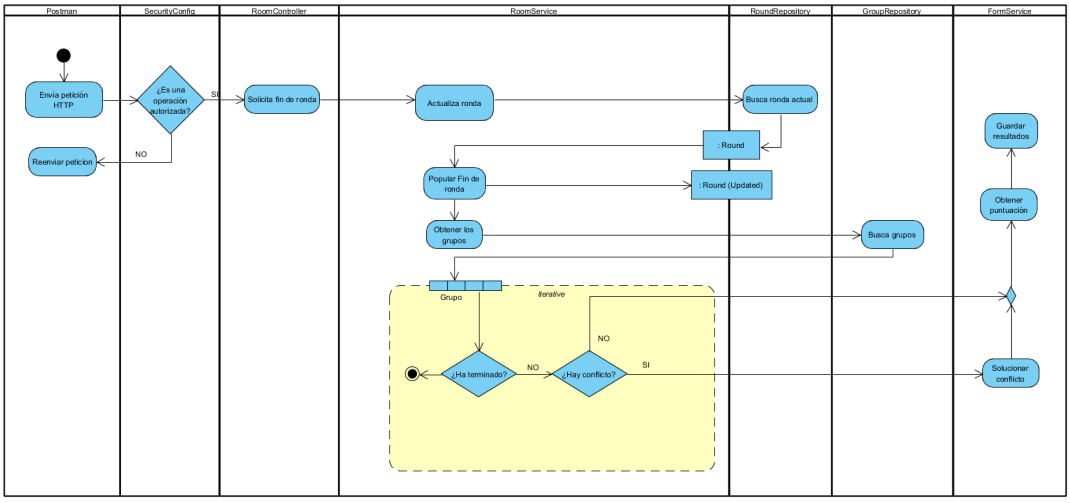

Figura 6.68: Diagrama de actividad con carriles de fin de ronda

Capítulo 7

Implementación y pruebas

7.1. Descripción de licencia y repositorio

Para este proyecto se ha decidido utilizar la licencia de GNU V3 [27], y estará alojado en un repositorio GitLab de la escuela.

7.2. Realización de las simulaciones

Las simulaciones se ejecutaron en Vensim. Este software permite abrir como modelo un script que acepta instrucciones para ejecutar en el programa [29]. Como se puede ver en la figura Porción del código de simulación de Vensim, ejecutamos opciones del menú del programa para abrir el modelo MEDEAS, preparar las entradas que recibirá y los resultados que se desean obtener. Una vez definida la simulación ejecutamos el análisis de sensibilidad. Este va realizando las 9720 simulaciones que contiene el fichero de entrada de datos y deposita los resultados en un fichero .vdf en la misma carpeta. Este script realiza este proceso 15 veces. Tras varias pruebas con el ordenador del laboratorio, se ha comprobado que la máquina es capaz de ejecutar concurrentemente 3 instancias de Vensim realizando análisis de sensibilidad sin aumentar el tiempo. Cada instancia ejecuta 15 sets de 9720 simulaciones. Cada set de simulaciones queda almacenado en un fichero identificado por el codigo de las preguntas fijas que se comparten entre las simulaciones del set, en este caso, las respuestas a las preguntas de las hipótesis del formulario. El tiempo de ejecución por set es de aproximadamente 22 horas, habiendo llevado un total de 14-15 días continuos de ejecución para obtener todos los resultados. Este proceso se ha tenido que realizar 2 veces debido a un problema en el guardado de los datos, puesto que en la primera iteración el fichero que estipula los datos a obtener de las simulaciones no estaba completo.

```
SPECIAL>LOADMODEL|MEDEAS-W_v165_ecopV14.mdl
SIMULATE>RUNNAME| H1bH2aH3b.vdf
SIMULATE>SENSITIVITY|  H1bH2aH3b.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0

SIMULATE>RUNNAME| H1bH2aH3c.vdf
SIMULATE>SENSITIVITY|  H1bH2aH3c.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0


SIMULATE>RUNNAME| H1bH2aH3d.vdf
SIMULATE>SENSITIVITY|  H1bH2aH3d.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0

SIMULATE>RUNNAME| H1bH2aH3e.vdf
SIMULATE>SENSITIVITY|  H1bH2aH3e.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0


SIMULATE>RUNNAME| H1bH2bH3a.vdf
SIMULATE>SENSITIVITY|  H1bH2bH3a.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0

SIMULATE>RUNNAME| H1bH2bH3b.vdf
SIMULATE>SENSITIVITY|  H1bH2bH3b.vsc
SIMULATE>SENSSAVELIST| VariablesSalida_v3.lst
MENU>RUN_SENSITIVITY|0
```

Figura 7.1: Porción del código de simulación de Vensim.

