Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

Desarrollo del front-end y mejoras en el back-end de un juego didáctico multijugador de competición y consenso sobre el cambio climático

---

Alumno: Manuel Alda Peñafiel

Tutores: Yania Crespo González-Carvajal
David Escudero Mancebo

A mi familia

Agradecimientos

A mi familia y amigos, que siempre han estado a mi lado en los buenos y en los malos momentos.

A mis tutores, Yania y David, por la gran dedicación que han mostrado durante todo este proyecto.

A Elena Rodríguez, Adrián Manzano y María Galindo, porque sin su colaboración no habría sido posible terminar este proyecto.

A mis compañeros de carrera, que me han ayudado a sacar lo mejor de mi mismo día a día.

Gracias a todos.

Resumen

El cambio climático es uno de los principales desafíos a los que se enfrentará la humanidad en los próximos años. Este proyecto se encuentra dentro del marco del proyecto LOCOMOTION, que busca la realización de actividades de concienciación para sensibilizar a la sociedad sobre el problema y la repercusión de las acciones individuales en el mismo.

El propósito de este proyecto es finalizar el desarrollo de un juego didático, multijugador, cooperativo y de competición sobre el cambio climático, mediante el desarrollo de un front-end y la modificación del back-end ya existente del juego. Está enfocado, principalmente, al ámbito educativo, principalmente Institutos de Educación Secundaria y Bachillerato.

El proyecto se ha desarrollado utilizando los frameworks Angular y Spring Boot, siguiendo siguiendo las guías del marco de trabajo ágil Scrum.

Como resultado se ha desarrollado un software llamado Crossroads, que se ha publicado bajo licencia GNU GPL v3 y que está listo para ser empleado en entornos educativos.

Abstract

Climate change is one of the main challenges that humanity will face in the coming years. This project is within the framework of the LOCOMOTION project, which seeks to carry out awareness-raising activities to sensitize society about the problem and the impact of individual actions on it.

The purpose of this project is to finalize the development of an educational, multiplayer, cooperative and competitive game on climate change, by developing a front-end and modifying the existing back-end of the game. It is focused mainly on the educational field, mainly (Senior) High Schools.

The project has been developed using the Angular and Spring Boot frameworks, following the guidelines of the agile framework Scrum.

As a result, a software called Crossroads has been developed, which has been released under the GNU GPL v3 license and is ready to be used in educational environments.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

En la actualidad el cambio climático es uno de los grandes problemas que afecta a nuestra sociedad. Según algunos expertos, nos estamos acercando a un punto de no retorno, a partir del cual cualquier acción que realicemos para tratar de remediar el problema no tendrá efecto alguno [48]. Como respuesta a este problema, han surgido numerosas propuestas que buscan concienciar a la población sobre los impactos de nuestras acciones en el medio ambiente para tratar de revertir la tendencia actual del cambio climático. Una de estas propuestas es el proyecto europeo LOCOMOTION [36], que pretende diseñar un conjunto de IAM (Modelos de evaluación integrados) que proporcionen una manera de evaluar la viabilidad, efectividad, costos y ramificaciones de diferentes opciones de políticas de sostenibilidad. A partir de estos IAM desarrollados en LOCOMOTION, se pretende construir un grupo de herramientas para acercar sus resultados a diferentes ámbitos de la sociedad.

Este Trabajo de Fin de Grado (TFG) surge como continuación de un trabajo de fin de grado anterior que se encuentra en el marco de dicho proyecto LOCOMOTION [37]. En dicho TFG se abordó la adaptación de una actividad educativa definida en el grupo GEEDS llamada Crossroads [21] que se apoya en el IAM desarrollado en un proyecto predecesor de LOCOMOTION: el proyecto MEDEAS [30]. Con dicha adaptación se pasaba de una actividad educativa a una primera versión de un juego educativo.

1.2. Motivación

Este proyecto busca la creación de una herramienta educativa, en forma de juego multijugador online que permita experimentar a los usuarios el efecto de sus acciones en el desarrollo del cambio climático. El público objetivo de este juego es, por tanto, personas pertenecientes al ámbito educativo, profesores y alumnos principalmente de institutos de Educación

Secundaria y Bachillerato.

El juego deberá responder tanto a las necesidades del profesor como a las de los alumnos. Además, deberá fomentar el trabajo en grupo y el diálogo entre alumnos. También deberá incluir elementos de juego que permitan favorecer la implicación activa de los estudiantes.

1.3. Objetivos

Los objetivos que se pretende haber alcanzado al término del proyecto son los siguientes:

- Desarrollar un front-end en Angular para el back-end ya desarrollado previamente en [37].
- Modificar dicho back-end cuando sea necesario para adaptarlo a las necesidades específicas del front-end o a necesidades no previstas en el desarrollo inicial.
- Desarrollar una primera versión funcional del juego Crossroads.
- Desplegar el back-end y el front-end del juego en una máquina virtual de la escuela para que pueda ser accesible al público.
- Añadir nueva funcionalidad al juego para dar cobertura a casos no previstos en las versiones iniciales.
- Aprendizaje en profundidad del framework de desarrollo web Angular.
- Aprendizaje del marco de trabajo ágil *Scrum* y sus principios básicos.

1.4. Estructura de la memoria

A continuación, se presenta la estructura de capítulos de la presente memoria, detallando para cada uno de ellos el contenido de los mismos:

Capítulo 2: Requisitos y planificación. Descripción del marco general de gestión del proyecto y requisitos del mismo.

Capítulo 3: Tecnologías utilizadas. Documentación de las herramientas utilidadas durante todo el proyecto, tanto para la gestión como para el desarrollo del mismo.

Capítulo 4: Análisis. Explicación de la evolución del modelado conceptual y de interacción con respecto al anterior TFG.

Capítulo 5: Diseño. Descripción del diseño final de la aplicación. Abarca desde la arquitectura de la aplicación y la integración de la misma con el servidor ya desarrollado, hasta el diseño visual de las pantallas de la aplicación.

Capítulo 6: Implementación y pruebas. Detalle del proceso de implementación y testing de la aplicación.

Capítulo 7: Seguimiento del proyecto. Documentación del avance del proyecto a lo largo de su duración.

Capítulo 8: Conclusiones y trabajo futuro. Exposición de las conclusiones alcanzadas al término del proyecto y revisión del cumplimiento de los objetivos. Incluye algunas propuestas de trabajo para el futuro.

Bibliografía. Referencias bibliográficas consultadas para la realización de este proyecto.

Anexo A. Manuales: Incluye algunos manuales de interés, como la guía de despliegue o el manual de usuario.

Anexo B Resumen de enlaces adicionales: Incluye los enlaces de interés sobre el proyecto, como, por ejemplo, el repositorio en el que se encuentra el código.

Capítulo 2

Requisitos y planificación

2.1. Scrum

2.1.1. Definición de Scrum y pilares sobre los que se sustenta

Scrum [35] es un marco de trabajo que define una manera de realizar la gestión de un proyecto. Se encuentra enmarcado dentro de los llamados procesos de desarrollo ágiles cuyo objetivo es reducir la complejidad en el desarrollo de proyectos mediante la mejora de la comunicación entre las partes implicadas en el desarrollo, así como la colaboración de los mismos. Scrum se sustenta sobre tres pilares:

- **Transparencia**. Es necesario hacer visibles todos los aspectos relevantes del desarrollo del proyecto para todos los interesados en el resultado del mismo.
- **Inspección**. Todos los involucrados en el marco de Scrum deben inspeccionar frecuentemente el progreso para detectar y corregir variaciones no deseadas en el desarrollo del proyecto
- **Adaptación**. Una vez detectadas las desviaciones en el desarrollo del proyecto, es necesario adaptar los procesos y artefactos de Scrum para reducir el impacto de dichas desviaciones.

Los componentes del marco de Scrum (Figura 2.1) son los roles, artefactos y eventos, de los cuales hablaremos en las siguientes secciones.

2.1.2. Artefactos

Los artefactos que se utilizan en el marco Scrum son los siguientes:

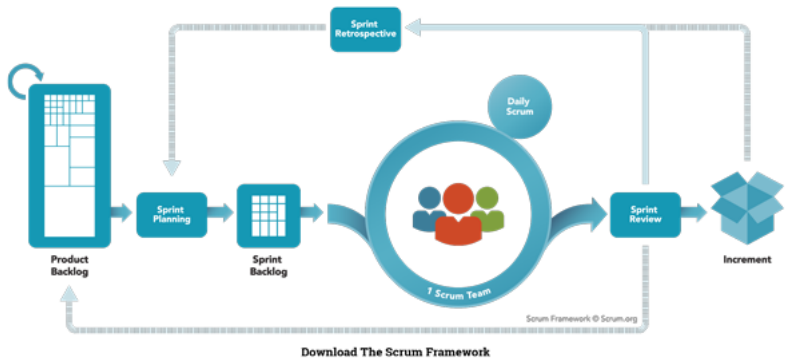

Figura 2.1: Marco general de Scrum

- **Product Backlog**. Lista ordenada de requisitos y características (o historias de usuario) que el cliente desea que tenga el producto final. En los primeros momentos del proyecto se crea una versión inicial que se va refinando a lo largo del desarrollo.
- **Sprint Backlog**. Conjunto de historias de usuario que se van a realizar en un sprint. La definición de sprint se presentará en la Sección 2.1.4 .
- **Incremento**. Parte del Product Backlog que se ha desarrollado durante un sprint. Al final de cada sprint el incremento debe estar en condiciones de poder ser utilizado.

2.1.3. Roles

Los roles que desempeñan los miembros del equipo Scrum son los siguientes:

- **Product Owner**. Es el responsable de maximizar el valor del producto. Es por esto que se encarga de gestionar el Product Backlog. Suele actuar en representación del cliente.
- **Scrum Master**. Es la persona que se encarga de gestionar todos los procesos de Scrum para que estos se adopten por parte del equipo.
- **Equipo de desarrollo**. Equipo habitualmente formado por entre 3 y 9 personas que se encargan de desarrollar el producto.

2.1.4. Eventos

Scrum establece una serie de eventos regulares con el objetivo de mejorar la comunicación entre los miembros del equipo Scrum y reducir el número de reuniones no planificadas. Los eventos tienen una duración fija.

- **Sprint**. Período de duración de 1 a 4 semanas en las que se produce un incremento potencialmente utilizable y desplegable. La duración de los sprints se recomienda que sea fija a lo largo del desarrollo del proyecto. Además, los requisitos del sprint se mantienen fijos mientras dure el sprint. Cada sprint comienza en cuanto termina el sprint anterior (en el caso del primero, en cuanto se puede comenzar el proyecto).
- **Sprint Planning**. Reunión en la que se establece el trabajo a realizar durante el sprint. En ella participan todos los miembros del equipo Scrum.
- **Sprint Goal**. Es la meta que se establece durante el Sprint Planning y que se consigue mediante la implementación de las historias de usuario del Sprint Backlog.
- **Daily Scrum**. Reunión diaria de unos 15 minutos en la que el equipo de desarrollo pone en común las actividades realizadas durante el día anterior y establece un plan para las 24 horas siguientes.
- **Sprint Review**. Reunión que se realiza al final de cada sprint. En esta reunión se revisa el incremento desarrollado durante el sprint y, si es necesario, se ajusta el Product Backlog.
- **Sprint Retrospective**. Reunión que tiene lugar entre el Sprint Review y el Sprint Planning del siguiente sprint. En esta reunión el equipo Scrum se autoevalúa para comprobar qué ha ido bien y qué no durante el sprint, a la vez que se establece un plan de mejoras que serán abordadas en el siguiente sprint.

2.2. Adaptación del marco de Scrum a este proyecto

Antes de poder aplicar el marco general de Scrum, es necesario realizar una adaptación del mismo a las características de este proyecto. En primer lugar, hay que adaptar los roles de los miembros. En el contexto de este TFG, los tutores del mismo desempeñarán los roles de *Scrum Master* y *Product Owner*, mientras que el alumno conformará el *Equipo de desarrollo*.

En segundo lugar, se ha establecido un tamaño de sprint de dos semanas. Esto, junto con la estimación en horas de trabajo realizada en la guía docente del TFG [68] y la estimación del número de sprints a realizar para poder finalizar el TFG (unos 10 sprints) nos da unas 30 horas de trabajo por cada sprint y unas 15 horas de trabajo semanales.

En cuanto a los eventos de Scrum que hay que realizar, el *Daily Scrum* se ha sustituido por una reunión semanal que tendrá lugar todos los martes, debido a que el equipo de desarrollo está formado por una sola persona. Por tanto, en cada sprint se realizan dos reuniones weekly, una para comprobar el progreso realizado a mitad de sprint y otra reunión final en la que se realizan el *Sprint review* y el *Sprint retrospective*, así como el *Sprint planning* del siguiente sprint.

Finalmente, se ha decidido utilizar una estrategia de estimación del peso de las historias de usuario basada en una escala de puntos lineal que va desde 1 a 5, donde 5 es el máximo de puntos que puede recibir una historia y en la que cada punto de historia equivale a 5 horas de trabajo.

2.3. Product Backlog inicial

En la Tabla 2.1 se puede observar la primera versión del product backlog inicial. Como las tres historias de usuario que contiene son historas épicas que abarcan mucha funcionalidad como para abordarla en un solo sprint, se ha decidido refinarlas en historias de usuario más pequeñas, dando lugar a la versión de partida del product backlog (Tablas 2.2, 2.3 y 2.4)

Este product backlog no es definitivo y se irá refinando durante el desarrollo del proyecto, añadiendo nuevas historias de usuario o modificando alguna de las ya existentes.

Número	Descripción
1	Como usuario quiero poder crear una cuenta en Crossroads y poder gestionarla.
2	Como usuario registrado quiero poder crear una sala de partida y poder moderarla.
3	Como jugador quiero poder unirme a una partida y poder jugarla.

Tabla 2.1: Historias de usuario épicas

Número	Descripción
1.1	Como usuario sin registrar quiero poder crear una cuenta en Crossroads.
1.2	Como usuario registrado quiero poder iniciar sesión en mi cuenta.
1.3	Como usuario registrado quiero poder recuperar mis credenciales.
1.4	Como usuario registrado quiero ver los datos del perfil de mi cuenta.
1.5	Como usuario registrado quiero editar los datos del perfil de mi cuenta.

Tabla 2.2: Historias de usuario de la Historia épica 1

Número	Descripción
2.1	Como usuario registrado quiero poder crear una sala de partida.
2.2	Como usuario moderador quiero poder empezar la nueva partida.
2.3	Como usuario moderador quiero poder empezar una nueva ronda.
2.4	Como usuario moderador quiero poder finalizar la ronda.
2.5	Como usuario moderador quiero poder terminar la partida.
2.6	Como usuario moderador quiero poder generar un informe con los resultados de la partida.
2.7	Como usuario moderador quiero poder enviar mensajes en el chat global.

Tabla 2.3: Historias de usuario de la Historia épica 2

Número	Descripción
3.1	Como usuario jugador quiero poder unirme a una sala de juego mediante un código.
3.2	Como usuario jugador quiero poder responder a las preguntas de cada ronda.
3.3	Como usuario jugador quiero poder enviar mensajes en el chat grupal.
3.4	Como grupo quiero poder enviar una propuesta para cada conjunto de medidas.
3.5	Como usuario jugador quiero poder resolver los conflictos que puedan surgir en el grupo a la hora de elaborar la propuesta.
3.6	Como usuario jugador quiero poder enviar mensajes en el chat global.
3.7	Como usuario jugador quiero poder ver los resultados de la ronda actual.
3.8	Como usuario jugador quiero poder ver los resultados finales de la partida.

Tabla 2.4: Historias de usuario de la Historia épica 3

2.4. Análisis de riesgos

Un riesgo [59] es un evento o condición inciertos que, de ocurrir, pueden tener un efecto positivo o negativo en la consecución de los objetivos del proyecto. Los riesgos hacen referencia a eventos futuros que tienen una causa y un efecto asociados.

Para evitar las consecuencias adversas que tendría la materialización de algunos errores, es necesario realizar un buen análisis e identificación de riesgos, así como establecer una serie de medidas de mitigación (que buscan evitar que se produzca el riesgo) y un plan de contingencia (que busca minimizar el impacto del riesgo cuando ocurre) para cada riesgo.

En las Tablas 2.5 a 2.13 se presenta el análisis de riesgos realizado. Para cada riesgo se presenta un título, una breve descripción, la probabilidad de ocurrir, el impacto del riesgo (tanto probabilidad como impacto en la escala bajo, medio y alto), las medidas de mitigación y el plan de contingencia.

Título	Planificación optimista
Descripción	Realización de una planificación poco realista que, junto con el aumento del tiempo de desarrollo de algunas tareas, desemboca en el no cumplimiento de la planificación del sprint
Probabilidad	Media
Impacto	Medio
Medidas de mitigación	Adición de un margen temporal en cada sprint que cubra el posible aumento del tiempo de desarrollo de algunas tareas
Plan de contingencia	Pasar las tareas que no se han podido completar al siguiente sprint y descartar la funcionalidad menos prioritaria si no es posible implementarla

Tabla 2.5: Riesgo de planificación optimista de un sprint

Título	Enfermedad del desarrollador
Descripción	El desarrollador contrae alguna enfermedad y no puede trabajar en el proyecto
Probabilidad	Bajo
Impacto	Alto
Medidas de mitigación	Planificación de dos sprints de refuerzo en el que poder abordar las tareas que se viesen afectadas
Plan de contingencia	Retrasar las tareas afectadas hasta el siguiente sprint

Tabla 2.6: Riesgo de enfermedad del desarrollador

Título	Cambio en las historias de usuario
Descripción	Se producen cambios en la funcionalidad a desarrollar una vez iniciado el proyecto
Probabilidad	Alta
Impacto	Medio
Medidas de mitigación	Gestión del proyecto mediante un proceso de desarrollo ágil basado en Scrum que permite adaptarse mejor a los cambios
Plan de contingencia	Modificar la prioridad de las historias de usuario y replanificar

Tabla 2.7: Riesgo de cambio en las historias de usuario

Título	Necesidad de cambiar el back-end
Descripción	La funcionalidad desarrollada del back-end no se adapta a las necesidades del front-end
Probabilidad	Alta
Impacto	Medio
Medidas de mitigación	Adaptar lo máximo posible la funcionalidad del front-end a la funcionalidad del back-end ya desarrollado
Plan de contingencia	Añadir las tareas correspondientes a la modificación del back-end al sprint actual o al siguiente, dependiendo de la prioridad de los cambios

Tabla 2.8: Riesgo de necesidad de cambiar el back-end

Título	Falta de formación
Descripción	El equipo de desarrollo no tiene los conocimientos necesarios acerca de las tecnologías a utilizar en el desarrollo del proyecto
Probabilidad	Alta
Impacto	Bajo
Medidas de mitigación	Realización de tutoriales y mini proyectos de ejemplo para ir aprendiendo a manejar las tecnologías que sean desconocidas
Plan de contingencia	Pedir ayuda

Tabla 2.9: Riesgo de falta de formación

Título	Desarrollo de funcionalidad incorrecta
Descripción	El equipo de desarrollo realiza una implementación que no se ciñe correctamente a las necesidades del cliente
Probabilidad	Media
Impacto	Alto
Medidas de mitigación	Realización de reuniones periódicas con el *Product Owner* para comprobar que la funcionalidad desarrollada coincida con las necesidades del cliente
Plan de contingencia	Cambiar prioridades de las historias de usuario y replanificar

Tabla 2.10: Riesgo de desarrollo de funcionalidad incorrecta

Título	Desarrollo de una interfaz de usuario poco usable
Descripción	El equipo desarrolla una interfaz de usuario poco amigable para el usuario o difícil de entender
Probabilidad	Media
Impacto	Medio
Medidas de mitigación	Realización periódica de pruebas de usabilidad con potenciales usuarios de la aplicación
Plan de contingencia	Añadir tareas nuevas en las que abordar los cambios en el siguiente sprint

Tabla 2.11: Riesgo de desarrollo de una interfaz de usuario poco usable

Título	Agravamiento en las condiciones actuales de la pandemia del Covid-19
Descripción	Las situación actual de la pandemia del coronavirus empeora, repercutiendo en algunas tareas
Probabilidad	Alta
Impacto	Medio
Medidas de mitigación	Procurar hacer todo el trabajo posible de forma telemática
Plan de contingencia	Revisar la planificación del proyecto

Tabla 2.12: Riesgo de agravamiento en la situación de la pandemia

Título	Conflictos de dependencias entre módulos de node
Descripción	Surge un conflicto de dependencias entre paquetes utilizados por la aplicación que impide ejecutarla
Probabilidad	Media
Impacto	Medio
Medidas de mitigación	Utilizar las dependencias mínimas necesarias
Plan de contingencia	Añadir una tarea al sprint actual en la que resolver el conflicto de dependencias

Tabla 2.13: Riesgo de conflicto de dependencias

2.5. Planificación inicial de los sprints

Teniendo en cuenta los aspectos definidos en la Sección 2.2 acerca del tamaño, duración y demás características de cada sprint, así como la estimación inicial del número de sprints, hemos elaborado una tabla con la planificación inicial para cada sprint (Tabla 2.14).

Número	Fecha inicio	Fecha fin	Observaciones
0	19/01/2021	09/02/2021	Sprint de preparación
1	09/02/2021	23/02/2021	
2	23/02/2021	09/03/2021	
3	09/03/2021	23/03/2021	
4	23/03/2021	13/04/2021	
5	13/04/2021	27/04/2021	
6	27/04/2021	11/05/2021	
7	11/05/2021	25/05/2021	
8	25/05/2021	08/06/2021	
9	08/06/2021	22/06/2021	Sprint de refuerzo
10	22/06/2021	06/07/2021	Sprint de refuerzo

Tabla 2.14: Planificación inicial de los sprints

2.6. Presupuesto inicial

Para el cómputo del presupuesto inicial del proyecto tendremos en cuenta costes de material, coste laboral y coste de recursos.

Según el portal web `https://es.indeed.com/`, el sueldo medio de un desarrollador de front-end a fecha de 15 de julio de 2021 es de 28.729 € al año. Las empresas tienen que pagar a la seguridad social un 31 % [33] aproximadamente. Por tanto, el coste que tiene que asumir

la empresa al año asciende a 37.634,99 €/año. Teniendo en cuenta que la jornada laboral de un empleo de este tipo es de 1800 horas/año, el coste por hora es de 20,91 €. Como la guía docente del trabajo de fin de grado [68] contempla una carga de trabajo de 300 horas, obtenemos un coste laboral de 6273 €.