Tras realizar las pruebas, había que transformar los datos a un formato legible. Vensim ofrece la posibilidad de exportar un fichero .vdf a un documento de texto por tabulaciones. Después de intentar trabajar con los comandos del programa sin obtener un resultado válido, se optó por utilizar un script de parseo automático con Python [32]. Para ello, fue necesario utilizar dos librerías de control de pantalla: pywinauto [20] y pyautogui [28]. Estas librerías permiten abrir aplicaciones, introducir entradas de teclado y ratón al ordenador, simulando la ejecución de un usuario normal en la máquina. El script simula el recorrido del usuario, abriendo el programa, seleccionando la opción de transformación de datos, ficheros necesarios y donde se quiere guardar. Para realizar este script fue necesaria una gran cantidad de pruebas con la inserción de teclas y macros para el control de las ventanas. El uso del ratón era más volátil, puesto que se dependía de la resolución de la pantalla, el tamaño de las ventanas y la posición de las mismas, por lo que se optó por controlar la acción a través de cambios de elementos por medio de tabulaciones.

En la figura Porción del código de exportación de los resultados del script Python se puede ver como se abre el programa y se inserta un script .cmd ilustrado en la figura Script .cdm ejecutado durante la exportación de los resultados, que realiza múltiples veces la llamada de exportación de datos. Se puede apreciar como tras cada acción se para el programa un determinado tiempo. Esto se debe a la necesidad de evitar que unos comandos pisen a los siguientes. Al realizar acciones sobre las ventanas del ordenador, este puede sufrir un retardo por la sobrecarga de la que disponga en el momento. En la figura Porción del código de exportación de los resultados del script Python se muestra como se elige el nombre y las opciones del fichero que se obtendrá, teniendo que pasar de opción en opción a través de tabulaciones.

```
SPECIAL>LOADMODEL|MEDEAS-W_v165_ecopV14.mdl
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>VDF2TAB
MENU>EXIT
```

Figura 7.2: Script .cdm ejecutado durante la exportación de los resultados.

```python
from pywinauto import application
from pywinauto.keyboard import send_keys
import pyautogui
import time

app = application.Application()
app.start(r"C:\Program Files (x86)\Vensim\vensim.exe")
time.sleep(8)
#app.send_keys('^o')
#app.Vensim.MenuSelect("File ->OpenModel")
#app.Vensim.MenuBar.MenuBarClickInput("#0->#1")

#Abriendo el script
pyautogui.hotkey("ctrl", "o")
time.sleep(1)
pyautogui.press(["tab", "down", "down", "down", "down", "down", "down", "enter"])
time.sleep(1)
pyautogui.hotkey("shift", "tab")
time.sleep(1)
pyautogui.hotkey("shift", "tab")
time.sleep(1)
pyautogui.hotkey("shift", "tab")
time.sleep(1)
pyautogui.press(["up", "down", "enter"])
time.sleep(15)

#Seleccionando el Dataset
##H1bH2aH3a
pyautogui.hotkey("shift", "tab")
time.sleep(1)
pyautogui.hotkey("shift", "tab")
time.sleep(1)
pyautogui.press(["down", "up", "enter"])
time.sleep(3)
pyautogui.press("enter")

```

Figura 7.3: Porción del código de exportación de los resultados del script Python.

```
292  ##H1bH2cH3a
293  pyautogui.hotkey("shift", "tab")
294  time.sleep(1)
295  pyautogui.hotkey("shift", "tab")
296  time.sleep(1)
297  pyautogui.press(["down", "down", "down", "down", "down", "down", "down", "down", "down", "down", "enter"])
298  time.sleep(3)
299  pyautogui.press("enter")
300
301  #Borrando el nombre anterior
302  pyautogui.press(["del", "del", "del", "del", "del", "del", "del", "del", "del", "del", "del", "del", "del"])
303
304  #Poniendo nombre al fichero y eligiendo las opciones
305  pyautogui.hotkey("shift", "h")
306  pyautogui.press(["1", "b"])
307  pyautogui.hotkey("shift", "h")
308  pyautogui.press(["2", "c"])
309  pyautogui.hotkey("shift", "h")
310  pyautogui.press(["3", "a", ".", "t", "a", "b"])
311  time.sleep(3)
312  pyautogui.press(["tab", "tab", "tab", "tab", "tab", "left", "tab", "left", "tab"])
313  time.sleep(10)
314  pyautogui.press("enter")
315  time.sleep(120)
316
317  ##H1bH2cH3b
318  pyautogui.hotkey("shift", "tab")
319  time.sleep(1)
320  pyautogui.hotkey("shift", "tab")
321  time.sleep(1)
322  pyautogui.press(["down", "down", "down", "down", "down", "down", "down", "down", "down", "down", "down", "enter"])
323  time.sleep(3)
324  pyautogui.press("enter")
```

Figura 7.4: Porción del código de exportación de los resultados del script Python.

7.3. Organización del código

El proyecto se encuentra distribuido en diferentes directorios, donde se alojan los ficheros de la aplicación. Dentro de la carpeta *src/main/java/com/crossroads/spring* se encuentra el código del backend, repartido en los siguientes paquetes:

- *config*: Contiene las clases de configuración de la aplicación. Asegura que los usuarios dispongan de los permisos necesarios para realizar las peticiones.
- *controller*: Contiene los controladores de las llamadas API REST del proyecto.
- *dto*: Contiene todas las clases de transferencia de datos. Son utilizados para la comunicación entre cliente y servidor.
- *exceptions*: Contiene las excepciones definidas del proyecto.
- *mapper*: Contiene las clases que controlan el mapeo entre la información de las clases de dominio y los DTO (Data Transfer Object).
- *model*: Contiene la definición de todas las entidades del dominio.
- *persistance*: Contiene las interfaces de todos los repositorios necesarios para las operaciones a realizar contra las bases de datos.
- *security*: Contiene las clases de verificación de autenticación de los usuarios.
- *service*: Contiene los servicios necesarios para la lógica de la aplicación, estructurados según la funcionalidad que desempeñen.

Dentro de *src/main/resources* se guardan las plantillas necesarias del proyecto, en este caso, la plantilla del correo a enviar al usuario cuando crea una cuenta. Además, también se encuentra en este directorio el archivo *application.properties* que contiene la configuración de la aplicación y los detalles de las bases de datos, y el fichero *crossroads.jks* que contiene el set de claves para la serialización de las contraseñas de los usuarios, asegurando así su encriptación en la base de datos MySQL.

7.4. Cambios y modificaciones realizados

A lo largo del proceso de implementación, han surgido problemas o carencias en el diseño del modelo. Estos cambios se han realizado por necesidad o para simplificar el funcionamiento de la aplicación. Los cambios son los siguientes:

- Se ha cambiado el mockup de preguntas. Previamente se mostraban todas las preguntas dentro de la misma ventana. Para simplificar la interacción, se ha decidido mostrar cada pregunta en una ventana.

- Se han relacionado las respuestas con los mensajes y no con las elecciones. Este cambio ha sido realizado para hacer posible la argumentación de las preguntas y respuestas sin tener que hacer una nueva elección cada vez. Además, esto permite poder argumentar sobre una posible opción sin tener que estar en la página de la pregunta en cuestión.

- Se ha modificado el diseño de la base de datos MongoDB. Previamente se pretendía almacenar los ficheros con los datos de las simulaciones y acceder a éstos para la obtención de los resultados. Aprovechando las colecciones que proporciona esta base de datos no relacional, se ha optado por el guardado de los datos en las propias colecciones y no en los ficheros, simplificando el acceso a los datos y ahorrando el tiempo de lectura a que supondría trabajar con ficheros.

- Se ha cambiado la lógica de resolución de conflictos ante un fin prematuro de ronda. En propuestas iniciales, se planteó como opción para la resolución de conflictos causados por falta de respuestas por los integrantes, el uso de respuestas por defecto para devolver un resultado a los jugadores. En futuras versiones, se decidió cambiar, y no devolver datos a los jugadores a menos que al menos uno de los jugadores haya respondido a todas las preguntas.