En cuanto a coste material, se ha decidido emplear un ordenador portátil HP por un coste de 800 €. También se ha empleado una máquina virtual proporcionada por la universidad. Consultando los precios de mercado para una máquina virtual de características similares [41], el precio de uso de la máquina asciende a 0,0043 €/hora, es decir, 1,29 € por las 300 horas planificadas. Con respecto al tema de licencias de software, se toma la decisión de que todo producto software a utilizar sea gratuito (o al menos la versión que se utilice sea su versión gratuita).

Finalmente, el coste de recursos (agua, luz, calefacción, etc) se ha estimado en 0 €, puesto que estos corren por cuenta de agentes externos al proyecto.

Teniendo en cuenta todo lo anterior, se obtiene un presupuesto inicial de 7073 €. En la Tabla 2.15 se puede observar un resumen de este presupuesto.

Tipo de coste	Coste estimado (€)
Laboral	6273
Material	801,29
Recursos	0
Total	7074,29

Tabla 2.15: Presupuesto simulado

2.7. Product Backlog final

Al término del presente proyecto, y después de realizar las modificaciones requeridas por las necesidades del *Product Owner* en las historias de usuario del backlog inicial, el estado del product backlog es el siguiente (Tablas 2.16, 2.17 y 2.18):

Este *Product Backlog* surge como la evolución de algunas historias de usuario, que se han tenido que dividir para especificar mejor la funcionalidad, pero también por la adición de algunas historias de usuario completamente nuevas. Se han añadido las historias 2.9 a 2.12 y se han dividido las historias 2.10 y 3.5.

Número	Descripción
1.1	Como usuario sin registrar quiero poder crear una cuenta en Crossroads.
1.2	Como usuario registrado quiero poder iniciar sesión en mi cuenta.
1.3	Como usuario registrado quiero poder recuperar mis credenciales.
1.4	Como usuario registrado quiero ver los datos del perfil de mi cuenta.
1.5	Como usuario registrado quiero editar los datos del perfil de mi cuenta.

Tabla 2.16: Historias de usuario de la Historia épica 1 en el product backlog final

Número	Descripción
2.1	Como usuario registrado quiero poder crear una sala de partida.
2.2	Como usuario moderador quiero poder empezar la nueva partida.
2.3	Como usuario moderador quiero poder empezar una nueva ronda.
2.4	Como usuario moderador quiero poder finalizar la ronda.
2.5	Como usuario moderador quiero poder terminar la partida.
2.6	Como usuario moderador quiero poder generar un informe con los resultados de la partida.
2.7	Como usuario moderador quiero poder enviar mensajes en el chat global.
2.8	Como usuario moderador quiero ver los jugadores que están en cada grupo con la información del estado en que se encuentran.
2.9	Como usuario moderador quiero poder ver la información sobre el desarrollo de cada ronda.
2.10a	Como usuario moderador quiero poder acceder a un listado de las salas que he creado.
2.10b	Como usuario moderador quiero poder acceder al detalle de una de las salas que he creado.
2.11	Como usuario moderador quiero poder enviar mensajes en el chat grupal de un grupo que haya seleccionado.
2.12	Como usuario moderador quiero poder expulsar a los miembros de un grupo.

Tabla 2.17: Historias de usuario de la Historia épica 2 en el product backlog final

Número	Descripción
3.1	Como usuario jugador quiero poder unirme a una sala de juego mediante un código.
3.2	Como usuario jugador quiero poder responder a las preguntas de cada ronda.
3.3	Como usuario jugador quiero poder enviar mensajes en el chat grupal.
3.4	Como grupo quiero poder enviar una propuesta para cada conjunto de medidas.
3.5a	Como usuario jugador quiero poder ver los conflictos que se han producido al elaborar la propuesta.
3.5b	Como usuario jugador quiero poder resolver los conflictos.
3.5c	Como usuario jugador quiero poder ver los comentarios a favor y en contra que ha recibido cada pregunta con conflicto.
3.6	Como usuario jugador quiero poder enviar mensajes en el chat global.
3.7	Como usuario jugador quiero poder ver los resultados de la ronda actual.
3.8	Como usuario jugador quiero poder ver los resultados finales de la partida.

Tabla 2.18: Historias de usuario de la Historia épica 3 en el product backlog final

Capítulo 3

Tecnologías utilizadas

En este Capítulo se presenta un resumen de las tecnologías utilizadas tanto para la gestión del proyecto, como para su desarrollo.

3.1. Tecnologías para la gestión del proyecto

3.1.1. Overleaf

Overleaf [53] es un editor online y colaborativo de *LaTeX* [67] que ofrece la posibilidad de editar el documento de manera simultánea por todos los miembros del equipo que participan en el documento. Es utilizado comúnmente en la elaboración de trabajos académicos y en este proyecto se ha empleado para la redacción de este documento.

3.1.2. Rocket.Chat

Rocket.Chat [60] es una aplicación de comunicación de código abierto de características similares a Slack. En este proyecto se utilizó para mantener una comunicación fluida entre los integrantes del equipo *Scrum* entre las reuniones semanales.

3.1.3. Webex

Cisco Webex [14] es una aplicación de comunicación que permite la realización de videoconferencias con características añadidas como la posibilidad de compartir pantalla. Muy utilizada en el ámbito académico durante la pandemia del Covid-19, está aplicación se utilizó para realizar las reuniones semanales de seguimiento del proyecto.

3.1.4. Git

Git es un software de control de versiones que permite almacenar y visionar todos los cambios realizados en los artefactos del proyecto a lo largo de toda su vida. Para este proyecto se han utilizado dos repositorios remotos almacenados en el Gitlab de la escuela, uno para el back-end y otro para el front-end.

El flujo de trabajo que se utilizó fue el modelo de ramas Gitflow [69], en el cual, se tienen dos ramas principales, master y develop. Con cada historia de usuario que se esté desarrollando, se crea una rama desde develop para trabajar en ella. Una vez finalizado el trabajo en esa rama, se realiza una fusión de los cambios de dicha rama en develop. Con cada incremento de funcionalidad que está listo para mostrar al cliente, se realiza una fusión de develop en master. La principal diferencia entre estas dos ramas reside en la estabilidad de los cambios, siendo la funcionalidad de la rama master mucho más estable que la de develop, que está sujeta más a cambios.

3.1.5. Gitlab y Gitlab Issue Tracker

Gitlab [27] es una plataforma que permite la creación y gestión de un repositorio Git remoto. Entre las múltiples herramientas que ofrece, se encuentra el Issue Tracker, una herramienta que actúa a modo de tablón, en el que se pueden añadir tareas para realizar, estimar el esfuerzo requerido para llevar a cabo cada tarea (en horas y minutos) y llevar el control del esfuerzo real utilizado para finalizar las tareas. En este proyecto se utilizó para llevar un control del desarrollo de las tareas que había que realizar en cada sprint.

3.1.6. Gitlab CI/CD

Gitlab CI/CD [28] es una herramienta que proporciona Gitlab para llevar a cabo tareas rutinarias a lo largo del proyecto, de forma automática. Para ello, hay que añadir un script al repositorio en el que se define el código necesario para realizar las tareas y la rama y la fase del proyecto en la que se lleva a cabo dicha tarea. En este proyecto se utilizó para llevar a cabo la compilación de los artefactos del front-end, así como para la realización del despliegue de dichos artefactos en un servidor web Apache en una máquina virtual de la escuela.

3.2. Tecnologías para el desarrollo del proyecto

3.2.1. Visual Studio Code

Visual Studio Code [43] es un editor de código desarrollado por Microsoft que permite una gran personalización de la experiencia de usuario mediante la instalación de extensiones. Es uno de los editores de código más versátiles y utilizados actualmente. En este proyecto se ha utilizado para el desarrollo del código del front-end en Angular.

3.2.2. Visual Paradigm

Visual Paradigm [70] es una herramienta de modelado de Software que permite la creación de diagramas UML (Unified Modeling Language). UML [49] es el lenguaje de modelado de sistemas software estándar. Esta herramienta se ha utilizado para la creación de los diagramas de análisis y diseño de la aplicación.

3.2.3. Node.js y NPM

Node.js [50] es un entorno de ejecución para JavaScript construido con el motor V8 de Google Chrome. Aunque está pensado para ejecutar código del lado del servidor, también puede ejecutar código de la parte de cliente. En este proyecto se ha utilizado para dar soporte al desarrollo y las pruebas del código Angular durante el tiempo de desarrollo.

Node Package Manager [73] (NPM) es un gestor de paquetes desarrollado en JavaScript que permite añadir nuevos módulos y paquetes a las aplicaciones NodeJS que estemos desarrollando. A continuación se presenta un pequeño detalle de los módulos más importantes que se han tenido que añadir al proyecto. De este listado se excluyen los módulos que añade el propio Angular al crear la aplicación.

Chart.js

Chart.js [13] es una biblioteca JavaScript gratuita que permite la creación de gráficas para la visualización de datos de manera rápida y sencilla. En el proyecto se utilizó para mostrar unas gráficas con los resultados a los grupos de jugadores.

Filesaver.js

Filesaver.js [25] es una biblioteca que permite almacenar un fichero descargado en el almacenamiento local del dispositivo. La carpeta por defecto en la que almacena los ficheros descargados es la carpeta de descargas del dispositivo. En el proyecto se utilizó para implementar la descarga de un fichero con los datos de la partida.

Node MySQL

Node MySQL [24] es una biblioteca que permite establecer una conexión con una base de datos MySQL y ejecutar comandos SQL en dicha base de datos. En este proyecto se utilizó para crear un plugin que implementase el borrado y sembrado de datos en la base MySQL que utiliza el back-end. Este plugin se utiliza en el testing e2e de la aplicación, para resetear la base de datos entre test y test y así poder hacer que los test sean independientes unos de otros.

PrimeNG

PrimeNG [54] es una biblioteca para Angular muy utilizada actualmente que proporciona gran cantidad de componentes visuales y plantillas personalizables muy fácilmente. En este proyecto se han utilizado, principalmente, un componente tabla [57], el paginador para las tablas [57] y un panel [55] que actúa a modo de pop-up en ciertas partes de la aplicación.

Cypress Code Coverage

Este plugin desarrollado para Cypress [16] permite la obtención de un informe de cobertura de código para los tests end-to-end. En este proyecto se ha utilizado para obtener el informe de cobertura de los tests presentado en el Capítulo 6.

3.2.4. Cypress

Cypress [17] es un framework que permite la realización de tests unitarios y e2e (end-to-end) de forma sencilla, automática y muy amigable con el desarrollador. El testing e2e [58] es un tipo de testing en el cual se emula el comportamiento del usuario a la hora de manejar la aplicación, realizando las acciones que realizaría dicho usuario en la interfaz de la app. El nombre end-to-end proviene del hecho de que, con estos test, no se prueban partes individuales de la aplicación, sino la aplicación entera, tanto el front-end, como el back-end. En este proyecto Cypress se ha utilizado a la hora de crear la batería de test e2e de la aplicación.

3.2.5. Angular

Angular [3] es un framework para crear aplicaciones web y móviles desarrollado en TypeScript y mantenido por Google. Se utiliza para crear aplicaciones de una sola página (SPA [72]), en la que todas las vistas se encuentran en la misma página y todo el código se carga de una sola vez, o de forma dinámica, cuando las acciones realizadas por el usuario requieran de la carga de ciertos recursos. De esta manera, la aplicación contiene un único fichero HTML llamado index.html en el que se van insertando el código de las vistas de manera estática o dinámica, según lo establezca el desarrollador en el código.

El desarrollo de aplicaciones con Angular está basado en componentes, en el que cada vista está implementada por un componente. Además, los componentes son reutilizables y pueden ser utilizados por otras vistas. Cada componente se encarga de mostrar y renderizar la vista correspondiente, implementada mediante un fichero HTML y su hoja de estilos.

Otra de las características que Angular ofrece es la posibilidad de realizar el desarrollo de la aplicación con TypeScript [42], un superconjunto del lenguaje de programación JavaScript que implementa conceptos de la programación orientada a objetos, como las clases e interfaces, además de un tipado estático, lo que reduce la complejidad de desarrollo. De hecho, en este proyecto se ha hecho uso de TypeScript para el desarrollo del front-end de la aplicación.

Los componentes Angular poseen un ciclo de vida que va desde cuando Angular crea una instancia del componente y renderiza su vista, hasta cuando este componente es destruido, pasando por la detección de cambios, en la que Angular comprueba si los datos a los que se vincula la vista han cambiado. El propio framework ofrece la posibilidad de introducir acciones a ejecutar en cada momento del ciclo de vida del componente mediante la implementación de la interfaz correspondiente a cada fase del ciclo de vida. En la Figura 3.1 se presenta un esquema del ciclo de vida de un componente. En ella se pueden observar las 8 fases por las que pasa un componente (sin contar la instanciación mediante el constructor). Después de llamar al constructor del componente, Angular ejecuta el código asociado a las fases del ciclo de vida en el orden en el que aparecen en la imagen.

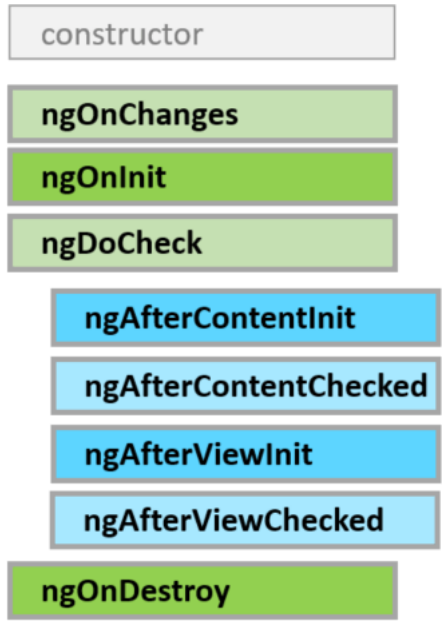

Figura 3.1: Ciclo de vida de un componente Angular

3.2.6. IntelliJ IDEA

IntelliJ IDEA [34] es un entorno de desarrollo integrado (IDE) desarrollado por JetBrains que soporta múltiples lenguajes de programación y que ofrece múltiples características, como soporte para el control de versiones desde el propio IDE o la aplicación de tareas comunes de refactoring de manera automática. Además, este IDE es extensible mediante la instalación de plugins, lo que lo convierte en una herramienta muy versátil. Este IDE ha sido utilizado para la modificación del back-end cuando ha sido necesario, pues es el IDE que se utilizó en el proyecto anterior para desarrollar dicho código de servidor.

3.2.7. Spring Boot

Spring Boot [61] es una versión simplificada del framework Spring, utilizado para crear aplicaciones del lado del servidor en Java, Kotlin y Groovy. La principal ventaja de Spring Boot es que permite crear aplicaciones Spring sin necesidad de realizar todas las configuraciones que habría que realizar si solo se utilizase Spring. El back-end de la aplicación fue desarrollado con Spring Boot y Java, por lo que en este proyecto se utilizará a la hora de introducir las modificaciones necesarias en el back-end de partida.

3.2.8. Tomcat

Apache Tomcat [10] es un contenedor de servlets desarrollado con Java que puede utilizarse como servidor web. En este proyecto, al igual que en el proyecto de partida, se utiliza a modo de servidor para el despliegue del back-end SpringBoot.

3.2.9. Flask

Flask [11] es un framework minimalista escrito en Python que permite crear aplicaciones web de manera rápida y sencilla. En este proyecto se ha utilizado para la creación de un servicio externo de recomendación para los jugadores desarrollado por Adrián Manzano,como parte de su Trabajo de Fin de Máster del Máster en Ingeniería Informática de la UVa.

3.2.10. MySQL y MongoDB

MySQL [51] es un gestor de bases de datos relacionales. MongoDB [44] es un gestor de bases de datos NoSQL orientado a documentos, que en lugar de guardar los datos en tablas como en las bases de datos relacionales, ésta lo hace en formato BSON, muy similar a JSON. MongoDB se organiza en torno a colecciones. En este proyecto se crean dos bases de datos, una con cada uno de los dos gestores de bases de datos utilizados. La base de datos gestionada con MongoDB se utiliza para guardar los datos de la simulación y otros datos necesarios para el servicio externo de recomendación, mientras que la base de datos gestionada con MySQL guardará el resto de datos relativos al dominio de los usuarios y el juego.

3.2.11. Apache HTTP Server

Apache HTTP Server [29] es un servidor web gratuito, multiplataforma y de código abierto, cuyo uso está muy extendido actualmente. En este proyecto se utiliza para desplegar el front-end de Crossroads.

3.2.12. Docker

Docker [23] es un proyecto de código abierto que automatiza el despliegue de aplicaciones mediante el uso de contenedores software. Dichos contenedores almacenarán todo lo que necesita la aplicación para ser desplegada, por lo que Docker ofrece un mecanismo de despliegue independiente de la plataforma que ejecute dichos contenedores. El servicio externo desarrollado por Adrián Manzano utiliza Docker para ser desplegado.

Capítulo 4

Análisis

En este capítulo se presentan los resultados del análisis realizado para identificar los conceptos relevantes del dominio del problema. Para realizar este proceso de análisis se han elaborado una serie de diagramas UML, cuyo objetivo es ilustrar de manera clara y precisa tanto las entidades que conforman el dominio del problema, como el modelado de los estados por los que pasan los usuarios con distintos roles durante el uso de la aplicación. Como este Trabajo de Fin de Grado parte de un proyecto anterior, buena parte del trabajo de análisis ya fue realizado, aunque en este capítulo se presenta una evolución de los diagramas realizados anteriormente, indicando debidamente en cada punto las novedades y actualizaciones aplicadas en dichos diagramas.

4.1. Modelo de dominio

En la Figura 4.1 se presenta un diagrama de clases con todas las entidades del dominio. En él, se muestran todas las entidades que se han identificado a través del código del back-end. Entre las principales diferencias con el diagrama de clases original, cabe destacar la adición de una nueva enumeración (clase **EstadoGrupo**) que representa el estado en el que está se encuentra un grupo, dependiendo de los estados individuales de sus jugadores. También se ha modificado el nombre de algunos atributos en la clase asociativa **ResultadoEnRonda**, debido a que, por un cambio en los requisitos, el sistema de puntuación ha cambiado y ahora se tienen en cuenta otros parámetros para el cómputo de la puntuación. Ambos cambios aparecen en color rojo en el diagrama de clases.

El otro gran cambio que ha sufrido el modelo de dominio ha sido la eliminación de las clases **Acción** y **TipoAcción**. Esta eliminación se ha realizado puesto que no se ha detectado soporte para estas clases en el código del back-end de Crossroads.

La clase **Jugador** pasa por diferentes estados durante el transcurso de una partida, es por esto, que se ha elaborado un diagrama de máquinas de estados que refleja esta condición (Figura 4.2).

En este diagrama los cambios han sido más de reorganización de las dependencias entre algunos estados, por ejemplo, el estado **pendienteEnvio** ahora precede a la comprobación de la existencia de conflictos, mientras que el jugador entra en estado **revisandoPreguntas** cuando revisa la respuesta que ha enviado en una pregunta (sin implicar necesariamente la resolución del conflicto en esa pregunta). Por otro lado, el jugador entra en estado **viendoResultadosRonda** cuando deja de haber conflictos entre las respuestas de los distintos jugadores del grupo. Para finalizar hay que destacar que se han añadido como subestados del estado **en partida** a los estados **viendoResultadosRonda** y **viendoResultadosFinales**.

En relación a los estados por los que pasa un grupo, se ha elaborado una tabla en la que se recojen las condiciones que deben cumplir los jugadores de dicho grupo para que el grupo pueda alcanzar dicho estado (Tabla 4.1).

Estado	**Descripción del estado**
contestandoPreguntas	Si, al menos uno de los jugadores está contestando preguntas, o está esperando a que otro jugador del grupo envíe sus respuestas o si está escribiendo la trama narrativa, significa que el grupo no ha terminado de enviar todas las respuestas, por lo que el estado del grupo es "Contestando preguntas"
resolviendoConflictos	Si uno de los jugadores está resolviendo conflictos, el resto de jugadores está resolviendo conflictos al menos indirectamente, por lo que este es el estado del grupo entero
resultadosDeRonda	Si uno de los jugadores está viendo los resultados de la ronda, el resto de jugadores del grupo están viendo los mismos resultados, por lo que el estado del grupo entero es "Viendo resultados de ronda".
desconectado	Si no se cumplen las condiciones de los estados anteriores, eso significa que todos los jugadores del grupo se han desconectado o han sido expulsados. Estos dos estados se agrupan en este estado

Tabla 4.1: Estados de un grupo durante una ronda

4.2. Modelado de la interacción

La interacción con el sistema de los usuarios con rol "Moderador" y "Jugador" también pueden modelarse como máquinas de estados, en el que cada estado podría asociarse con cada una de las pantallas por las que pasaría dicho usuario al utilizar el sistema y las transiciones entre estados serían los eventos de usuario o de sistema que provocan los cambios de pantalla en el sistema. En las Figuras 4.3, 4.4 y 4.5 se presentan las máquinas de estados que modelan la interacción con el sistema de ambos roles de usuario.

Respecto del proyecto de partida, en el diagrama de la Figura 4.3, se han simplificado los subestados del estado **Sala de control**, pues las acciones que se indicaban dentro de dicho

estado en la versión anterior de los diagramas no eran más que una forma de monitorización de la partida. También se han fusionado los estados **Finalizar partida** e **imprimir resultados**, pues esa es una funcionalidad que puede realizar el moderador una vez haya finalizado la partida. Además, se han añadido los estados **Buscar sala** y **Revisión partida**, que reflejan la posibilidad que tiene el moderador de poder revisar las salas que ha creado y poder reconectarse a una sala que esté activa.

En segundo lugar, el diagrama del la Figura 4.4 se ha mantenido sin grandes cambios, tan solo se ha añadido un estado en el cual el jugador tiene que introducir el código de la sala para unirse a la partida.

Finalmente, el diagrama de máquina de estados del jugador durante la ronda ha sido modificado para reflejar los cambios en la máquina de estados de la clase **Jugador**, principalmente en la parte de resolución de conflictos.

4.3. Modelo de procesos de negocio

En la Figura 4.6 se presenta el modelado de la interacción entre el usuario "Moderador" y el usuario "Jugador" durante el transcurso de una partida. Hay que indicar que este diagrama representa una secuencia principal del desarrollo de una partida, en la que no se representa, por ejemplo, el caso en el que el moderador interrumpe el transcurso habitual de una ronda y la finaliza antes de que los jugadores hayan resuelto todos los conflictos.

Este diagrama no tiene un predecesor en el proyecto de partida, pero se ha añadido debido a la necesidad de modelar la sincronía y asincronía de ambos roles mediante el modelado de los procesos de negocio.