- Se ha cambiado la interpretación de la *Trama narrativa*. Previamente se iba a considerar esta parte del cuestionario como un mensaje, pero no había forma de identificar fácilmente cuando era un mensaje del chat o el final del cuestionario. Se ha decidido crear una pregunta nueva con una única respuesta y tipo de pregunta para identificar este escenario. Se ha añadido a la enumeracion de *TypeQuestion*, ilustrada en el diagrama Diagrama de recursos en diseño 1 el tipo 'Explanation', para poder identificarlo.

7.5. Pruebas de integración

Para comprobar la correcta funcionalidad de la aplicación, se han realizado una batería de pruebas de caja negra de cada una de las operaciones API de la sección Estructura de las API REST, a través de la aplicación Postman. A continuación se expondrán las pruebas más relevantes ya sea por la complejidad de la logica de la API o porque han permitido detectar errores en la aplicación.

7.5.1. Casos de prueba

CP01	Registro de un usuario
Descripción	Registrar un usuario nuevo en el sistema
API REST	auth/signup
Entrada	▪ {'firstName': 'Lucas', 'surname': 'Gonzalez', 'username': 'lucgonz', 'birthday': '1998-01-01', 'email': 'lucgonz@email.com', 'country': 'Spain', 'passwd': 'lucgonz1'}
Resultado esperado	Creación del usuario en la base de datos y envío del correo de verificación

CP02	Registro de un usuario ya registrado
Descripción	Registrar un usuario ya guardado en el sistema
API REST	auth/signup
Entrada	▪ {'firstName': 'Lucas', 'surname': 'Gonzalez', 'username': 'lucgonz', 'birthday': '1998-01-01', 'email': 'lucgonz@email.com', 'country': 'Spain', 'passwd': 'lucgonz1'}
Resultado esperado	Mensaje de error indicando la existencia de un usuario

CP03	Inicio de sesión
Descripción	Iniciar sesión con un usuario registrado y verificado
API REST	auth/login
Entrada	▪ {'username': 'lucgonz1', 'passwd': 'lucgonz'}
Resultado esperado	Inicio de sesión. Se devuelven el token de refresco el token de autenticación y la fecha de caducidad del token.

CP04	Inicio de sesión con cuenta no verificada
Descripción	Iniciar sesión con un usuario registrado no verificado
API REST	auth/login
Entrada	▪ {'username': 'notverified', 'passwd': 'test'}
Resultado esperado	Mennsaje de error indicando que el usuario no ha verificado su cuenta.

CP05	Inicio de sesión con datos incorrectos
Descripción	Iniciar sesión con un usuario no registrado o con datos incorrectos
API REST	auth/login
Entrada	▪ {'username': 'notregistered', 'passwd': 'test'}
Resultado esperado	Mensaje de error indicando que los datos son erróneos.

CP06	Cierre de sesión
Descripción	Desconexión de un usuario de la aplicación
API REST	auth/logout
Entrada	▪ { 'refreshToken: '89af3459-1887-4d1b-922f-e2b9392d1524', 'username': 'test1' }
Resultado esperado	Elimina el token de refresco de la base de datos, impidiendo la generación de nuevos tokens de autenticación hasta que se vuelva a iniciar sesión.

CP07	Crear sala
Descripción	Crear una sala
API REST	room
Entrada	▪ { 'roomName': 'Full Game', 'maxGroups': 3, 'sizeGroup': 4, 'maxRounds': 4, 'roomCode': '', 'creator': 'test1' }
Resultado esperado	Creación de una sala de juego y generación de un código de ingreso..

CP08	Comprobar sala activa
Descripción	Comprobar si existe alguna sala activa con el código enviado
API REST	room/is-joinable/{roomCode}
Entrada	-
Resultado esperado	True si la sala existe una sala activa, false en caso contrario.

CP09	Generar documento en una partida terminada
Descripción	Generación del documento en una partida inactiva.
API REST	room/download/{id}
Entrada	-
Resultado esperado	Devolución de documento autogenerado con los datos de la partida.

CP10	Intentar generar documento en una partida en juego
Descripción	Intentar generar el documento en una partida aún en juego.
API REST	room/download/{id}
Entrada	-
Resultado esperado	Mensaje de error indicando que no se puede generar el documento hasta que no se termine la partida.

CP11	Crear jugador
Descripción	Creación de un jugador al unirse a una sala.
API REST	player
Entrada	▪ { 'nickname': 'player3Group1', 'roomCode': 'AAAAA', 'country': 'Spain', 'age': 19, 'state': 'NOGROUP' }
Resultado esperado	Creación del personaje y retorno de información de la sala.

CP12	Unirse a un grupo
Descripción	Cambiar o añadir a un grupo a un jugador.
API REST	player/update
Entrada	▪ { 'idPlayer': 15, 'idGroup': 15 }
Resultado esperado	Modificación del grupo al que pertenece el jugador en la base de datos.

CP13	Enviar elección
Descripción	Guardado de la respuesta de un jugador a una pregunta, junto con su argumentación.
API REST	form/choice
Entrada	▪ { 'round': 1, 'message': { 'idQuestion': 15, 'idAnswer': 37, 'message': 'Trama narrativa', 'typeChat': 'GROUP-CHAT', 'timeSend': '2020-11-14T22:45:00.000Z', 'typeMessage': 'POSITIVE', 'idPlayer': 1 } }
Resultado esperado	Guardado del mensaje y de la elección en la base de datos, y retorno del estado actual del grupo en la ronda.

CP14	Comprobar estado de la ronda
Descripción	Obtención del estado de los jugadores y el formulario en la ronda actual.
API REST	form/status
Entrada	▪ { 'roomCode': 'AAAAA', 'round': 1, 'numberGroup': 2 }
Resultado esperado	Retorno del estado actual del grupo en la ronda.

CP15	Obtener gráficas
Descripción	Solicitud de las gráficas con los resultados y sus patrones correspondientes.
API REST	graphics/get-results
Entrada	String[]
Resultado esperado	Coleción de datos con los puntos de las gráficas a representar.

CP16	Fin de ronda
Descripción	Finalización de la ronda y gestión de conflictos para los grupos que sea necesario.
API REST	room/end-round/{round}
Entrada	-
Resultado esperado	Populado del intante de finalización y generación de resultados para los grupos que se encuentren en conflicto.

CP17	Contraseñas no válidas
Descripción	Registro de usuario con contraseñas no válidas.
API REST	signup
Entrada	▪ {'firstName': 'Lucas', 'surname': 'Gonzalez', 'username': 'lucgonz', 'birthday': '1998-01-01', 'email': 'lucgonz@email.com', 'country': 'Spain', 'passwd': 'lucgonz'}
Resultado esperado	Mensaje de error infomando del problema.

7.5.2. Resultados de las pruebas

Tras realizar las pruebas, estos son los resultados obtenidos, y las soluciones aplicadas en las que han generado fallo:

Prueba	Resultado	Solución
CP01	OK	-
CP02	Fallo: no guarda un duplicado, pero marca la operación como exitosa	Añadir una comprobación previa para la existencia de la cuenta con un lanzamiento de excepción
CP03	OK	-
CP04	OK	-
CP05	OK	-
CP06	OK	-
CP07	OK	-
CP08	OK	-
CP09	OK	-
CP10	Fallo: Generaba el documento independientemente del estado	Añadida comprobación del estado de la sala.
CP11	OK	-
CP12	OK	-
CP13	OK	-
CP14	OK	-
CP15	OK	-
CP16	OK	-
CP17	Fallo: Permite la inserción de contraseñas de cualquier tipo	Añadido filtro para detectar si existe mínimo una letra y numero, y los caracteres especiales.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En esta sección se detallarán las conclusiones obtenidas en el desarrollo de este trabajo de fin de grado

- Se ha conseguido obtener los resultados de las simulaciones de Vensim.
- Se ha conseguido automatizar las simulaciones de Vensim y parsear sus datos a un formato mas legible y comprimido, reduciendo el peso en más 80 % de los ficheros.
- Se ha investigado sobre el funcionamiento del framework Springboot
- Se ha investigado sobre el funcionamiento de las tecnologías Postman, MySQL Workbench y MongoDB Compass
- Se ha conseguido una gamificación de la versión anterior del juego.
- Se ha conseguido implementar un backend basado en Springboot con conexión a dos bases de datos.
- Se ha diseñado el prototipo base del juego Crossroads II.
- Se ha conseguido reducir los tiempos de espera de los resultados considerablemente.
- Se ha conseguido desarrollar una aplicación bien estructurada en la que se puede añadir nueva funcionalidad sin mayor dificultad.
- Se ha conseguido obtener un informe de las partidas estructurado.
- Se ha conseguido solucionar el problema de los conflictos en las partidas.