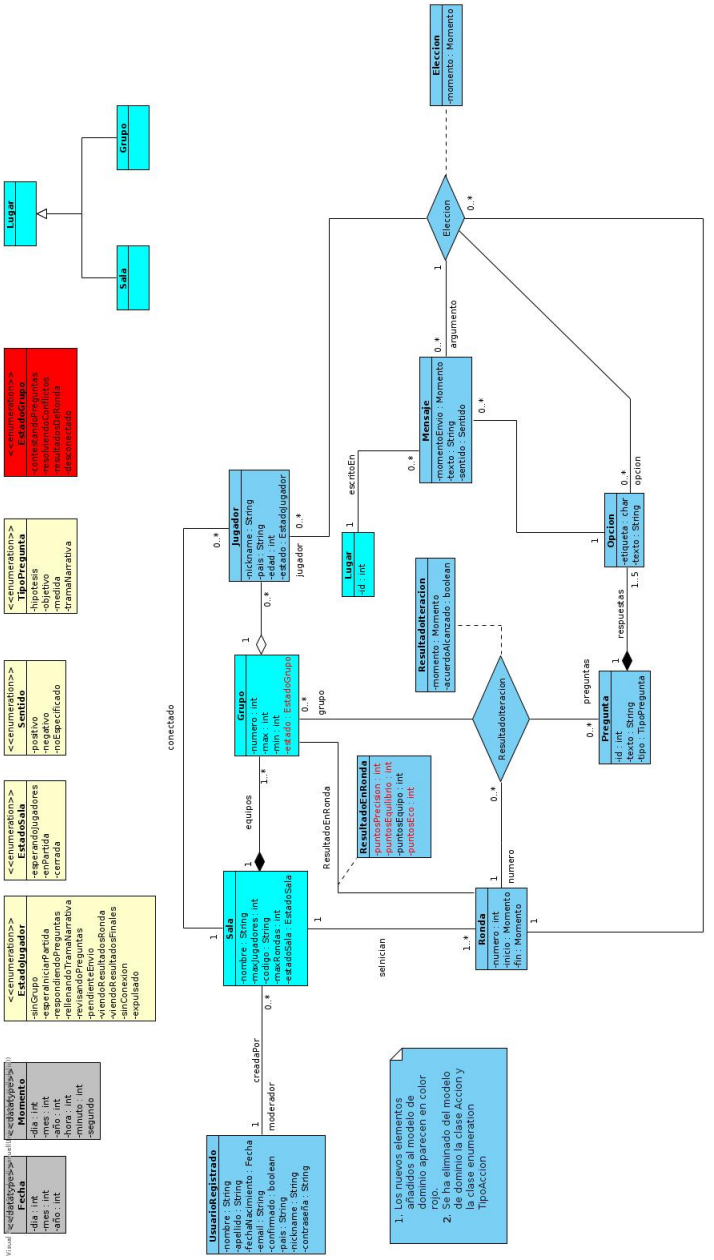

Figura 4.1: Diagrama de clases con las clases del dominio

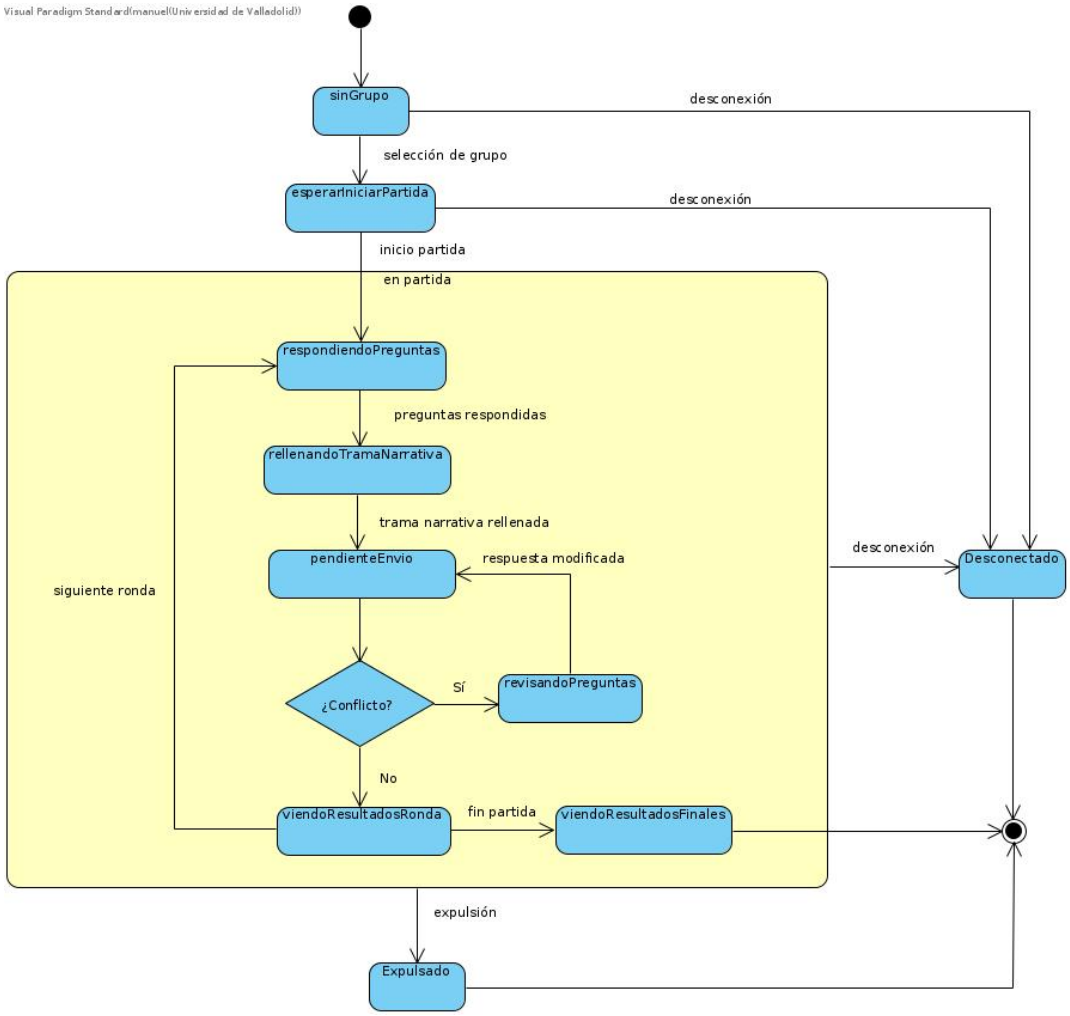

Figura 4.2: Diagrama de máquina de estados de la clase Jugador

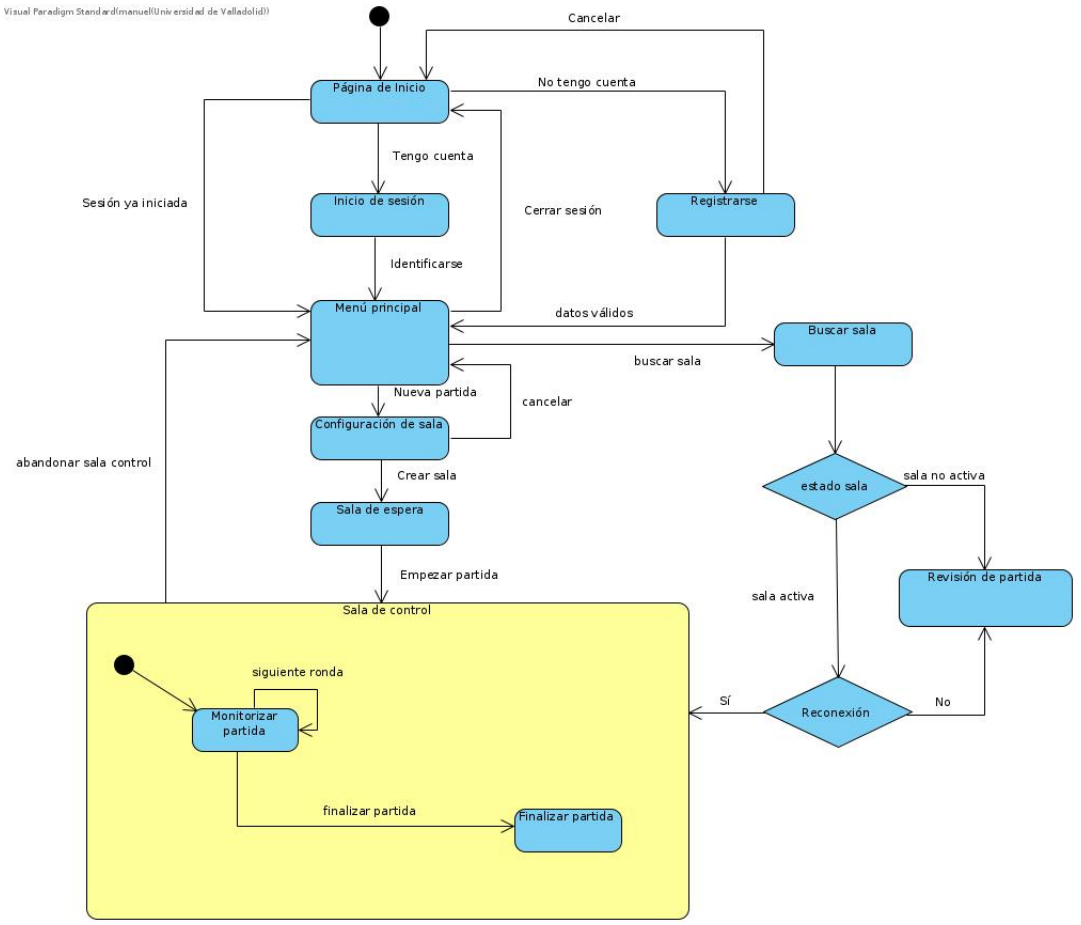

Figura 4.3: Diagrama de máquina de estados de un usuario con rol “Moderador”

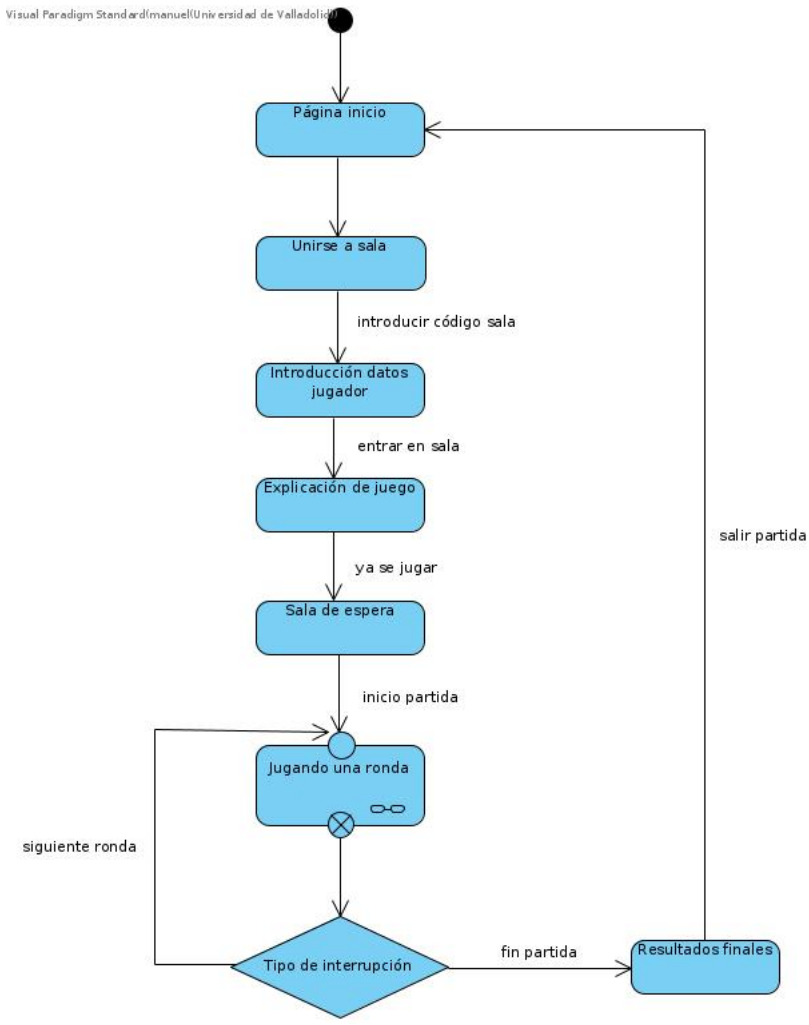

Figura 4.4: Diagrama de máquina de estados de un usuario con rol “Jugador”

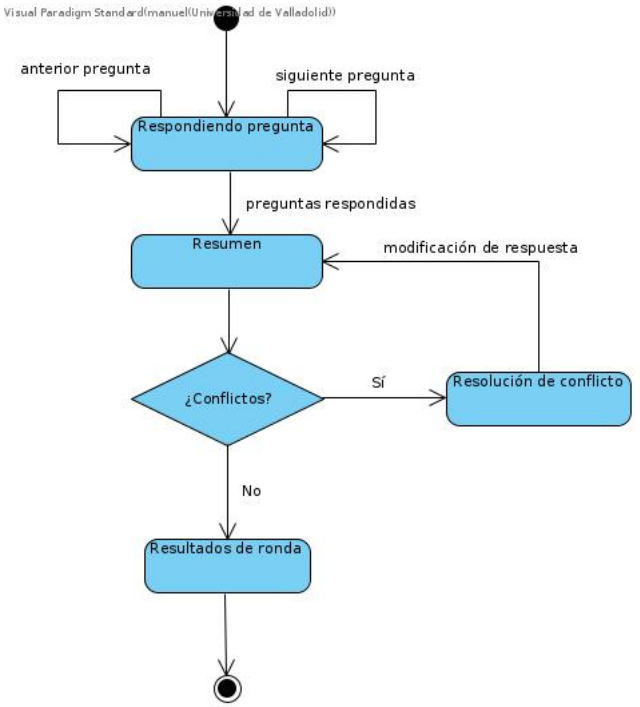

Figura 4.5: Subdiagrama de máquina de estados de un usuario con rol “Jugador” durante el transcurso de una ronda

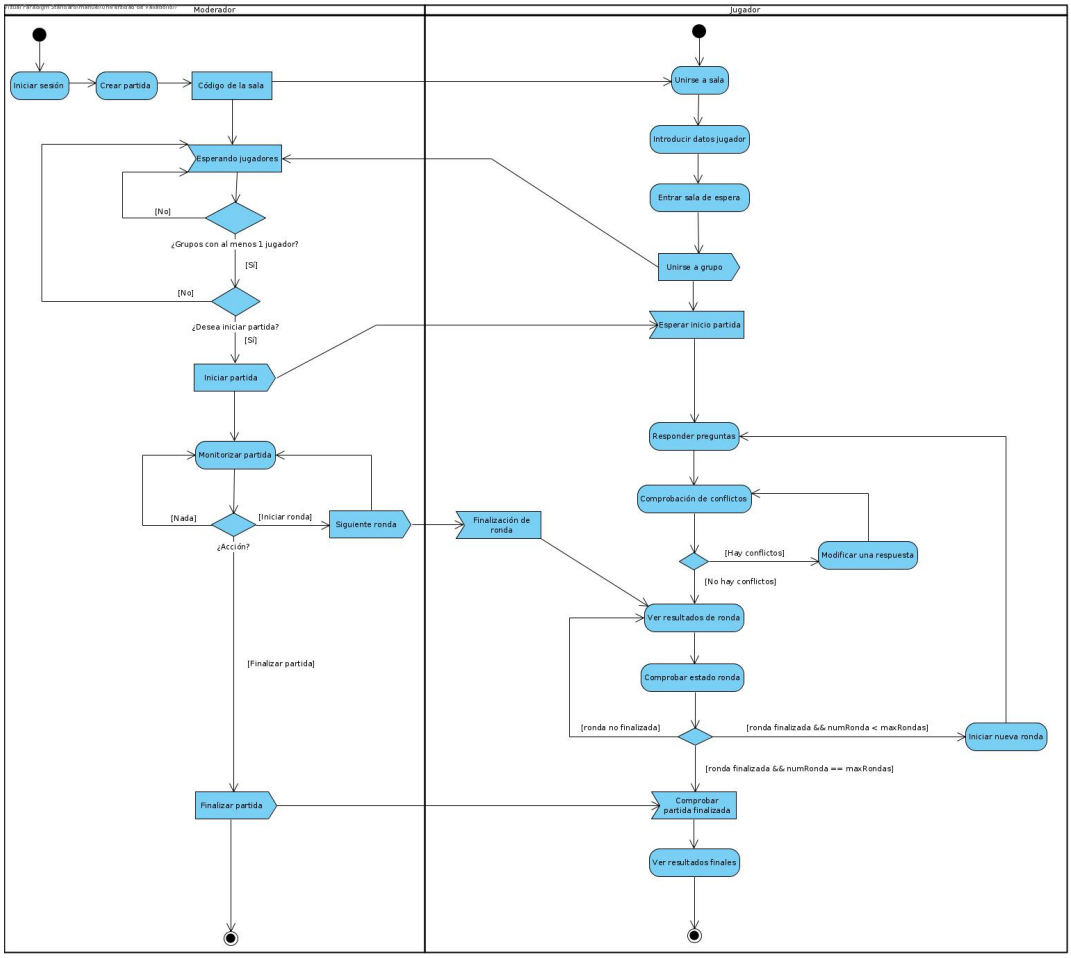

Figura 4.6: Diagrama de actividades con el modelado del proceso de negocio en el que participan juntos los usuarios con rol “Moderador” y “Jugador”

Capítulo 5

Diseño

Una vez realizado el proceso de análisis, es momento de realizar el diseño de la aplicación final. Durante este proceso se definirá la arquitectura del sistema, así como el modelo de datos subyacente que se utilizará en la base de datos para guardar la información de las entidades del modelo de dominio. Además, se elaborarán los bocetos de la interfaz gráfica y se establecerán los patrones de diseño que hay que aplicar en la construcción de la aplicación final. Para terminar, se realizará un modelo de despliegue que ilustrará la forma en la quedarán distribuidos los diferentes subsistemas de la aplicación en producción.

En este capítulo se presentan los resultados del proceso de diseño realizado para el front-end. El diseño realizado para el back-end se encuentra en el TFG de partida [37].

5.1. MVVM

Modelo-Vista-Modelo de vista (*Model-View-ViewModel* o MVVM [62]) es un patrón arquitectónico que busca la separación entre el código de la interfaz de usuario y el de la lógica de negocio. Este patrón define tres componentes principales:

- **Modelo.** Componente independiente de los demás. Contiene la lógica del negocio y el estado de la aplicación. Este componente no contiene ninguna referencia a ninguno de los otros dos componentes, es decir, no "conoce" a ninguno de ellos.
- **Modelo de vista.** Componente que actúa de intermediario entre el modelo y la vista. Se encarga de recibir las peticiones realizadas en la interfaz y actualizar el modelo de forma adecuada. Este componente "conoce" al **Modelo**, pero no a la **Vista**.
- **Vista**: Componente que contiene todos los elementos visuales que se mostrarán al usuario de la aplicación. Este componente "conoce" al **Modelo de vista**.

En la Figura 5.1 se puede observar el esquema de organización que siguen los componentes que define este patrón arquitectónico. Aunque el **Modelo** no tenga una referencia directa al **Modelo de vista**, este último puede detectar los cambios en el **Modelo** mediante el uso del patrón Observador [31]. De igual manera ocurre con el **Modelo de vista** y la **Vista**, esta vez mediante el Data binding [46].

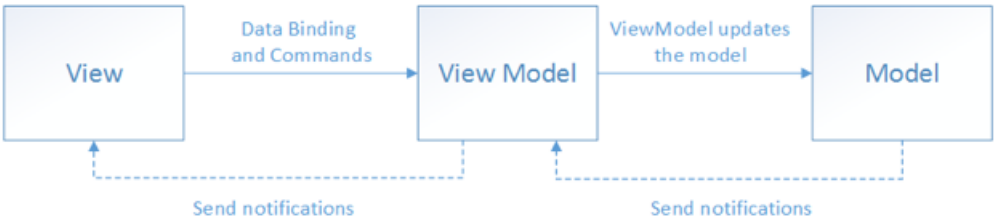

Figura 5.1: Esquema del patrón MVVM [40]

5.1.1. MVVM en Angular

Aunque el framework Angular no realice una implementación al uso de este patrón, sí que se pueden identificar los distintos elementos de este patrón en la estructura de componentes de Angular. A continuación, se presenta en detalle la correspondencia de los distintos elementos que define Angular para sus componentes con cada una de los elementos de este patrón [47]:

- **Vista.** Este elemento se corresponde con el fichero plantilla (.html) del componente.
- **Modelo de Vista.** Este elemento se corresponde con la clase TypeScript que recibe los eventos de la interfaz.
- **Modelo.** Este elemento encuentra correspondencia en multitud de elementos del framework, desde las clases de dominio, hasta los servicios inyectables mediante inyección de dependencias.

5.2. Desarrollo basado en componentes

El desarrollo de software basado en componentes consiste en la reutilización de "piezas" de software para la construcción de un nuevo sistema. De esta forma, se reduce la complejidad, se simplifica el proceso de testing (pues buena parte de estas se realizan para cada componente de manera individual) y mejora el mantenimiento del sistema resultado, pues los componentes se desarrollan bajo la premisa de lograr un bajo acoplamiento entre ellos, lo que facilita la introducción de cambios en el sistema.

5.2.1. Componentes angular

Angular implementa un modelo de desarrollo orientado a componentes. La estructura básica de un componente Angular se encuentra en la Figura 5.2. Como se puede observar, este componente está formado por una clase TypeScript y una plantilla HTML.

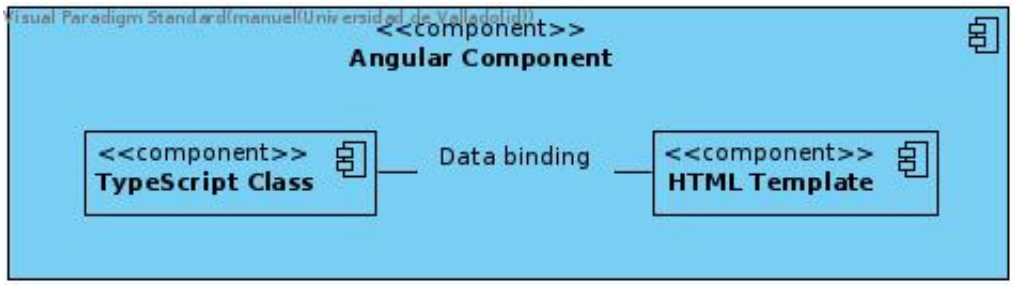

Figura 5.2: Estructura de un componente Angular básico

Los componentes Angular también pueden reutilizarse dentro de otros componentes, estableciéndose una jerarquía padre-hijo. En la Figura 5.3 se presenta la estructura de un componente que hace uso de un componente hijo. Dicho componente hijo se inserta en la plantilla HTML del componente Angular.