8.2. Valoraciones personales

Este trabajo de fin de grado me ha permitido aprender múltiples tecnologías como Spring-Boot, Mockplus, MongoDB, Postman o el simulador de Vensim, y afianzar conocimientos previos que han sido enseñados a lo largo de la carrera. A nivel personal, este proyecto ha sido un trabajo muy duro, tanto por la situación personal como por el reto que suponía. Todas las herramientas a utilizar eran prácticamente nuevas y el proceso de aprendizaje ha sido lento y costoso, pero me ha enseñado nuevas tecnologías para desarrollo de aplicaciones web.

8.3. Trabajo futuro

En esta sección se mencionarán futuras funcionalidades o mejoras que no han podido desarrollarse y que podrían incluirse en las futuras versiones de la aplicación:

- Permitir que los usuarios registrados puedan unirse a partidas como jugadores, no solo como moderadores.
- Añadir niveles de dificultad, cambiando el número de preguntas, añadiendo mas información u opciones posibles.
- La posibilidad de poder realizar presentaciones al final de una ronda, donde los jugadores de cada grupo puedan exponer sus resultados y el por qué de sus decisiones.
- Permitir que los jugadores de distintos grupos puedan valorar los resultados o exposiciones de otros grupos, añadiendo más elementos de calificación e interacción entre ellos.
- Añadir un sistema de contactos, donde los usuarios se puedan agregar para jugar partidas en grupo.
- Mejorar la experiencia del jugador en la partida.

Apéndice A

Manual de despliegue

A.1. Instalación

En esta sección se detallaran los requisitos necesarios para poder poner en marcha el servidor que contendrá la aplicación y el proceso para realizarlo.

A.1.1. Requisitos

Será necesario tener:

- Java ver. 1.8
- Maven ver. 4.0
- MySQL Server ver. 5.7.31
- MongoDB 4.4.1
- Apache Tomcat ver. 9.0.24

Estas versiones son las utilizadas en el desarrollo de la aplicación.

A.1.2. Ejecución

La primera acción necesaria será descomprimir el archivo del proyecto y ejecutar desde la consola de comandos *mvn spring-boot:run*. Después de lanzar la aplicación, antes de realizar cualquier tipo de operación, será necesario abrir el servidor MySQL y ejecutar el script *populateMySQL.sql*, alojado en el repositorio Git. Una vez ejecutado, el siguiente paso es

hacer la llamada API REST de uploadPatterns y uploadGraphics. Los ficheros necesarios se encuentran dentro de la carpeta del Google Drive indicada en el Apéndice B. Para la primera llamada API será necesario utilizar como body el JSON descrito en el fichero *patterns.txt*, alojado en el repositorio git. Para las gráficas, será necesario enviar como parámetros los ficheros *dataTemp.txt*, *dataPIB.txt* y *dataToPattern.csv*. Estos ficheros poblarán la base MongoDB con los datos de las simulaciones de Vensim. Una vez realizadas estas operaciones, la aplicación esta lista para su uso. Para realizar las llamadas sólo habría que acceder a *localhost:8080* desde el navegador enviando los parámetros necesarios o desde una aplicación de tratamiendo de APIs como Postman.

A.2. Manual de programador

Toda la documentación necesaria para el uso de las APIs de la aplicación viene definida en la sección Estructura de las API REST. Para mayor información, la aplicación dispone de un enlace donde visualizar todas las interfaces gracias a Swagger. Una vez desplegado en local, el enlace es `http://localhost:8080/swagger-ui.html#/`

Apéndice B

Resumen de enlaces adicionales

Los documentos de interes de este Trabajo de fin de grado se encuentran en el siguiente repositorio git:

`https://gitlab.inf.uva.es/lucgonz/crossroads-backend`

En este repositorio podemos encontrar:

- Código fuente de la aplicación.
- Documento .PDF de la memoria del proyecto
- Fichero .sql con los datos iniciales de la base de datos MySQL.

Para poblar la base de datos de MongoDB seran necesarios los ficheros del siguiente enlace de Google Drive:

`https://drive.google.com/drive/folders/1gOKDDb-FAmwgrAK9K4NV49MJ-helC5IH?usp=sharing`

En la carpeta podemos encontrar:

- Datos de temperatura
- Datos del PIB
- Datos de los patrones
- Tabla de relacion entre simulacion y patrones

Bibliografía

[1] AJDPSowft. Tomcat, apache tomcat, jakarta tomcat. https://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769.