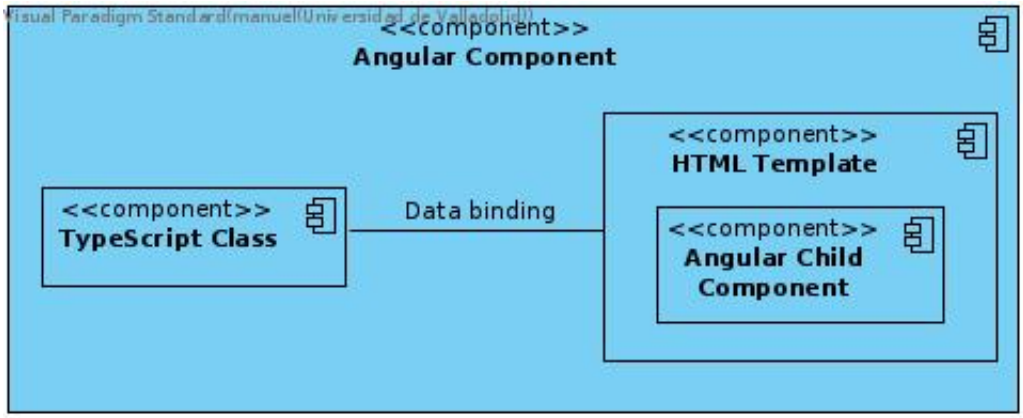

Figura 5.3: Estructura de un componente Angular padre que contiene un componente Angular hijo

Finalmente, los componentes pueden requerir de uno o más servicios para poder realizar su función. Dichos servicios se modelan como una interfaz requerida por el componente (Figura 5.4).

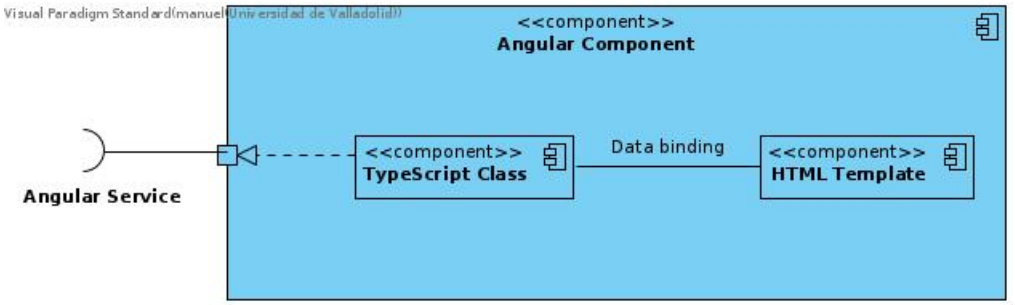

Figura 5.4: Estructura de un componente Angular que requiere de un servicio para funcionar

5.2.2. Componentes creados para el front-end

A continuación, se enumeran todos los componentes creados para el front-end, indicando en cada caso la funcionalidad principal que implementa. En la Figura 5.5 se muestra un diagrama con todos los componentes del sistema y las dependencias existentes entre todos ellos.

- **App:** contiene la aplicación completa.
- **Home:** página de bienvenida de la aplicación.
- **Register:** página de registro de usuarios. Los usuarios registrados podrán ser moderadores de una partida.
- **LoginChair:** página de inicio de sesión para los usuarios que tienen cuenta.
- **Menu:** menú principal para los usuarios registrados. Presenta las tareas que puede realizar el usuario registrado.
- **SelectRoom:** página de búsqueda de las salas que ha creado el usuario. Permite reconectarse a las salas que se encuentran activas y el acceso a los datos de la partida en las partidas no activas.
- **ProfileChair:** página con los datos del perfil del usuario registrado.
- **EditProfileChair:** modificación de los datos del perfil del usuario registrado.
- **CreateRoom:** componente encargado de la creación de una sala de partida.
- **WaitingChair:** sala de espera a través de la cual, el moderador controla los jugadores que se van uniendo a la partida y los jugadores que entran en cada grupo.
- **Dashboard:** pantalla de monitorización de la partida. Permite al moderador comprobar el estado de un grupo en un momento dado, o la monitorización de todos los grupos durante una ronda, así como la finalización de la partida o el inicio de una nueva ronda, entre otros.

- **LoginPlayer:** página a través de la cual, el jugador se une a una sala a partir del código de la sala.
- **ProfileDataPlayer:** pantalla en la que el jugador introduce los datos identificativos que utilizará durante toda la partida (nick, país y edad).
- **InfoPlayer:** componente que muestra información acerca del juego y del proyecto a los jugadores. Controla el acceso a la sala de espera de los jugadores.
- **WaitingPlayer:** sala de espera para los jugadores. El jugador tendrá que seleccionar un grupo y esperar para a que el moderador inicie la partida.
- **Quest:** página que muestra el formulario de preguntas que el jugador tiene que responder para elaborar la propuesta del grupo. Hace uso del componente **Chat**.
- **ConflictCheck:** componente encargado de la lógica de mostrar el estado actual de la propuesta que está elaborando el grupo en un momento dado de la ronda. También muestra informacióna acerca del estado en el que se encuentran los jugadores del grupo en ese preciso instante. Hace uso del componente **Chat**.
- **ConflictSolve:** componente que se encarga de la lógica de resolución de un conflicto para una pregunta determinada en la que los jugadores no han llegado a consenso. Muestra información acerca de que jugadores han elegido cada respuesta y el número de argumentos a favor y en contra que han recibido todas las respuestas de esa pregunta. Hace uso del componente **Chat**.
- **ResultRound:** página que muestra los resultados de la simulación en formato de gráficas. Además, muestra los resultados del jugador en la ronda, de acuerdo con las puntuaciones establecidas durante el análisis (Capítulo 4). Para finalizar, muestra a los integrantes del grupo un mensaje a modo de recomendación para la siguiente ronda. Hace uso del componente **Chat**.
- **ResultFinal:** pantalla que muestra los resultados finales de la partida de manera similar a la propuesta para el componente **ResultRound**. Se ha establecido que las gráficas de la simulación que se muestren sean las de la ronda en la que el grupo alcanzó el mayor valor en la puntuación **equilibrio**. En el apartado de puntuaciones, se muestra un podio con los tres mejores grupos en cada uno de los baremos utilizados. También se muestran las gráficas del mejor grupo con respecto a la puntuación **equilibrio**. Este componente ofrece un mensaje de recomendación para mejorar los resultados obtenidos por el mejor grupo.
- **Chat:** componente encargado de llevar el debate entre los jugadores. Permite el envío de mensajes a favor, en contra o neutros para una respuesta de una pregunta dada. Implementa dos tipos de chat, uno grupal y uno para todos los jugadores de la sala.

El detalle de las clases utilizadas para cada componente se encuentra en los diagramas de las Figuras 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 y 5.12. En todos estos diagramas se muestran las clases TypeScript de cada componente, cada una con sus atributos y métodos. Además, cada clase declara una serie de dependencias a inyectar, añadiendo dichas dependencias al constructor de la clase TypeScript. Este mecanismo de inyección de dependencias permite controlar el

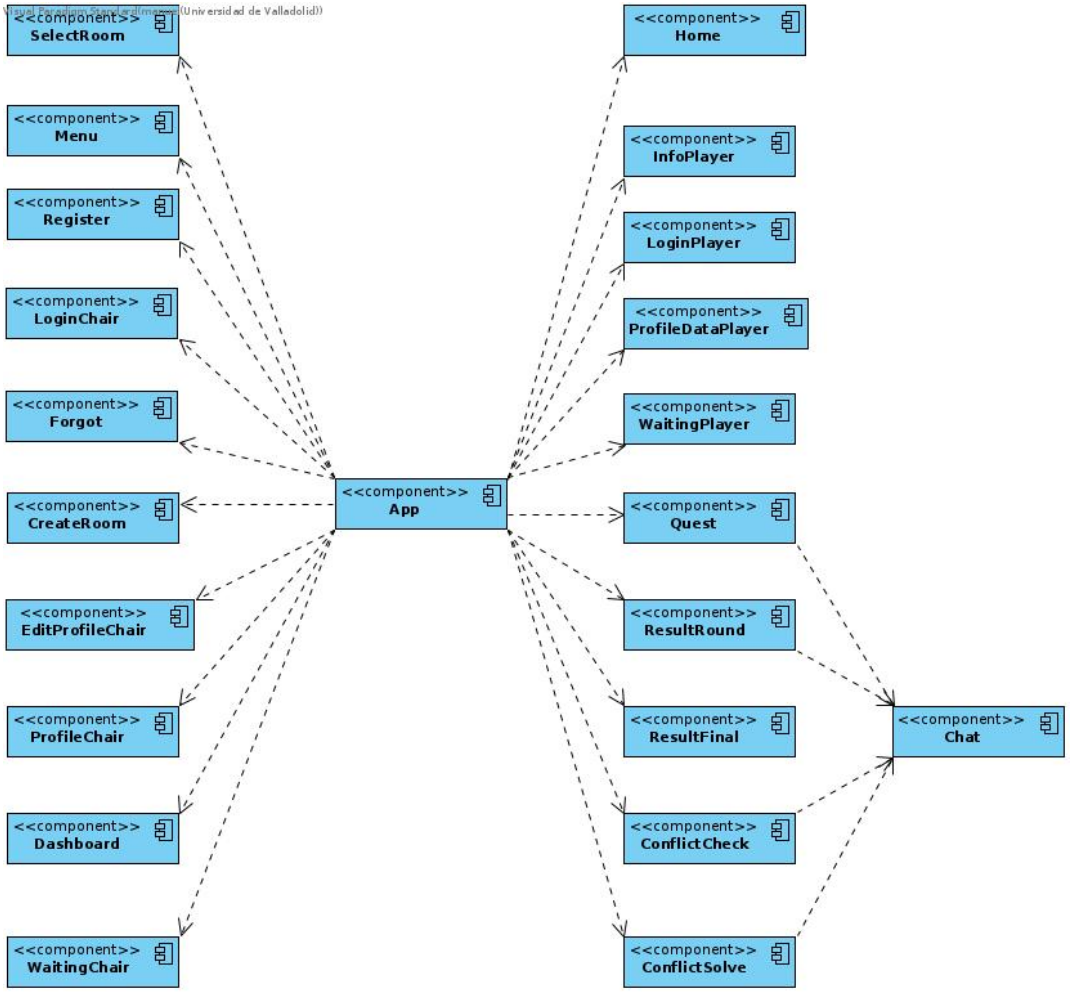

Figura 5.5: Estructura de componentes del front-end

acceso a las instancias de los servicios y componentes, garantizando el acceso a una única instancia de los mismos [52]. La mayor parte de estas dependencias son servicios creados para la aplicación. Dichos servicios se presentan en los diagramas de las Figuras 5.13 y 5.14. En estos diagramas se presentan las dependencias entre algunos de estos servicios. Estas dependencias también se resuelven mediante el mecanismo de inyección de dependencias que ofrece Angular a través del constructor de la clase.

5.2.3. Aplicación del patrón Observador

Para finalizar la exposición acerca de los componentes Angular, hay que hablar de la aplicación del patrón Observador en cada una de las clases de los diagramas. Esto se lleva

a cabo mediante la declaración de atributos en las clases de tipo Observable y Subject y la adición de métodos cuyo tipo de retorno es Observable. Tanto el tipo Observable como el tipo Subject permiten aplicar un mecanismo de "suscripción" sobre los objetos de dicho tipo que realizan una notificación a los "suscriptores" cuando su valor cambia.

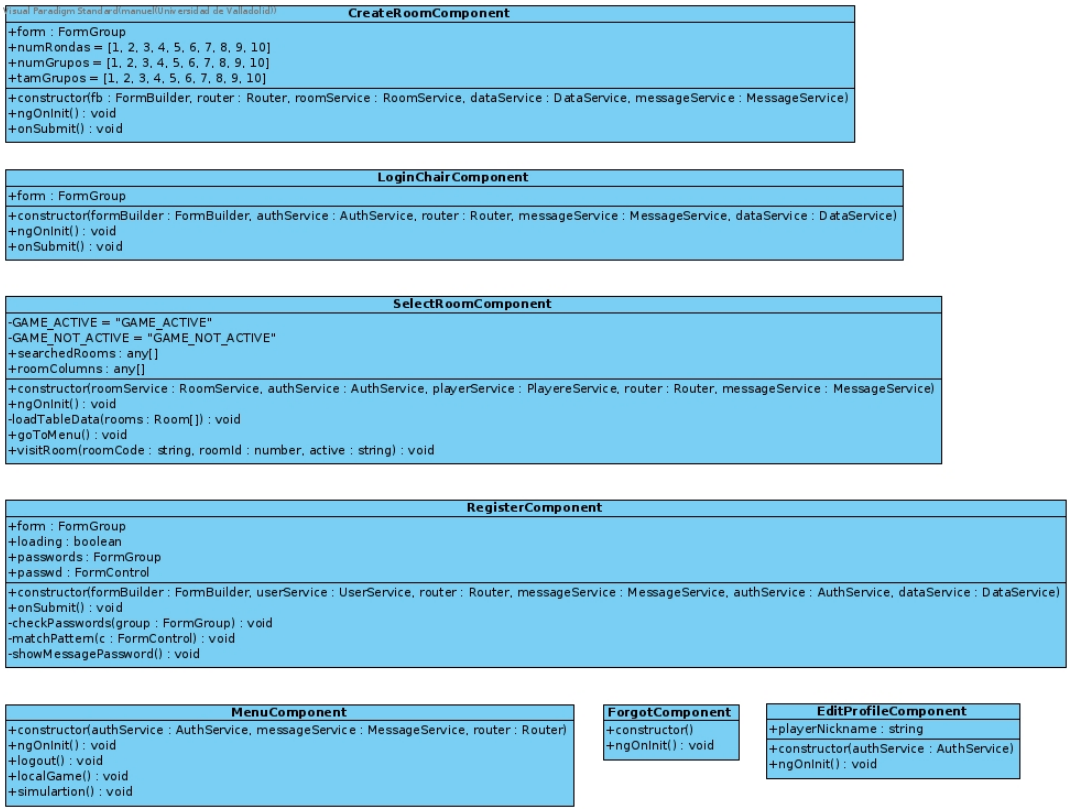

Figura 5.6: Diagrama de clases con las clases TypeScript de cada componente. Componentes del moderador (Parte 1)

5.3. Diseño de datos

5.3.1. Base de datos MySQL

El esquema de los datos almacenados en la base de datos no ha cambiado con respecto a la versión anterior, tan solo habría que indicar el cambio en el nombre de las columnas **points_perf** y **points_time** de la entity **result_in_round** por **points_precision** y **points_balance** para reflejar los cambios en el sistema de puntuación (ver modelo de dominio en el Capítulo 4). El esquema de la base de datos MySQL se presenta en la Figura 5.15, indicando los cambios con color rojo.

5.3.2. Base de datos MongoDB

El esquema de los datos que se almacenarán se encuentra en la Figura 5.16. La principal diferencia con respecto a la anterior versión de este diagrama reside en la adición de dos nuevas colecciones. Dichas colecciones almacenan datos que serán necesarios para la obtención de recomendaciones que hará el sistema a los grupos, dependiendo de las respuestas que hayan introducido. Dicha funcionalidad será provista por un servicio externo a la app desarrollado por Adrián Manzano, estudiante de TFM mencionado anteriormente, por lo que no se explicará en detalle la naturaleza de las dos colecciones añadidas.

5.4. Despliegue de la aplicación

Para realizar el despliegue de la aplicación, se ha decidido utilizar una máquina virtual que almacene tanto el front-end como el back-end de la aplicación. Además, se prevé que el servicio externo que ofrece una recomendación a los grupos se despliegue en la misma máquina virtual, con el objetivo de evitar tener que duplicar el contenido de la base de datos MongoDB en dos máquinas distintas. Dicho servicio viene desplegado en docker, por lo que es necesario incluirlo como entorno de ejecución. El resultado del diseño del despliegue se encuentra en la Figura 5.17.

5.5. Bocetos de la interfaz de usuario

En esta sección se presentan todos los bocetos de la interfaz de usuario (Figuras 5.18 a 5.36). Cada imagen viene acompañada con un mensaje ilustrativo que indica, en líneas generales, la funcionalidad de dicha pantalla y el tipo de usuario que se encontrará con dicha pantalla. El diseño final de las pantallas se presentará en el Anexo A. Dicho diseño final ha sido desarrollado por la estudiante de Diseño Industrial y Desarrollo de Producto Elena Rodríguez.

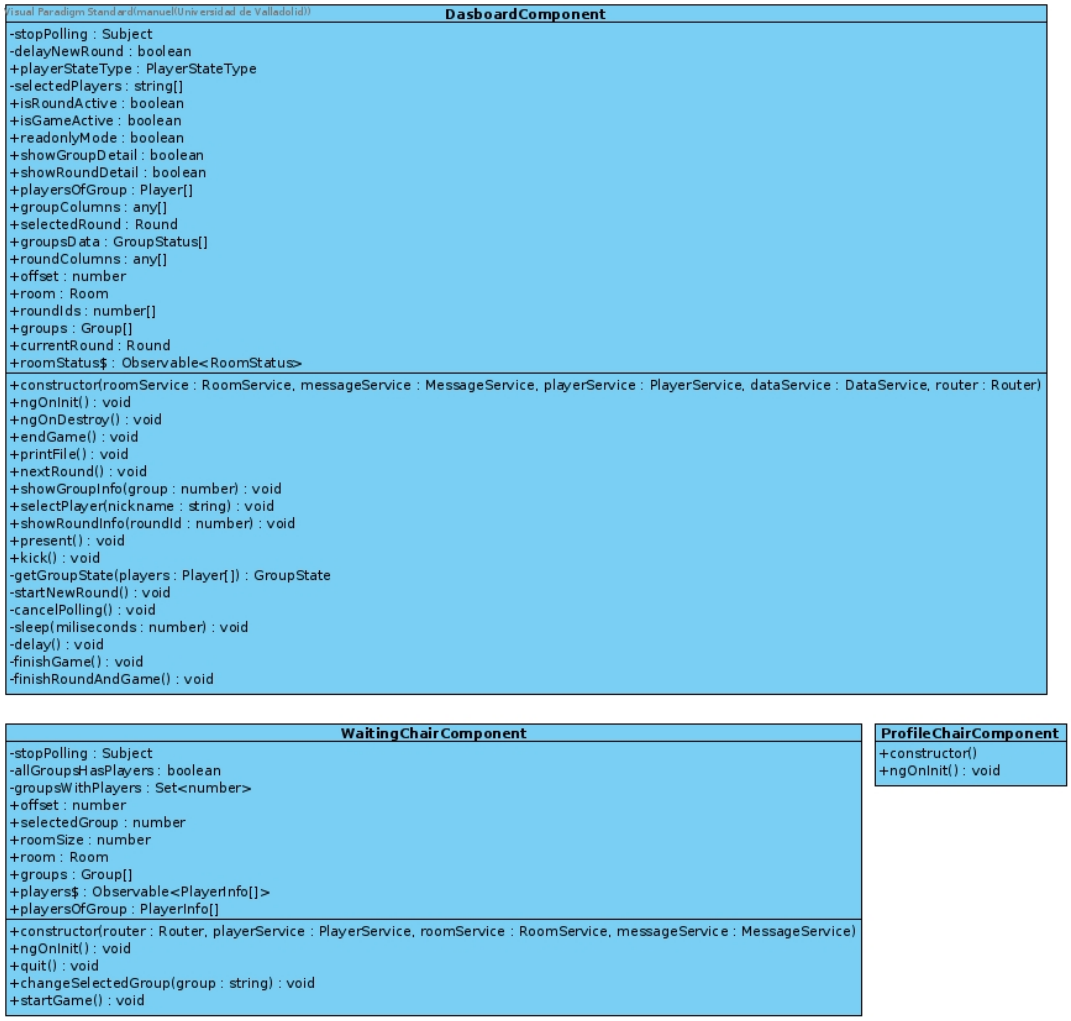

Figura 5.7: Diagrama de clases con las clases TypeScript de cada componente. Componentes del moderador (Parte 2)

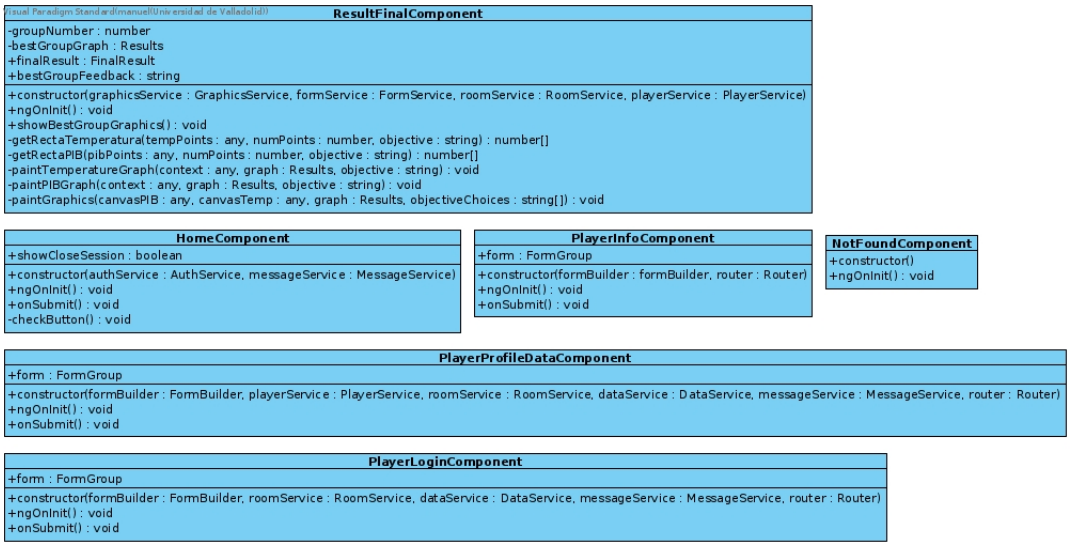

Figura 5.8: Diagrama de clases con las clases TypeScript de cada componente. Componentes comunes y componentes del jugador (Parte 1)