[2] Atlassian. Sourcetree. https://www.sourcetreeapp.com/.

[3] Céline Bellard, Cleo Bertelsmeier, Paul Leadley, Wilfried Thuiller, and Franck Courchamp. Impacts of climate change on the future of biodiversity. *Ecology Letters*, 15(4):365–377, 2012.

[4] Iñigo Capellán-Pérez, Ignacio de Blas, Jaime Nieto, Carlos de Castro, Luis Javier Miguel, Óscar Carpintero, Margarita Mediavilla, Luis Fernando Lobejón, Noelia Ferreras-Alonso, Paula Rodrigo, Fernando Frechoso, and David Álvarez Antelo. Medeas: a new modeling framework integrating global biophysical and socioeconomic constraints. *Energy Environ. Sci.*, 13:986–1017, 2020.

[5] Iñigo Capellán-Pérez, David Álvarez Antelo, and Luis J. Miguel. Global sustainability crossroads: A participatory simulation game to educate in the energy and sustainability challenges of the 21st century. *Sustainability*, 11(13):3672, Jul 2019.

[6] LOCOMOTION consortium. Global sustainability crossroads ii. https://www.locomotion-h2020.eu/locomotion-models/global-sustainability-crossroads-ii/.

[7] Rocket.Chat Technologies Corp. Rocketchat. https://rocket.chat/es/.

[8] Oracle Corporation. Mysql workbench. https://www.mysql.com/products/workbench/.

[9] Víctor Cuervo. ¿qué es postman? http://www.arquitectoit.com/postman/que-es-postman/.

[10] Facebook. Whatsapp app. https://www.whatsapp.com/?lang=es.

[11] Kirill Fakhroutdinov. The unified modeling languaje. https://www.uml-diagrams.org/.

[12] Jackbox Games. What is jackbox? https://www.jackboxgames.com/what-is-jackbox/.

[13] LOCOMOTION H2020. Low-carbon society: an enhanced modelling tool for the transition to sustainability. https://www.locomotion-h2020.eu/about-project/overview/, 2020.

[14] Sublime HQ. Sublime text 3. http://www.sublimetext.com/.

[15] iMisut. Reseña: The resistance: Avalon. https://misutmeeple.com/2015/01/resena-resistance-avalon/, 2015.

[16] MongoDB Inc. Mongodb. https://www.mongodb.com/.

[17] I. Jacobson, G. Booch, and J. Rumbaugh. *El proceso unificado de desarrollo de software*. Fuera de colección Out of series. Pearson Educación, 2000.

[18] JetBrains. Intellij idea community. https://www.jetbrains.com/es-es/idea/.

[19] John Hammersley & John Lees-Miller. Overleaf documentation. https://www.overleaf.com/learn.

[20] Mark Mc Mahon and Contributors. Pywinauto. https://pywinauto.readthedocs.io/en/latest/, 2006-2018.

[21] Microsoft. Ms project. https://www.microsoft.com/es-es/microsoft-365/project/project-management-software.

[22] Microsoft. Skype. https://www.skype.com/es/.

[23] NASA. The effects of climate change. https://climate.nasa.gov/effects/, 2020.

[24] Rubén Pahino. ¿qué son spring framework y spring boot? https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx#:~:text=Spring%20Boot%20permite%20compilar%20nuestras,de%20aplicaciones%20en%20el%20propio%20.

[25] LaTeX Project. Latex main page. https://www.latex-project.org/.

[26] SmartBear Software. Swagger documentation. https://swagger.io/.

[27] Free Software Foundation & Richard Stallman. Gnu general public license v3. https://www.gnu.org/licenses/gpl-3.0.html, 25/02/1989.

[28] Al Sweigart. Pyautogui. https://pyautogui.readthedocs.io/en/latest/.

[29] Ventana System. Vensim dll overview. https://www.vensim.com/documentation/index.html?dss_dll.htm.

[30] Ventana System. Vensim official webpage. https://vensim.com/.

[31] Sai Upadhyayula. Programming techie. https://programmingtechie.com/2020/05/14/building-a-reddit-clone-with-spring-boot-and-angular/.

[32] Guido van Rossum. Python. https://www.python.org/.

[33] Emiliano Zublena. Api-first development. https://medium.com/@emilianozublena/api-first-development-c202a61cf3b2.