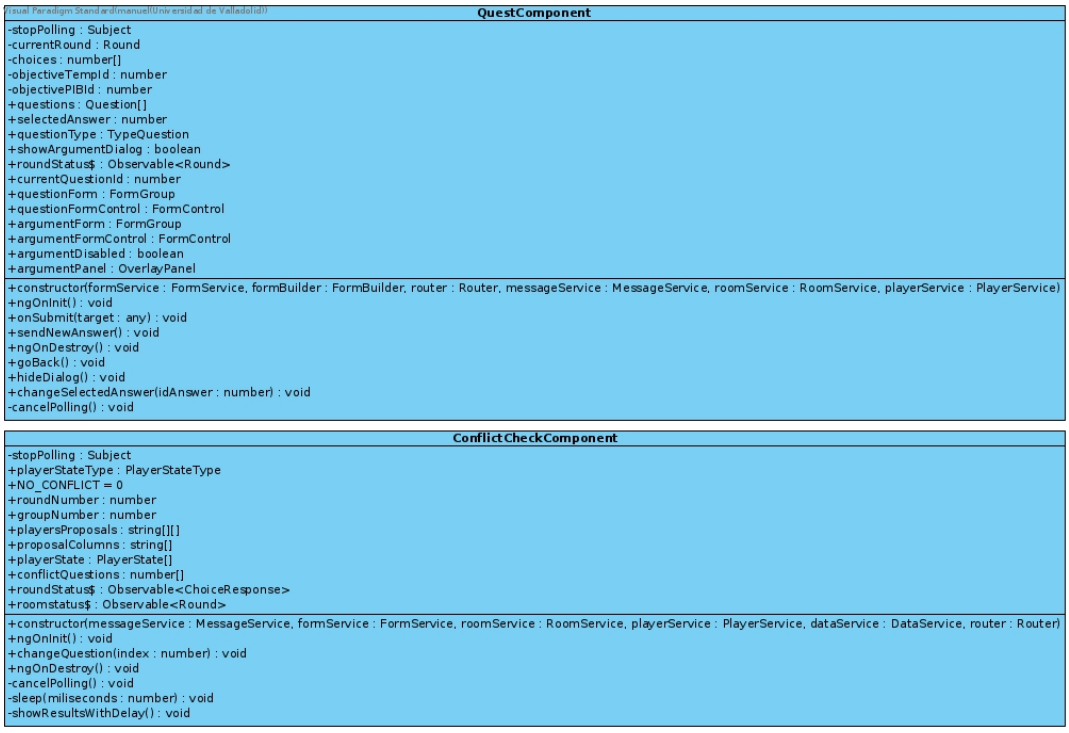

Figura 5.9: Diagrama de clases con las clases TypeScript de cada componente. Componentes comunes y componentes del jugador (Parte 2)

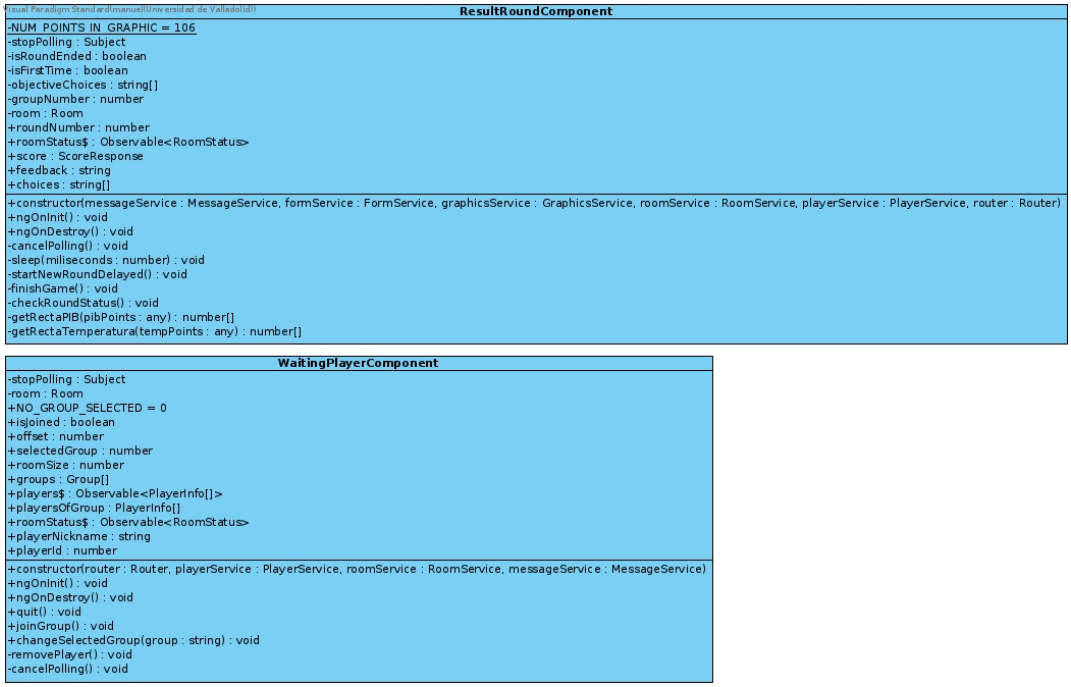

Figura 5.10: Diagrama de clases con las clases TypeScript de cada componente. Componentes comunes y componentes del jugador (Parte 3)

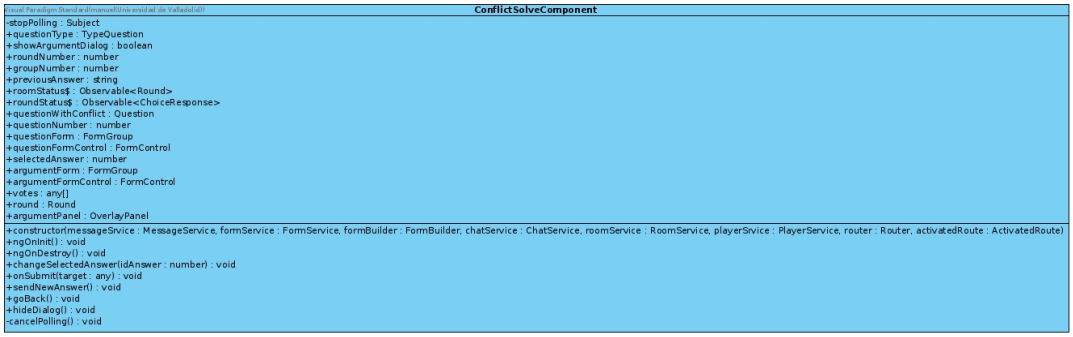

Figura 5.11: Diagrama de clases con las clases TypeScript de cada componente. Componentes comunes y componentes del jugador (Parte 4)

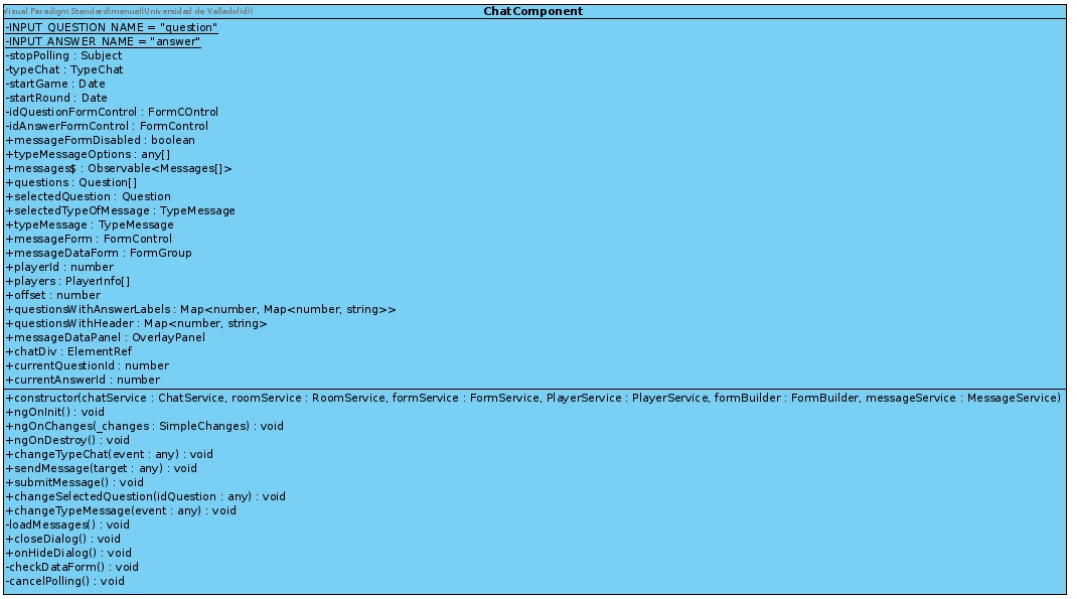

Figura 5.12: Diagrama de clases con las clases TypeScript de cada componente. Componentes comunes y componentes del jugador (Parte 5)

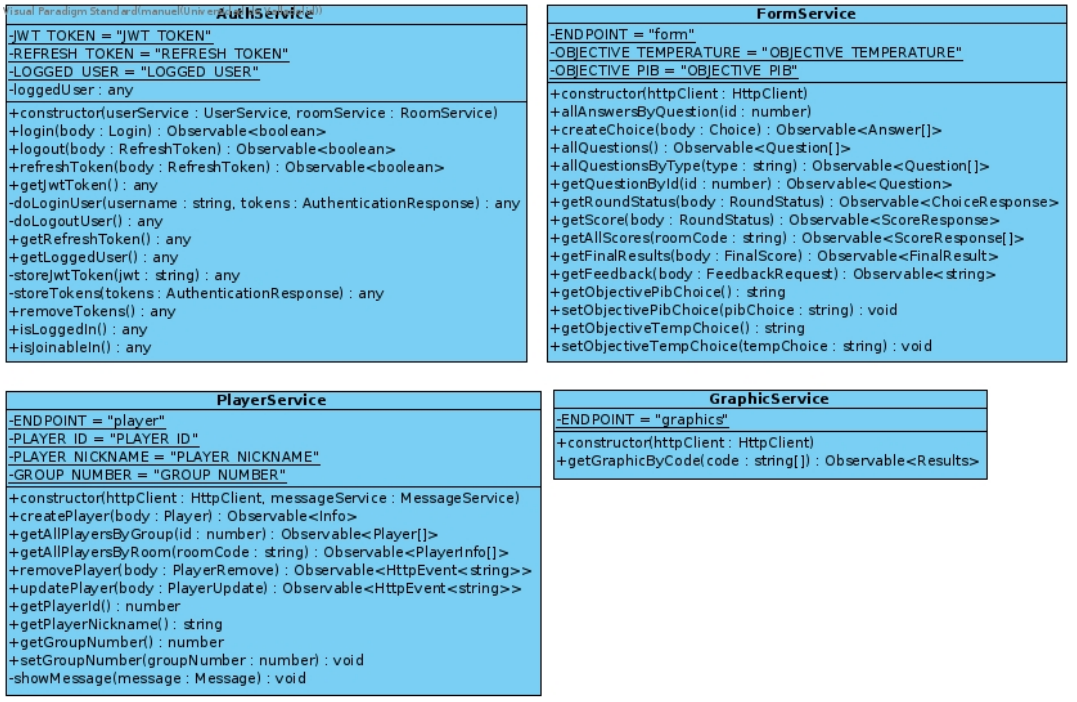

Figura 5.13: Diagrama de clases con los servicios que utilizarán los componentes (Parte 1)

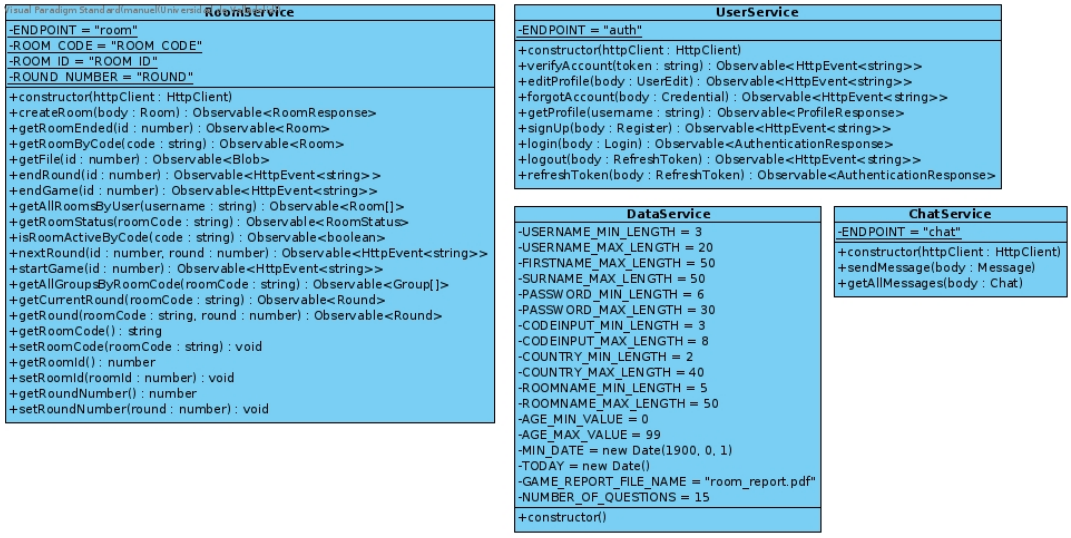

Figura 5.14: Diagrama de clases con los servicios que utilizarán los componentes (Parte 2)

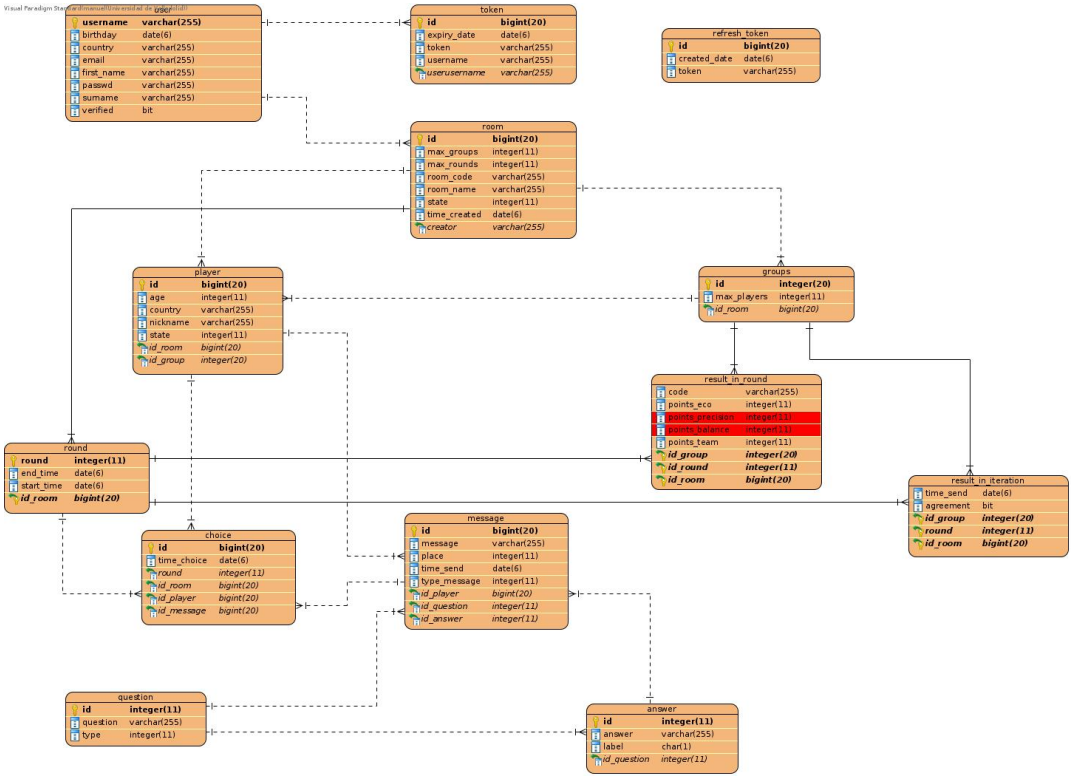

Figura 5.15: Base de datos MySQL

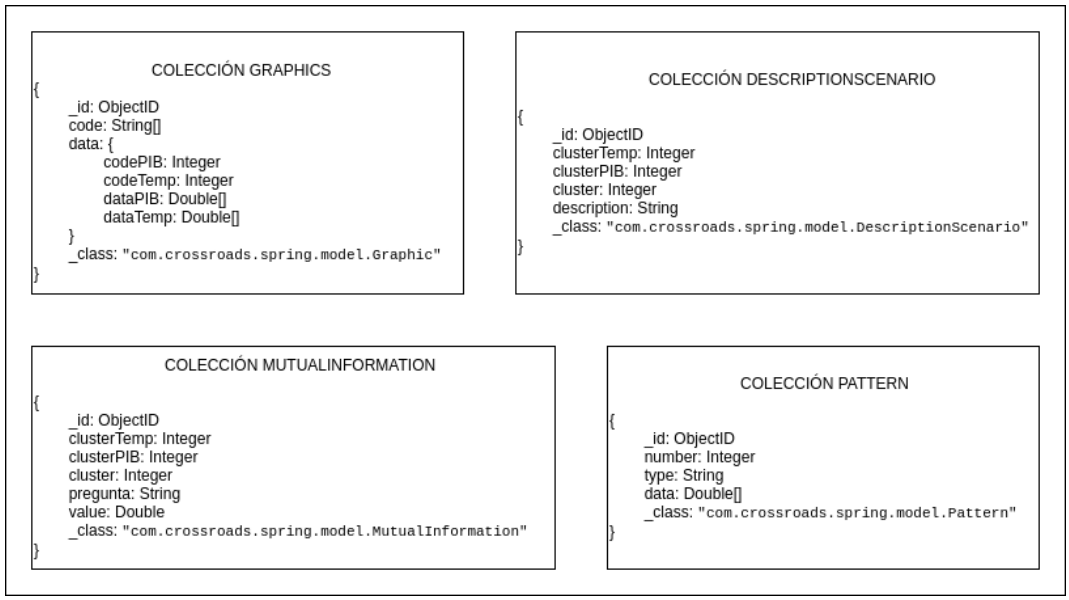

Figura 5.16: Colecciones de la base de datos Mongo

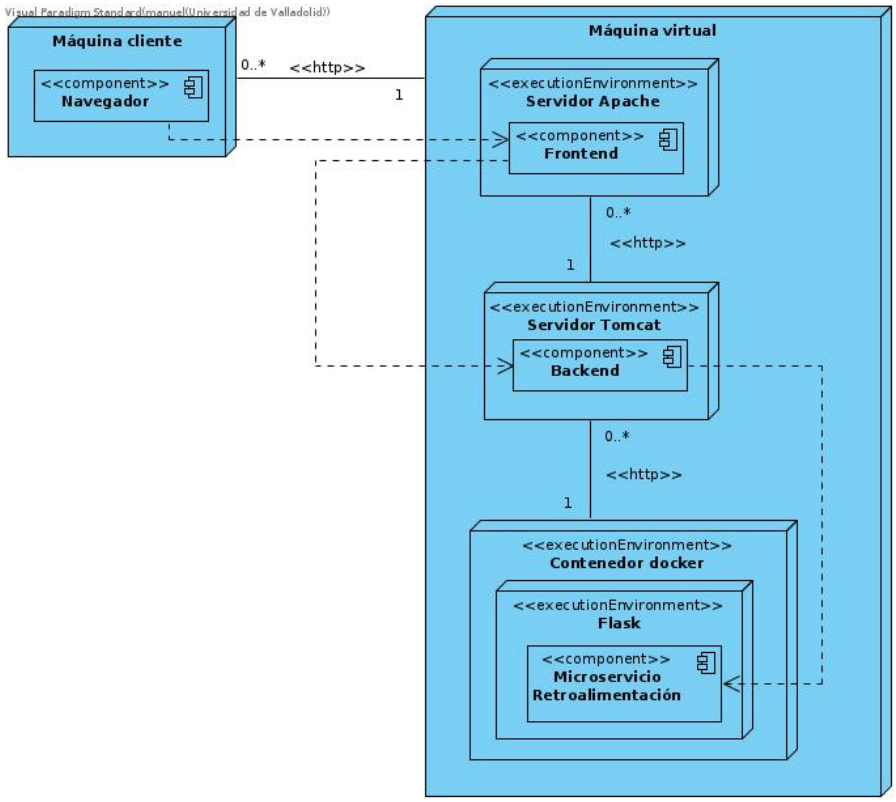

Figura 5.17: Diagrama de despliegue de la aplicación

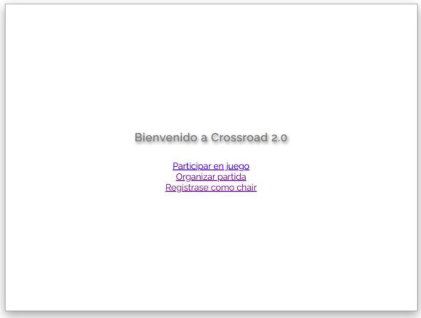

Figura 5.18: Boceto de la pantalla principal de la app

Figura 5.19: Boceto de la pantalla de registro

Figura 5.20: Boceto de la pantalla de inicio de sesión

Figura 5.21: Boceto de la pantalla de recuperación de cuenta

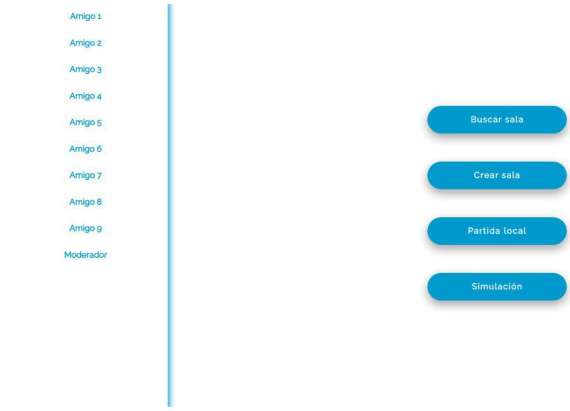

Figura 5.22: Boceto de la pantalla de menú para los usuarios que han iniciado sesión

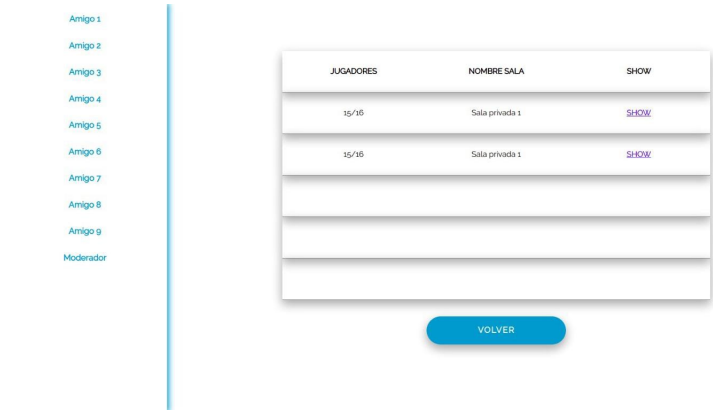

Figura 5.23: Boceto de la pantalla de búsqueda de las salas que ha creado un usuario

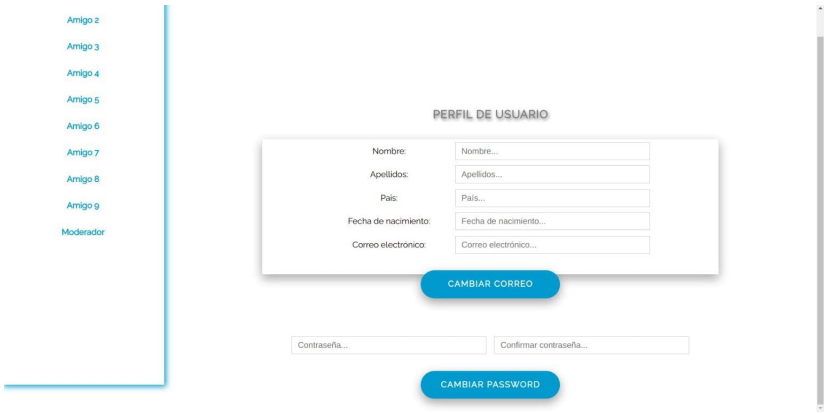

Figura 5.24: Boceto de la pantalla de edición de los datos del perfil

Figura 5.25: Boceto de la pantalla de crear sala

Figura 5.26: Boceto de la pantalla de sala de espera del moderador de la partida

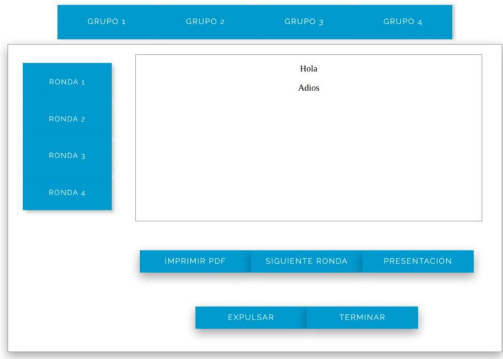

Figura 5.27: Boceto de la pantalla de monitorización de partida para el moderador

Figura 5.28: Boceto de la pantalla en la que el jugador se une a una sala a partir del código

Figura 5.29: Boceto de la pantalla en la que el jugador introduce sus datos para la partida

Figura 5.30: Boceto de la pantalla de información para los jugadores

Figura 5.31: Boceto de la pantalla de sala de espera de los jugadores

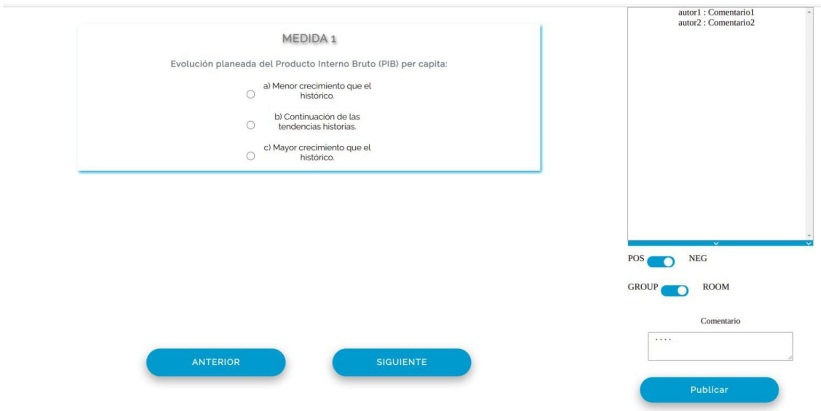

Figura 5.32: Boceto de la pantalla de preguntas para los jugadores

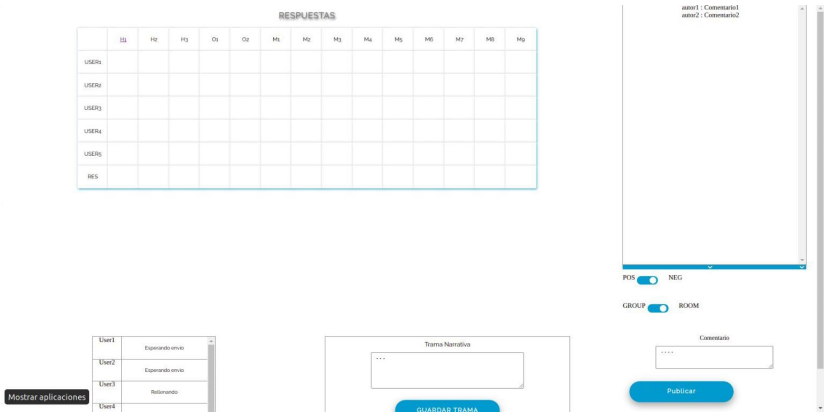

Figura 5.33: Boceto de la pantalla de comprobación de conflictos

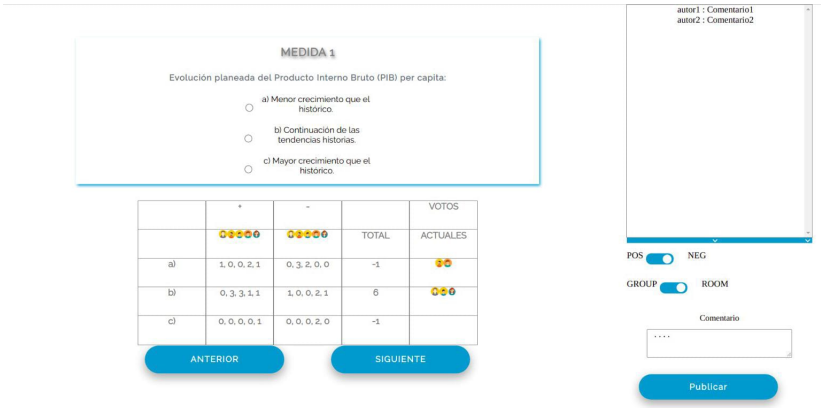

Figura 5.34: Boceto de la pantalla de resolución del conflicto en una pregunta

Figura 5.35: Boceto de la pantalla de resultados de una ronda

Figura 5.36: Boceto de la pantalla de resultados finales del grupo

Capítulo 6

Implementación y pruebas

En este capítulo se detalla todo el proceso de implementación y testing de la aplicación, haciendo hincapié en las decisiones que se han tenido que tomar y que afectan a los modelos realizados durante el análisis y el diseño. El principio que se ha seguido como guía a la hora de realizar este proceso de implementación es el *Principio Scout*, enunciado por Robert C. Martin en el libro *Clean Code* [39]. Dicho principio dice así: "Dejar las cosas mejor de como las he encontrado".

6.1. CI/CD

Durante las primeras fases del proceso de implementación del proyecto, se decidió preparar un pipeline de integración y despliegue continuos con el objetivo de automatizar algunas tareas repetitivas de este proceso de implementación. Aunque inicialmente se planteó la adición de un trabajo que realizase las tareas de testing, pronto se comprobó que el trabajo que ejecutaba los tests de la aplicación tardaba demasiado tiempo en ejecutarse y se alcanzaba el timeout establecido para la ejecución de un pipeline por la plataforma, por lo que se decidió eliminar este trabajo del pipeline.

A continuación, se mostrará el fichero final de integración continua, omitiendo los comentarios que se han añadido a dicho fichero a modo de documentación:

```
image: ubuntu:20.04

variables:
  CYPRESS: node_modules/.bin/cypress

cache:
  key: ${CI_COMMIT_REF_SLUG}

before_script:
```

```
  - 'which ssh-agent || ( apt-get update -y && apt-get install
     openssh-client -y )'
  - eval $(ssh-agent -s)
  - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
  - echo "$SSH_PRIVATE_KEY" > key.prv
  - chmod 600 key.prv
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh
  - git config --global user.email "manuel.alda@alumnos.uva.es"
  - git config --global user.name "manalda"

stages:
  - build
  - test
  - deploy

deploy:
  stage: deploy
  script:
    - ssh -i key.prv  -oStrictHostKeyChecking=no -p 20041
       usuario@virtual.lab.inf.uva.es "cd git/frontend && git
       checkout develop && git stash && git pull origin ${
       CI_COMMIT_BRANCH} && export PATH=/home/usuario/.nvm/
       versions/node/v12.20.2/bin:$PATH && npm install && npm
       run build && cp deploy/.htaccess dist/crossroads-
       frontend/ && echo $SERVER_PASSWORD | sudo -S cp -r dist/
       crossroads-frontend/ /var/www/html/ && echo
       $SERVER_PASSWORD | sudo -S systemctl restart apache2"
  only:
    - develop
    - master
```

En los siguientes apartados de esta sección se explica en detalle el contenido de este fichero.

6.1.1. Preparación

En la sección de preparación del fichero se obtiene una imagen docker en la que se ejecutará el pipeline. Además, se establecen los elementos que se guardarán en la memoria caché (para no tener que descargarlos entre ejecución y ejecución del pipeline) y se definen las fases de las que consta dicho pipeline. Como nota, indicar que no se ha añadido ningún trabajo, de momento, en la fase "build" del pipeline. Pasa de igual manera con la fase "test", por los motivos mencionados anteriormente.

También se define en esta sección un script con comandos que se ejecutarán antes de los

trabajos. En el caso concreto de este fichero, el script realiza la configuración del cliente SSH para poder conectarnos a la máquina virtual y ejecutar comandos en ella.

El fragmento de código que se encarga de la preparación es el siguiente:

```
image: ubuntu:20.04

variables:
  CYPRESS: node_modules/.bin/cypress

cache:
  key: ${CI_COMMIT_REF_SLUG}

before_script:
  - 'which ssh-agent || ( apt-get update -y && apt-get install
     openssh-client -y )'
  - eval $(ssh-agent -s)
  - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
  - echo "$SSH_PRIVATE_KEY" > key.prv
  - chmod 600 key.prv
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh

stages:
  - build
  - test
  - deploy

...
```

6.1.2. Despliegue

En el código restante del fichero de CI/CD se define un trabajo encargado de realizar el despliegue del front-end en la máquina virtual. Para ello realiza una conexión mediante SSH a dicha máquina, hace un checkout a la rama que desencadenó la ejecución del trabajo, realiza una instalación de las dependencias y compila el front-end con la configuración de producción. El resto de comandos que siguen a la compilación son para realizar el despliegue en un servidor Apache (ver instrucciones de despliegue del front-end en el Anexo A). Este trabajo solo se ejecuta en las ramas develop y master.

El fragmento de código encargado de realizar el despliegue es el siguiente:

```
...

deploy:
  stage: deploy
```

```
script:
  - ssh -i key.prv  -oStrictHostKeyChecking=no -p 20041
      usuario@virtual.lab.inf.uva.es "cd git/frontend && git
      checkout develop && git stash && git pull origin ${
      CI_COMMIT_BRANCH} && export PATH=/home/usuario/.nvm/
      versions/node/v12.20.2/bin:$PATH && npm install && npm
      run build && cp deploy/.htaccess dist/crossroads-
      frontend/ && echo $SERVER_PASSWORD | sudo -S cp -r dist/
      crossroads-frontend/ /var/www/html/ && echo
      $SERVER_PASSWORD | sudo -S systemctl restart apache2"
only:
  - develop
  - master
```

6.2. Cambios realizados durante la implementación

A lo largo del proceso de implementación del proyecto ha sido necesario tomar algunas decisiones importantes que afectaban a las conclusiones obtenidas de los procesos de análisis y diseño. En esta sección se detallan algunos de los cambios más importantes que se han aplicado. Esta sección es complementaria al Capítulo 7 (seguimiento), pues en dicho capítulo se presentan los cambios contextualizados en el sprint en el que se decidió aplicarlos.

6.2.1. Cambios en el back-end

Para poder realizar la implementación del front-end, ha sido necesario implementar algunas operaciones en el back-end, pues había datos que se necesitaban y no eran accesibles a través de ninguna de las operaciones existentes en la versión original de dicho back-end. También se añadió una operación nueva para realizar la integración con el servicio externo de recomendación para los grupos. En concreto, se han añadido cinco nuevas operaciones, cuyo detalle se presenta en las Figuras 6.1 a 6.5.

También fue necesario modificar el comportamiento de la operación **getRoomByCode** de **RoomController**, pues dicha operación devolvía los datos de una sala a partir de su código, siempre y cuando la sala estuviese en estado de espera, lo que evitaba la posibilidad de llamar a esa operación en otros puntos en los que la sala se encontraba en otro estado y hacía falta realizar esa llamada. Con la modificación actual, se devuelven los datos de la sala a partir del código de la misma independientemente del estado en que se encuentra dicha sala.

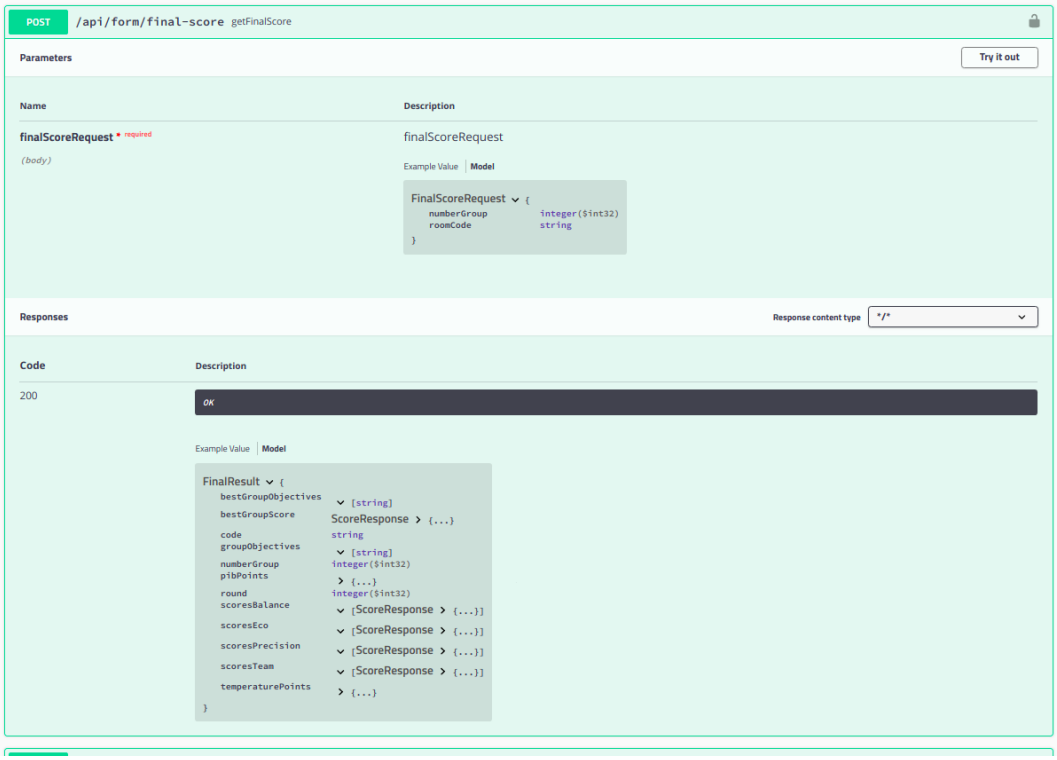

Figura 6.1: Esquema de la operación getFinalScore

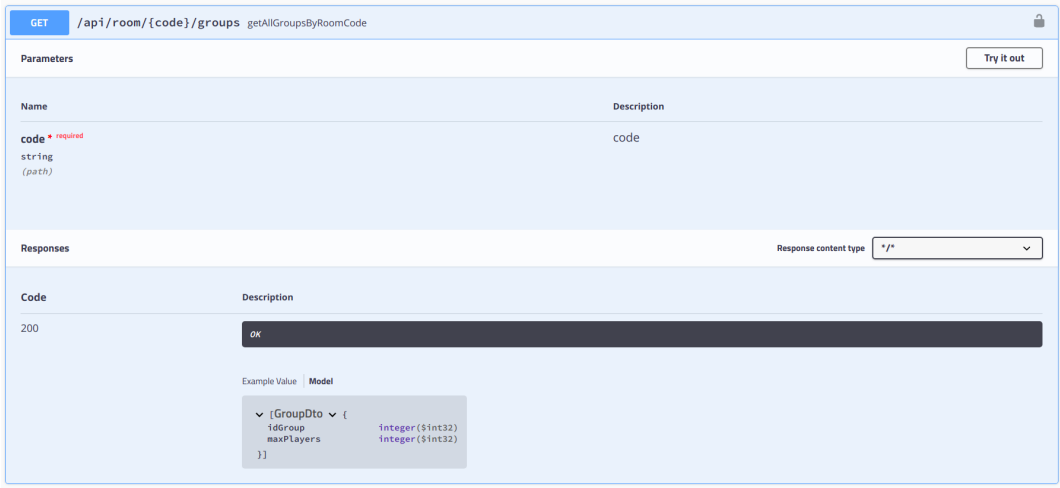

Figura 6.2: Esquema de la operación getAllGroups

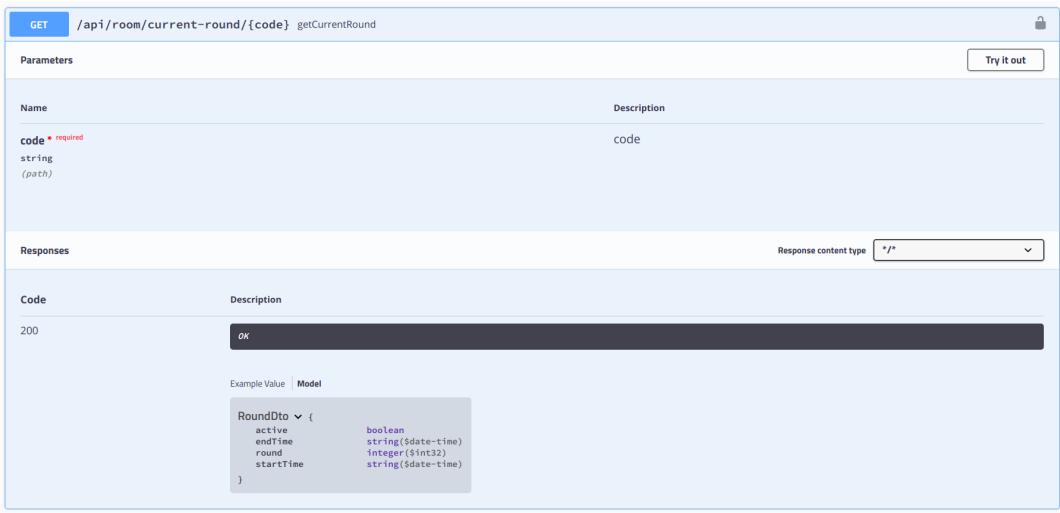

Figura 6.3: Esquema de la operación getCurrentRound

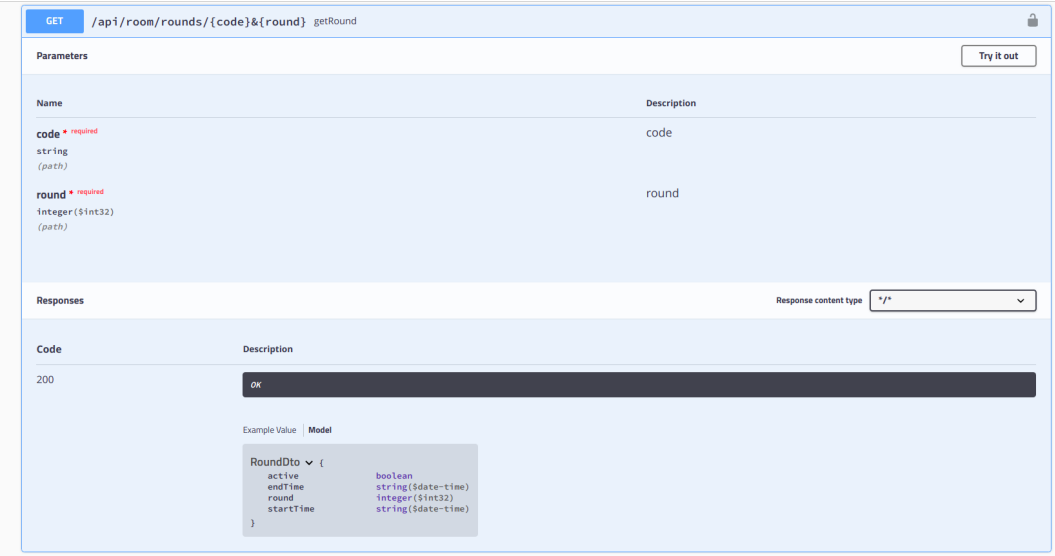

Figura 6.4: Esquema de la operación getRound

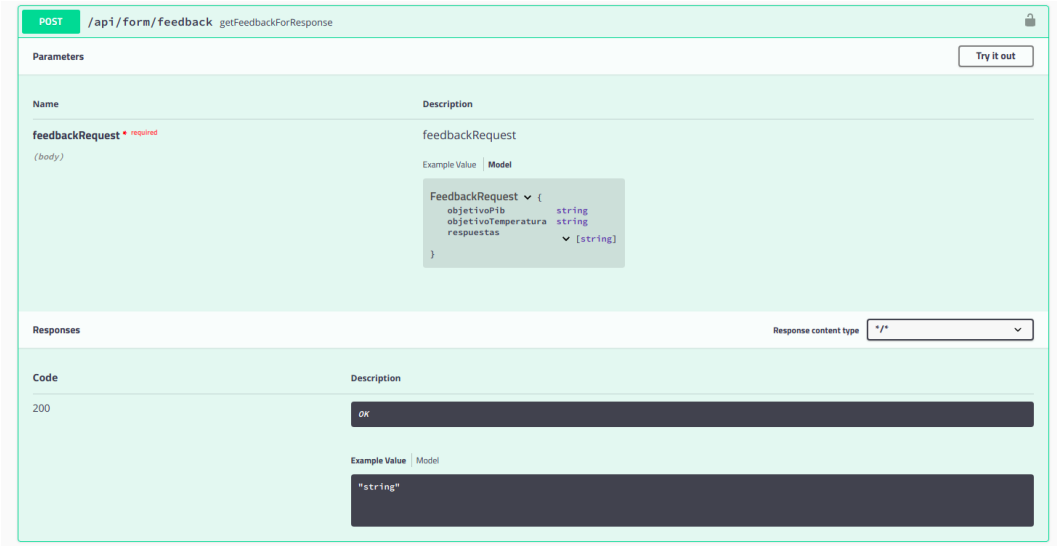

Figura 6.5: Esquema de la operación getFeedback

6.2.2. Cambios en los modelos

El otro tipo de cambios que hubo que realizar fue en los modelos. Los principales cambios se resumen en el siguiente listado:

- Eliminación de la trama narrativa.
- Implementación del estado del grupo en el front-end.

Todos estos cambios se abordaron por motivos de calendario y/o complejidad técnica. En el caso del estado del grupo, se prevé abordar la implementación inicial en un futuro.

Aquí también habría que citar los cambios realizados durante el proceso de análisis con respecto a la versión original de esos modelos (Capítulo 4). Por motivos de brevedad, no volveremos a incluir el detalle de cada uno de ellos.

6.3. Testing

El testing es el proceso mediante el cual se realizan pruebas de la aplicación para detectar la existencia de errores en el software desarrollado. Se dice que el resultado de un test es positivo si dicho test detecta un error y negativo en caso contrario.

El testing de Crossroads se ha dividido en dos partes, una encargada de probar el back-end y otra encargada de hacer probar el front-end. Como las pruebas del back-end ya fueron

realizadas en el TFG de partida [37], nos centraremos en esta sección en las pruebas realizadas para el front-end.

Se decidió que se realizarían pruebas end-to-end (e2e [58]), pues estas pruebas nos permitirían probar la aplicación de manera similar a como lo haría el usuario. Además, el testing e2e serviría para reforzar los resultados del testing del back-end, puesto que en este tipo de pruebas se utiliza el servidor con su implementación real, en lugar de utilizar un "servidor falso" con el comportamiento esperado del mismo, como ocurre con los test unitarios. De esta forma, también se podrían probar las nuevas operaciones que fue necesario añadir al back-end.

También se decidió realizar pruebas de usabilidad con usuarios reales, pero no se han podido abordar por motivos de tiempo, por lo que quedan pendientes para realizarlas en el futuro.

6.3.1. Testing e2e

Para realizar el testing e2e se ha elaborado una batería de pruebas con 135 casos de prueba. Los casos de prueba están agrupados en ficheros JavaScript, donde cada fichero almacena los casos de prueba del componente al que hace referencia el fichero. Como las pruebas son automáticas (por la naturaleza del testing e2e), no se detallarán aquí cada uno de los casos de prueba, pues la documentación de los mismos reside en el código que realiza las pruebas.

Para mejorar la capacidad de mantenimiento de los test, se decidió aplicar el patrón de diseño PageObject [38], mediante el cual se crean unas clases que encapsulan el acceso a los elementos de la interfaz. De esta manera, los test se programan haciendo uso de la interfaz que proporcionan los PageObjects y, si la interfaz cambia, solamente tendría que cambiar el PageObject correspondiente, en lugar de tener que cambiar el caso de prueba.

En cuanto al propósito de conseguir lograr que unos casos de prueba sean independientes de otros y que sus resultados sean deterministas, hubo que crear un plugin para Cypress que se encargase de borrar los datos de la base de datos MySQL entre ejecución y ejecución de los casos de prueba. También se creó un plugin encargado de eliminar el contenido de una carpeta, pues se hacía necesario dicho comportamiento para algunos casos de prueba que implicaban la descarga de un fichero pdf.

El proceso de testing ha tenido un resultado positivo, pues ha permitido sacar a la luz errores en la aplicación. En el momento de finalización del proyecto (Capítulo 7), el porcentaje de éxito de los tests es del 100 %, es decir, todos los tests obtienen el resultado esperado para ellos. Además, se ha obtenido un informe de los distintos tipos de cobertura de código [15], para comprobar la calidad del proceso de testing realizado.

Dicho informe se encuentra en la Figura 6.6. Como se puede observar, se han obtenido porcentajes altos de cobertura, lo que es un buen indicador de la calidad del testing realizado.

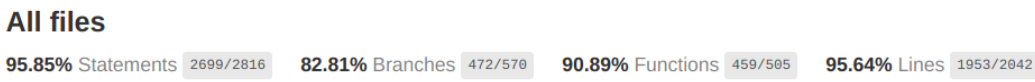

Figura 6.6: Informe de cobertura del código

6.4. Organización del código

A continuación, se presenta la organización del repositorio que contiene el código del front-end. Por motivos de brevedad, se omitirá el contenido de algunos directorios, aunque se explicará el contenido de los mismos. También se omitirán algunos ficheros de configuración, por no ser demasiado relevantes.

```
|-- angular.json
|-- package.json
|-- package-lock.json
|-- cypress.json
|-- README.md
|-- LICENSE
|-- .gitlab-cy.yml
|-- .gitignore
|-- cypress
    |-- clear_dir
    |-- db
    |-- fixtures
    |-- integration
    |-- pages
    |-- plugins
    |-- resources
    |-- support
|-- deploy
|-- src
    |-- app
        |-- enums
        |-- guards
        |-- interceptors
        |-- models
        |-- pages
        |-- services
        |-- app-routing.module.ts
        |-- app.component.html
        |-- app.component.scss
        |-- app.component.ts
        |-- app.module.ts
    |-- assets
    |-- environments
    |-- styles
```

```
    |-- index.html
```

En el siguiente listado, se comentan los contenidos de cada carpeta de la estructura anterior:

- Los primeros ficheros son de configuración para el proyecto, para Angular, para Cypress, entre otros. También se incluye el fichero README con información acerca del repositorio, el fichero de integración continua, el gitignore y la licencia.
- Dentro de la carpeta **cypress**, encontramos un conjunto de subcarpetas con todos los elementos necesarios para ejecutar los test e2e. A destacar, la carpeta **integration** que contiene los ficheros con los tests, la carpeta **pages** que contiene todos los PageObjects que se han creado para encapsular el contenido de los ficheros HTML, la carpeta **fixtures** que contiene algunos ficheros JSON con datos de prueba, la carpeta **resources** con recursos utilizados en los test y las carpetas **clear_dir**, **db**, **plugins** y **support**, que incluyen los elementos necesarios para definir nuevos plugins y comandos para utilizar durante los test y algunos de estos elementos definidos.
- La carpeta **cypress** contiene unas intrucciones para realizar el despliegue de la aplicación, entre otros.
- La carpeta **src** contiene todo el código fuente de la aplicación (carpeta **app**), así como archivos de estilo para la aplicación (carpeta **styles**), archivos de recursos (carpeta **assets**), variables de entorno que cambian dependiendo de si se compila la app en modo producción o en modo desarrollo (carpeta **environments**) y la página principal de la app (fichero index.html).
- Dentro de la carpeta **app**, encontramos las subcarpetas **enums** y **models**, que incluyen las clases del dominio definidas durante el análisis (Capítulo 4). También encontramos aquí, las carpetas **pages** y **services** que incluyen los componentes Angular y los servicios que conforman la aplicación. Para finalizar, encontramos las carpetas **guards** e **interceptors** que contienen elementos de control para la navegación y la realización de peticiones HTTP, respectivamente.
- El resto de ficheros de la carpeta **app** constituyen el componente principal de la aplicación.

6.5. Problemas y dificultades superadas

En esta sección se presentan algunos de los principales problemas que se han tenido que abordar durante el proceso de implementación. Para obtener una exposición más exhaustiva de todas estas dificultades, es necesario consultar el Capítulo 7, que describe el seguimiento del proyecto sprint a sprint.

Las principales dificultades superadas aparecen en el siguiente listado, junto a la solución alcanzada para resolverlo:

- Errores por la política CORS al realizar peticiones HTTP desde el front-end hasta el back-end en la máquina virtual. Para su resolución hubo que añadir una nueva clase de configuración al back-end que interceptase la respuesta a cada petición para añadir las cabeceras correspondientes que eviten que el navegador de un error por la política CORS.
- Aplicación de una cuenta atrás en algunas partes de la aplicación como la carga de los resultados finales de la partida. Para ello se crearon unas funciones que realizan la parada del hilo principal de la aplicación para, al reanudar su ejecución, realizar la tarea correspondiente. Varias funciones de este tipo se han añadido en distintos componentes del front-end.
- Reutilización del código del chat y creación de una interfaz uniforme para el acceso a su funcionalidad. Para conseguirlo se creó un nuevo componente. Además, hubo que reducir al mínimo los datos externos que necesitaría dicho componente para realizar su función. Estos datos externos que puede recibir el chat son los identificadores asociados a una pregunta y una respuesta y que actúan como valores por defecto de la pregunta y la respuesta que llevan asociados los mensajes que se envían por el chat. Además, dichos valores no son estrictamente necesarios para que el chat funcione, por lo que este componente puede ser reutilizado en otros componentes que no tengan acceso a estos datos.

6.6. Licencia

Una vez finalizada la primera versión jugable de la aplicación, se publicará el código bajo una licencia GNU GPL v3 [26]. El código podrá encontrarse en dos repositorios Gitlab, uno para el front-end y otro para el back-end (Anexo B)

Capítulo 7

Seguimiento del proyecto

En este Capítulo se verá en detalle el transcurso de todos los sprints realizados para el desarrollo del proyecto, incluyendo la relación de las tareas que se han abordado en cada uno de ellos, las decisiones que se han tenido que tomar en cada momento del proyecto y las dificultades que ha habido que hacer frente a lo largo del mismo. Al final del Capítulo, se presenta una valoración del seguimiento del proyecto, realizando una comparación con la planificación (Capítulo 2).

7.1. Introducción

El proyecto comenzó el día 19 de enero de 2021, pero antes de empezar con el desarrollo del mismo se decidió dejar un período inicial de tres semanas en que se realizaron labores de formación y preparación a modo de sprint 0.

7.2. Sprint 0

Durante el sprint 0 se llevó una toma de contacto con el dominio del proyecto Crossroads, con el objetivo de familiarizarme con los requisitos y conceptos relativos al dominio de la aplicación. Asimismo, se realizaron tareas de aprendizaje sobre Angular, mediante la realización de tutoriales (como el propio tutorial de Angular [8]) y pequeños proyectos de ejemplo. Además, se procedió a realizar la instalación del entorno de desarrollo para el front-end, así como el despliegue del back-end en el equipo de desarrollo (mirar guía de despliegue en el Anexo A)

7.3. Sprint 1

Durante el sprint planning se decidió que en este sprint se abordaría el desarrollo de las historias de usuario 1.1, 1.2 y 2.1. La relación de puntos de historia asignados a cada historia de usuario puede observarse en la Tabla 7.1. En la Tabla 7.2 se presenta el detalle de todas las tareas realizadas para cada historia de usuario en el sprint 1.

Historia de usuario	Puntos de historia
1.1	2
1.2	1
2.1	3

Tabla 7.1: Asignación de puntos de historia para el sprint backlog 1

En este sprint se abordaron las primeras historias de usuario. El trabajo de implementación de dichas historias no causó casi ningún problema, debido a que no se requerían muchas llamadas a la API de Crossroads para implementarlas y a que estas historias no diferían demasiado de los trabajos realizados durante el sprint 0. De estas tareas de implementación solo hay que mencionar la necesidad de realizar un estudio inicial sobre conceptos relativos a los *reactive forms* y a un problema con la política *CORS* [45] a la hora de implementar (y probar manualmente) la historia de usuario 2.1, problema que, después de consultarlo con la tutora, se resolvió mediante el uso de un navegador chrome con la política CORS desactivada [22].

La mayor fuente de problemas residió en las tareas de testing, principalmente por el desconocimiento del framework de testing e2e Cypress [17]. A pesar del estudio inicial del framework realizado durante este sprint, fueron surgiendo problemas, de entre los cuales hay que destacar la dependencia que había entre los test que iba creando y la imposibilidad de repetir los test con el mismo resultado. Estos dos problemas se debían a los datos que se introducían en la base de datos del back-end al ejecutar los test, por lo que, para solucionarlo, se decidió crear una utilidad que borrase la base de datos y la volviese a poblar antes de la ejecución de cada suite de tests. En un primer instante, se decidió crear un script bash que realizase estas dos tareas. Dicho script sería ejecutado mediante un nuevo comando Cypress que había que añadir en el fichero correspondiente. El script fue creado y funcionaba, pero no se consiguió que Cypress encontrase dicho script para ejecutarlo. El siguiente paso fue ejecutar los comandos directamente desde el comando que provee Cypress [18], pero tampoco tuvo éxito esta solución debido a que los comandos hacían uso de un fichero sql y, una vez más, no conseguí que Cypress leyese un fichero. Finalmente, se optó por una solución bastante tosca que hacía uso de un módulo de Node.js que implementaba conexiones con bases de datos MySQL [24, 71]. Para ello se creó un plugin Cypress [19] que ejecutaba mediante este módulo de Node unas sentencias SQL que estaban almacenadas en un array. El plugin se probó a ejecutar desde Cypress y realizaba bien la tarea, pero se decidió realizar un refactoring de dicho plugin en siguientes sprints.

Debido a todos estos problemas, no se pudo acabar las tareas de testing de las historias de usuario 1.2 y 2.1. La finalización de estas tareas se ha decidido dejarla para el siguiente sprint.

HU	Tarea	T. estimado	T. empleado	Estado
-	Estudio de angular sobre reactive forms	2h	3h 30m	Completado
-	Estudio de Cypress para testing e2e	2h	4h	Completado
-	Creación de una utilidad para resetear la base de datos entre test y test	2h	5h 6m	Completado
1.1	Implementación registrarse	4h	2h 48m	Completado
1.1	Documentación registrarse	30m	30m	Completado
1.1	Testing registrarse	5h 30m	5h 42m	Completado
1.1	Cambiar mensaje registro fallido por contraseña por algo que ayude más al usuario	30m	36m	Completado
1.1	Cambiar expresiones regulares que comprueban la contraseña en el back-end por fallo detectado	30m	18m	Completado
1.2	Implementación login	1h	35m	Completado
1.2	Documentación login	30m	13m	Completado
1.2	Testing login	3h 30m	2h 45m	En proceso
2.1	Implementación crear sala	5h	4h 3m	Completado
2.1	Documentación crear sala	30m	12m	Completado
2.1	Testing crear sala	9h 30m	-	No iniciado
Total		37h	30h 18m	12:14

Tabla 7.2: Seguimiento del sprint 1

7.4. Sprint 2

Durante el sprint planning se decidió abordar el desarrollo de las historias de usuario 3.1 y 2.2. Además, se decidió acabar las tareas de testing que quedaron del sprint anterior, así como la revisión y refactorización de la utilidad para resetear la base de datos antes de cada test y la realización del despliegue del back-end y el front-end en una máquina virtual proporcionada por la Escuela (junto con la creación de un script de integración continua para Gitlab que realice el despliegue y ejecute los tests).

En la Tabla 7.3 se muestra la asignación de puntos de historia para las historias de usuario de este sprint y en la Tabla 7.4 se muestra al detalle el seguimiento de todas las tareas a realizar en este sprint.

Historia de usuario	Puntos de historia
3.1	3
2.2	3

Tabla 7.3: Asignación de puntos de historia para el sprint backlog 2

De este sprint cabe destacar el surgimiento de varias incidencias a mitad de sprint que hubo que solucionar antes de poder avanzar. En primer lugar, durante la primera semana de sprint, a sugerencia de la tutora del TFG hubo que realizar un estudio sobre distintos patrones de diseño para testing e2e, con el objetivo de mejorar la reutilización de código y disminuir la dependencia de los tests del HTML de las páginas. Se decidió aplicar el patrón PageObject con algunas modificaciones para cumplir con las buenas prácticas de Cypress (por ejemplo, los PageObjects no devuelven otros PageObject ya que, en algunos casos, para ello hacía falta añadir lógica condicional a los tests). Esto llevó a cambiar los tests realizados en el sprint anterior para que éstos utilizasen los PageObjects creados. En segundo lugar, durante la implementación de la historia de usuario unirse a sala se detectó que en el back-end hacía falta una operación que devolviese todos los grupos de una sala, ya que, con las operaciones disponibles en el back-end no se podía obtener información suficiente como para controlar los grupos de la sala y la unión de jugadores a los mismos (hacía falta el id del grupo). Además, también se descubrió que en el back-end no se realizaba un control del número de usuarios que se unen a la sala o a los grupos, por lo que podía darse el caso de una sala con más jugadores del máximo establecido en la creación o de un grupo con más jugadores del permitido. Por ello, hubo que aplicar el plan de contingencia del riesgo descrito en la Tabla 2.8 y añadir unas tareas en el sprint actual en las que modificar el back-end. Para finalizar, hubo problemas con el despliegue del back-end y el front-end en la máquina virtual, por lo que hubo que pedir a los técnicos de la escuela más espacio de disco en la máquina virtual.

Todas estas incidencias, junto con la realización de una planificación optimista de las tareas contempladas para el sprint actual, desembocaron en la imposibilidad de completar muchas tareas, tareas que, según lo establecido en el plan de contingencia del riesgo descrito en la Tabla 2.5 se pasarán al sprint siguiente.

7.5. Sprint 3

Durante el sprint planning de este sprint se decidió completar todas las tareas que quedaron en estado *No iniciado* o en estado *En proceso* del sprint número dos. Como la estimación de la carga de trabajo de esas tareas alcanza las 30 horas de trabajo estimadas para el sprint, se ha decidido no añadir ninguna historia de usuario o tarea nuevas al backlog del sprint. La relación de tareas realizadas durante este sprint se encuentra en la Tabla 7.5.

Como hechos relevantes de este sprint, cabe destacar que, para ahorrar tiempo de desarrollo y aprovechar buena parte del código desarrollado en el sprint anterior de la historia de usuario 2.2, se decidió reutilizar el componente de Angular utilizado para la vista de dicha historia de usuario para la historia de usuario 3.1, debido a que buena parte de los datos y de la funcionalidad necesarios para la historia 3.1 ya estaban desarrollados en la historia 2.2. También hay que mencionar que, hacia la mitad del sprint, se planteó la posibilidad de colaborar con otro proyecto, por lo que hubo que realizar una integración del código del otro proyecto con el de este, tanto en el back-end como en el front-end.

Al finalizar este sprint, se terminaron todas las tareas que quedaban pendientes del sprint anterior, excepto el despliegue del front-end y del back-end mediante Gitlab CI/CD, debido a

HU	Tarea	T. estimado	T. empleado	Estado
_	Añadir al back-end una operación para obtener todos los grupos de una sala	2h	1h 7m	Completado
_	Desplegar front-end y back-end en una máquina virtual	10h	9h 3m	En proceso
_	Estudio sobre patrones de diseño para testing e2e	1h 30m	1h 13m	Completado
_	Crear PageObjects para los test de los componentes home y regsiter	1h 20m	1h 42m	Completado
_	Preparar sesión antes de los tests	3h	57m	Completado
_	Añadir una carpeta de recursos de test y modificar plugin reset DB	2h	1h 50m	Completado
1.1	Terminar testing login	45m	58m	Completado
2.1	Terminar testing crear sala	9h 30m	1h 12m	Completado
2.2	Implementación iniciar partida	6h 30m	0h	No iniciado
2.2	Documentación iniciar partida	30m	0h	No iniciado
2.2	Testing iniciar partida	8h	0h	No iniciado
3.1	Implementación unirse a sala	8h	12h 28m	En proceso
3.1	Documentación unirse a sala	30m	30m	Completado
3.1	Testing unirse a sala	7h	0h	No iniciado
Total		62h 30m	32h	8:14

Tabla 7.4: Seguimiento del sprint 2

que se descubrió la imposibilidad de hacer un pull del repositorio desde la máquina virtual a través de SSH. Este hecho se descubrió que ocurría a causa de un problema de configuración en el Gitlab de la escuela, que no permitía la realización de operaciones a través del protocolo SSH.

7.6. Sprint 4

Este sprint es especial porque, debido a la planificación de las vacaciones de semana santa en el desarrollo del proyecto, dura tres semanas en lugar de dos. Para este sprint se planificó la finalización del despliegue del front-end y back-end en la máquina virtual mediante CI. Asimismo, se planificó el desarrollo de dos historias de usuario, cuya relación de puntos de historia aparece en la Tabla 7.6. Antes de pasar a comentar el desarrollo de este sprint, es necesario indicar que, a partir de este momento, las tareas de documentación del código se incluirán en las tareas de implementación de la historia de usuario correspondiente. Esto se debe a que esas tareas tienen una duración muy pequeña en comparación con el resto de tareas, por lo que no tiene mucho sentido añadir una tarea para tan poco tiempo.

HU	Tarea	T. estimado	T. empleado	Estado
-	Integrar cambios en el back-end de otro proyecto colaborador	2h	2h	Completado
-	Integrar cambios en el front-end de otro proyecto colaborador	2h	1h 30m	Completado
-	Añadir en el back-end control sobre la cantidad de usuarios que entran en la sala y en un grupo	1h 20m	1h	Completado
-	Desplegar front-end y back-end en una máquina virtual	10h	4h 47m	En proceso
2.2	Implementación iniciar partida	6h 30m	4h 1m	Completado
2.2	Documentación iniciar partida	30m	23m	Completado
2.2	Testing iniciar partida	8h	4h 6m	Completado
3.1	Implementación unirse a sala	8h	2h 20m	Completado
3.1	Testing unirse a sala	7h	6h 15m	Completado
Total		45h 30m	26h 22m	8:9

Tabla 7.5: Seguimiento del sprint 3

Historia de usuario	Puntos de historia
2.3	2
2.4	3

Tabla 7.6: Asignación de puntos de historia para el sprint backlog 4

Este sprint fue el primer sprint en el que se alcanzaron todos los objetivos y se terminaron todas las tareas. En cuanto a hechos reseñables, el despliegue mediante SSH siguió dando problemas, debido a que, a pesar de que los técnicos de la escuela habilitaron la posibilidad de ejecutar los comandos *git pull* y *git clone* mediante SSH, estas operaciones no se podían realizar desde la red de la escuela y, por tanto, no se podían realizar desde la máquina virtual para hacer el despliegue. Una vez puesto en conocimiento de los técnicos de la escuela, se arregló dicho error, lo que terminaría en la finalización del despliegue del proyecto mediante el CI de Gitlab. También se intentó abordar el despliegue del back-end de forma automática, pero, después de muchos intentos y de búsqueda de información, se encontró en *Stackoverflow* un post en el que se indicaba que no era posible mantener la ejecución del back-end en segundo plano una vez terminado el trabajo del script de CI [64], por lo que se descartó esta parte del despliegue automático.

Además, durante la implementación de las historias de usuario 2.3 y 2.4, se detectó la necesidad de añadir dos operaciones en el back-end, una para obtener los datos de la ronda actual y otra para obtener el estado de una ronda a partir del número de ronda y del código de la sala, debido a que, la única operación que había en el back-end para comprobar el estado de la partida no proporcionaba información suficiente como para realizar un control adecuado desde el front-end. También se decidió cambiar la operación del back-end que devolvía una sala a partir de su código si dicha sala estaba en estado *WAITING* debido a que esta operación se hacía necesaria cuando la sala estaba en otros estados. Como estas

HU	Tarea	T. estimado	T. empleado	Estado
_	Despliegue manual de la versión integrada del proyecto en la máquina virtual	1h	53m	Completado
_	Desplegar front-end y back-end en una máquina virtual mediante CI/CD	10h	6h 57m	Completado
-	Comienzo preparación tests funcionalidad integrada	1h	50m	Completado
2.3	Implementación iniciar nueva ronda	5h	3h 25m	Completado
2.3	Testing iniciar nueva ronda	5h	5h 37m	Completado
2.4	Implementación terminar ronda	8h	12h 23m	Completado
2.4	Testing terminar ronda	7h	2h 18m	Completado
Total		37h	32h 23m	7:7

Tabla 7.7: Seguimiento del sprint 4

modificaciones del back-end eran esenciales para la implementación de las historias 2.3 y 2.4, se decidió no añadir tareas nuevas y contemplar esos cambios como parte de las tareas de implementación de dichas historias.

7.7. Sprint 5

Para el sprint número cinco se planificó realizar una nueva integración con el proyecto colaborador, para añadir la funcionalidad desarrollada por dicho proyecto en la parte del jugador a la funcionalidad de este proyecto. Además, se planificó arreglar la funcionalidad del logout, debido a que se detectó que no se realizaba correctamente en el caso de que el token JWT estuviese caducado y abordar el desarrollo de las historias de usuario 2.5 y 2.6. La relación de los puntos de historia asignados en este sprint aparece en la Tabla 7.8

Historia de usuario	Puntos de historia
2.5	3
2.6	1

Tabla 7.8: Asignación de puntos de historia para el sprint backlog 5

En este sprint se realizó una prueba con el *Product Owner* en la que se detectó un error en el despliegue mediante CI. El error en concreto consistía en que no se había actualizado correctamente la versión de la aplicación desplegada en la máquina virtual. Tras un proceso de debbuging se comprobó que la sintaxis del comando de copia que ejecutaba el script de CI era incorrecta, por lo que no se estaban copiando bien los artefactos de la aplicación web a la carpeta del servidor Apache de la máquina virtual. Una vez corregida la sintaxis de dicho comando, el despliegue se realizó de manera adecuada.

HU	Tarea	T. estimado	T. empleado	Estado
-	Integración de los cambios del otro proyecto colaborador	5h	1h 18m	Completado
-	Arreglar el logout	5h	5h 27m	Completado
2.5	Implementación terminar partida	7h 30m	1h 20m	Completado
2.5	Testing terminar partida	7h 30m	1h 4m	Completado
2.6	Implementación generar un informe con los resultados de la partida	3h	1h 47m	Completado
2.6	Testing generar un informe con los resultados de la partida	2h	1h 44m	Completado
2.4 y 2.5	Arreglar bugs en terminar partida y terminar ronda	5h 20m	7h 26m	Completado
-	Arreglar bug con el despliegue automático	1h	42m	Completado
2.2	Corrección bug empezar partida	1h	50m	Completado
Total		37h 20m	21h 38m	9:9

Tabla 7.9: Seguimiento del sprint 5

También se detectaron bugs durante la integración con el otro proyecto en la funcionalidad de terminar ronda y la versión desarrollada en este sprint de terminar partida, pues hasta este momento sólo se tenían en cuenta los casos en los que el jugador no contestaba preguntas y, al integrar esta funcionalidad, se comprobó que la lógica no servía para ambos casos, por lo que hubo que actualizarla.

Para la implementación de la historia de usuario 2.6 (generar informe de partida) hubo que instalar mediante NPM un paquete llamado *File Saver* [25] que proporciona la funcionalidad de guardar un fichero en la carpeta de descargas del ordenador. En relación al testing de esta historia de usuario, hubo que crear un plugin para Cypress que, de manera similar al plugin que reseteaba la base de datos, borrase el contenido de la carpeta de descargas, con el objetivo de hacer que los test se pudiesen repetir (ya que, si el fichero ya existe de un test anterior puede enmascararse un error). Para ello se hizo uso de un paquete de NodeJS llamado *File System* que proporciona operaciones sobre ficheros en la máquina en la que se está ejecutando la aplicación. La funcionalidad concreta del plugin consite en la eliminación de manera recursiva de los ficheros de la carpeta de descargas del navegador que utiliza Cypress para los tests.

Finalmente, se encontró y corrigió el error que había con la funcionalidad de logout. Este error consistía en un fallo a la hora de refrescar el token en el back-end, puesto que no se devolvía en el back-end un código de error 401 cuando debería hacerlo si el token estaba caducado [32]. Además, en el front-end no se gestionaban bien los errores en la operación de refrescar el token, por lo que, a pesar de haber arreglado la parte del back-end, se producía

el mismo error para ciertos casos en los que el token de refresco era incorrecto. Una vez arreglados estos errores, el logout pasó a funcionar correctamente.

7.8. Sprint 6

En este sprint se añadieron dos nuevas historias al product backlog, la historia 2.8 que dice así: "Como moderador quiero ver los jugadores que están en cada grupo con la información del estado en que se encuentraz la historia 2.9 que dice lo siguiente: "Como moderador quiero poder ver la información sobre el desarrollo de cada ronda". Estas dos historias se han planificado para realizarlas en este sprint número 6 y la relación de puntos de historia asignados a cada una de ellas aparece en la Tabla 7.10. También se planificó realizar una nueva integración de funcionalidad con el proyecto colaborador.

Historia de usuario	Puntos de historia
2.8	2
2.9	4

Tabla 7.10: Asignación de puntos de historia para el sprint backlog 6

En la Tabla 7.11 se puede observar el detalle de las tareas abordadas en el sprint. Cabe destacar que, para implementar ambas historias de usuario se utilizó una tabla en la que se mostraban en cada fila, en el caso de la historia 2.8, los datos de los jugadores de un grupo (nickname, edad, país y estado) o los datos de un grupo durante una ronda (número de grupo, número de jugadores del grupo y estado del grupo), en el caso de la historia 2.9. Para crear la tabla de forma dinámica, se utilizó el componente *Table* que proporciona el framework visual PrimeNG [54, 57] pues con dicho componente se simplificaba mucho la tarea de crear la tabla con los datos.

En relación al estado de un grupo durante una ronda, se ha comprobado que no hay soporte en el servidor para realizarla. De momento, se ha realizado en el front-end, dejando para más adelante su implementación en el back-end. Esta decisión se ha tomado por motivos de calendarización y bajo previa consulta con el *Product Owner*.

Para finalizar el sprint, se realizó la integración con el otro proyecto. Este proceso se realizó sin muchos problemas, tan sólo un problema de dependencias que se solucionó al actualizar el proyecto a Angular 12.

7.9. Sprint 7

En este sprint se añadió al product backlog una nueva historia de usuario compuesta por dos historias de usuario más pequeñas. Esta historia se numeró como la historia 2.10 que dice así: "Como usuario moderador quiero poder acceder a los datos de salas que haya creado". Esta historia se dividió en dos partes, la parte 2.10a que refleja la funcionalidad relacionada

HU	Tarea	T. estimado	T. empleado	Estado
-	Integración de la funcionalidad del jugador y del moderador	5h	2h 21m	Completado
2.8	Implementación mostrar información de un grupo	5h	6h 2m	Completado
2.8	Testing mostrar información de un grupo	5h	4h 46m	Completado
2.9	Implementación mostrar información de una ronda	10h	8h 7m	Completado
2.9	Testing mostrar información de una ronda	10h	9h 8m	Completado
Total		35h	31h 24m	5:5

Tabla 7.11: Seguimiento del sprint 6

con la búsqueda de las salas creadas y la parte 2.10b que refleja los aspectos relacionados con el acceso a los datos de una sala en concreto. Para este sprint se planificó el desarrollo de estas dos historias de usuario nuevas (Tabla 7.12) y la adición de un proxy al front-end para evitar los errores por CORS y así poder utilizar el navegador sin la seguridad desactivada, como veníamos haciendo desde el sprint 1.

Historia de usuario	Puntos de historia
2.10a	2
2.10b	2

Tabla 7.12: Asignación de puntos de historia para el sprint backlog 7

El detalle de las tareas realizadas durante este sprint se presenta en la Tabla 7.13. Para la implementación de la historia de usuario 2.10a se utilizó el mismo componente visual de tablas que en el sprint anterior [57] junto con el componente de paginación [56] puesto que se consideró como necesario el construir una tabla con todas las salas creadas por un usuario y mostrar las salas de cinco en cinco. Las columnas de dicha tabla mostraban la proporción de jugadores de la sala, el nombre de la misma y un botón para implementar el acceso a la sala. Para el acceso a los datos de la sala se reutilizó el componente *Dashboard*, puesto que la vista que se quería implementar era muy similar a la de dicho componente. Para diferenciar la funcionalidad que debía permitir dicho componente se realiza una comprobación de la URL que ha llamado a dicho componente, de manera similar a lo realizado con el componente *Waiting* en el sprint número 3.

Debido a esta reutilización, se planteó a mitad de sprint la posibilidad de permitir al moderador reconectarse a la sala si la había abandonado o había perdido la conexión. Para ello se añadió una nueva columna a la tabla en la que se muestra el estado de la sala y, si la partida está activa, se pide confirmación al usuario para volver a conectarse.

Para finalizar el sprint, se realizó un estudio sobre el funcionamiento de un proxy de angular [5, 20, 66] con el objetivo de añadir un proxy que evitase los fallos por la política CORS al realizar llamadas al back-end. Al final se añadieron dos proxys, uno para tiempo de

desarrollo que es reemplazado por otro cuando se compila la aplicación con la configuración de producción, ya que con esta configuración hay que utilizar la URL de la aplicación en la máquina virtual. El proxy con la configuración de producción no se ha podido probar, debido a que apareció un error en el despliegue en la máquina virtual tras actualizar el proyecto a Angular 12.

HU	Tarea	T. estimado	T. empleado	Estado
-	Introducción de un proxy para evitar errores con el CORS al utilizar un navegador con la seguridad activada	5h	3h	Completado
-	Modificar botón de inicio para que redirija a la pantalla de menú si hay un usuario logeado	1h	2h	Completado
2.10a	Implementación buscar partidas	7h	3h 20m	Completado
2.10b	Implementación acceso a las partidas buscadas	5h	2h 31m	Completado
2.10	Testing búsqueda de partidas y acceso a partida	8h	5h 17m	Completado
2.10	Reconexión del moderador a una partida en juego (a través de buscar partida)	3h	1h 37m	Completado
Total		39h	17h 45m	6:6

Tabla 7.13: Seguimiento del sprint 7

7.10. Sprint 8

En este sprint se decide dar prioridad absoluta a arreglar el despliegue de la aplicación en la máquina virtual. Además, se decide dejar de colaborar con el proyecto con el que se venía colaborando desde el sprint número 3, por lo que se decidió abordar la finalización y revisión de algunas de las historias de usuario que se estaban desarrollando desde el otro proyecto. Para finalizar, se añadió una tarea en la que revisar la pantalla de espera del moderador, pues con la última integración con el código del otro proyecto surgieron inconsistencias. La asignación final de puntos de historia se detalla en la Tabla 7.14.

En la Tabla 7.15 se presenta el resumen de las tareas realizadas durante este sprint. En primer lugar, se abordó el problema de despliegue de la app en la máquina virtual. Al final, se descubrió que el error que salía al compilar la aplicación se debía a un conflicto de dependencias, ya que la versión de *Node* utilizada no era compatible con la versión 12 de Angular [63, 9]. Una vez actualizado *Node* a la versión *12.20.2* se solucionó el error y se pudo

Historia de usuario	Puntos de historia
3.5a	2
3.5b	2
3.7	2

Tabla 7.14: Asignación de puntos de historia para el sprint backlog 8

realizar el despliegue de la app en la máquina virtual. En este momento se realizaron pruebas del proxy en la máquina virtual, pero sin resultados positivos. Investigando, se descubrió que el proxy solo funciona con el servidor de desarrollo que proporciona Angular [5], por lo que se decidió dejar de lado, de momento, la resolución de los problemas con el CORS en la máquina virtual puesto que la parte más prioritaria de esta tarea (arreglar el despliegue) había sido realizada.

En segundo lugar, se revisan las inconsistencias en la interfaz de la sala de espera del moderador. Para ello, se decidió cambiar la interfaz de usuario de dicha sala de espera por la versión anterior, pues la versión actual sólo era correcta para los jugadores. Como las pantalla de las dos salas de espera pasaron a ser muy diferentes, se hizo necesaria la división del componente que albergaba la funcionalidad de ambos en dos componentes diferenciados.

En cuanto a las historias de usuario asignadas para este sprint, hubo que modificar buena parte del código ya existente. Para maximizar la productividad, se decidió realizar en primer lugar la implementación de todas y cada una de las historias y, si sobraba tiempo en el sprint, realizar el testing de todas las historias después. También se estableció un orden para realizar la implementación, comenzando por la historia 3.5a, pasando después a la historia 3.5b y finalizando con la historia 3.7. Además, se acordó la realización de una presentación con el grupo GEEDS acerca del etado actual de la aplicación, con el objetivo de poder recibir sugerencias. Por este motivo se le dio prioridad absoluta a la implementación de las historias, para poder tener una versión en la que se pudiese notar la interacción entre jugadores.

Para la historia 3.5a no fue necesario realizar grandes modificaciones, aunque si que se decidió cambiar el método del back-end del que se leían los datos (el método **getRoundStatus** de **FormController**) pues con este método había que realizar menos llamadas al back-end que con la implementación original. En este método se detectó un pequeño error, ya que estaba marcado como un método GET, pero el protocolo HTTP no permite realizar peticiones GET con body, por lo que se decidió cambiar dicho método a POST. Para finalizar con esta historia de usuario, se decidió junto con el *Product Owner* la eliminación momentánea de la trama narrativa, pues se consideró que era un elemento poco prioritario y con las restricciones temporales incrementaba la complejidad de la solución a implementar. De todas formas, se dejó abierta la posibilidad de su implementación en un futuro.

En el caso de la historia 3.5b, el código que lo implementaba hubo que rehacerlo casi desde cero, pues apenas funcionaba en algún caso muy muy concreto. Se decidió que a esa pantalla se accediese a través de un icono en la pantalla de la historia anterior (al hacer click en un icono que indica la presencia de un conflicto entre las respuestas de los jugadores del grupo para esa pregunta). Para la parte de mostrar la pregunta y respuesta correspondiente a los datos indicados por el jugador anteriormente en el formulario, se decidió pasar el número de

pregunta y la letra de la opción marcada como parámetros de la consulta, pues esta opción era mucho más simple que la que estaba anteriormente implementada y que incluía el uso del almacenamiento local del navegador.

Para la historia 3.7, hubo que realizar una reorganización de código, que consistió en la eliminación de código duplicado. Además, hubo que modificar el sistema de cómputo de puntos, pues el que se venía utilizando hasta este momento había dejado de ser aplicable por la modificación de los requisitos (modelo de dominio en el Capítulo 4). Por la complejidad técnica de realizar la implementación de dicho sistema en el back-end y las necesidades de calendario con la presentación ante el grupo GEEDS, se realizó una primera implementación en el front-end y se añadió una tarea para el siguiente sprint en la que se abordase la migración del nuevo sistema de cálculo de puntos al back-end.

Tras la presentación, en la que se hizo una demo con los principales aspectos de la aplicación, se añadió una tarea al sprint actual mediante la que abordar las sugerencias que se recibieron a modo de feedback después de la presentación.

Como no quedó tiempo sufuciente en el sprint para realizar ninguna de las tareas de testing, se establece como "definición de hecho"(*definition of done* [2]) de este sprint el resultado de la demostración ante los stakeholders y la posterior integración de los cambios y sugerencias proporcionados por los mismos.

7.11. Sprint 9

Para este sprint se ha decidido abordar la implementación de las historias de usuario 3.8 y 3.5c. Además, se ha decidido revisar el estado en el que se encuentra la implementación realizada por el otro proyecto para la historia 3.2. Además, se ha decidido añadir una tarea por cada tarea de testing que se quedó sin realizar en el sprint anterior. La asignación de puntos de historia de este sprint se encuentra en la Tabla 7.16.

En la Tabla 7.17 se encuentra el resumen de tareas realizadas durante este sprint. Como aspectos a destacar de este sprint, habría que mencionar el cambio exitoso del sistema de puntos del front-end al back-end. También se detectó un bug en la sala de espera del moderador, el cual impedía empezar la partida con menos jugadores de los establecidos para el grupo (a pesar de que sí se satisfacía la condición de tener, al menos, un jugador cada grupo).

En relación a la implementación de los resultados finales, se decidió añadir una nueva operación en el back-end que realizase su lectura. Para ello fue necesario establecer los datos que necesitaría dicho método para llevar a cabo su función, así como el DTO que enviaría como respuesta. Por este motivo, se han definido dos DTOs uno llamado **FinalScoreRequest**, que incluye el código de la sala y el número del grupo del que se necesitan los resultados finales, y otro llamado **FinalResult** que incluye los datos asociados al resultado final del grupo. Se ha definido que este DTO incluya el código de las gráficas en las que dicho grupo obtuvo el mejor resultado en la puntuación de equilibrio tiene. Además, este DTO incluye los datos de los tres mejores grupos en cada uno de los cuatro parámetros de puntuación

HU	Tarea	T. estimado	T. empleado	Estado
-	Arreglar el despliegue en la máquina virtual y comprobar proxy en la máquina virtual	10h	4h 4m	No completado
-	Arreglar inconsistencias en la sala de espera de moderador	2h	3h 50m	Completado
-	Bugs y sugerencias surgidos durante la presentación con el grupo GEEDS	4h	3h 19m	Completado
3.5a	Implementación de comprobación de conflictos	6h	5h 39m	Completado
3.5a	Testing de comprobación de conflictos	4h	-	No iniciado
3.5b	Implementación de resolución de conflictos	5h	10h 5m	Completado
3.5b	Testing de resolución de conflictos	5h	-	No iniciado
3.7	Implementación ver resultados de ronda	6h	6h 16m	Completado
3.7	Cambiar el sistema de puntuación por uno nuevo	5h	4h 37m	Completado
3.7	Testing ver resultados de ronda	4h	-	No iniciado
Total		51h	37h 50m	6:10

Tabla 7.15: Seguimiento del sprint 8

que se establecieron en el análisis y la puntuación del mejor grupo (según la puntuación de equilibrio en cualquiera de las rondas), así como las respuestas que eligió el mejor grupo para obtener dichos resultados.

Para finalizar, la parte de testing se desarrolló sin mayor problema, dando por concluido este sprint.

7.12. Sprint 10

En este último sprint se decidió abordar la implementación del chat para los jugadores, mediante el desarrollo de las historias de usuario 3.3 y 3.4. De esta forma, se pretende dar por finalizada la creación de la primera versión jugable de Crossroads. Además, se propuso realizar una integración de un servicio desarrollado por otro TFG que se encarga de elaborar un texto de recomendación a los jugadores de un grupo a partir de la propuesta que han elaborado, por lo que se ha añadido una tarea al sprint para ello. Para finalizar, se ha añadido

Historia de usuario	Puntos de historia
3.8	3
3.5c	2
3.2	1

Tabla 7.16: Asignación de puntos de historia para el sprint backlog 9

HU	**Tarea**	**T. estimado**	**T. empleado**	**Estado**
-	Arreglar bug en la sala de espera del moderador	1h 30m	2h 10m	Completado
3.5c	Implementación comentarios en resolución de conflicto	5h	5h 12m	Completado
3.5a	Testing de comprobación de conflictos	4h	3h 44m	Completado
3.5b y c	Testing de resolución de conflictos	9h	2h 54m	Completado
3.7	Testing ver resultados de ronda	4h	2h 14m	Completado
3.7	Cambiar el sistema de puntuación nuevo del front-end al back-end	5h	4h	Completado
3.8	Implementación resultados finales	9h	13h 12m	Completado
3.8	Testing resultados finales	6h	2h 50m	Completado
3.2	Revisar el estado de responder preguntas	5h	3h 44m	Completado
Total		48h 30m	40h	9:9

Tabla 7.17: Seguimiento del sprint 9

alguna tarea de refactoring y corrección de bugs menores, así como una tarea en la que se pretende dar uniformidad a todo el estilo de la aplicación, pues había algunas pantallas que no habían sido actualizadas anteriormente. La relación de puntos de historia asignados para las historias de usuario se encuentra en la Tabla 7.18 y el detalle de las tareas realizadas en este sprint se presenta en la Tabla 7.19.

Historia de usuario	Puntos de historia
3.3	5
3.4	2

Tabla 7.18: Asignación de puntos de historia para el sprint backlog 10

Para realizar la implementación del chat, se decidió crear un componente reutilizable que fuese utilizado por el resto de componentes en los que hiciese falta el chat. Para ello, se creó un módulo que incluyese al nuevo componente y que realizase una exportación del mismo [7]. Se definió para el componente una interfaz a través de la cual, los otros componentes le podrían

indicar al chat la pregunta y respuesta que tenían seleccionada, para que el chat hiciese uso de esos valores como valor por defecto en la elaboración del mensaje. Con este objetivo en mente, se añadieron al componente dos atributos **Input** [6] que estarían vinculados a los identificadores de la pregunta y la respuesta marcadas. Aunque la primera lectura realizada al iniciar el componente se hacía de manera correcta, se detectó que este valor no cambiaba cuando lo hacía el valor en el componente padre del componente chat, por lo que hubo que investigar el por qué de este hecho. Al final, se encontró la solución en el ciclo de vida de los componentes angular, mediante la redefinición del método **ngOnChanges** del ciclo de vida [4]. De esta manera se solucionó el problema. Una vez definida la interfaz del componente, hubo que decidir el formato de los mensajes, que consistiría en un texto predefinido acompañando al texto introducido por los jugadores. Este texto incluiría una referencia a la pregunta y respuesta a la que se refieren, así como al sentido (A favor, en contra o neutro) de dicho mensaje. Además, se estableció que en los mensajes neutros el usuario no tendría que indicar la pregunta y la respuesta, sino que se haría uso de los valores por defecto.

Para llevar a cabo la integración del servicio que ofrece retroalimentación a los usuarios, hubo que añadir una operación al back-end. Dicha operación recibe un DTO que incluye las letras asociadas a las respuestas a los objetivos dadas por el grupo, así como las letras asociadas al resto de respuestas respuestas, realiza una petición a este servicio externo [12], y devuelve el resultado, en formato **String**. Una vez añadida esta operación en el back-end, solo hizo falta añadir el código encargado para realizar esta llamada en el front-end.

Para finalizar este sprint, se decidió probar una solución para el problema del CORS que fue ofrecida por el proyecto con el que se dejó de colaborar [65]. Dicha solución añade un filtro en el servidor que intercepta la respuesta a una petición y le añade unas cabeceras apropiadas para evitar que se produzca el fallo por CORS en el navegador. La solución ofrecida fue un éxito, terminándose así de arreglar el despliegue en la máquina virtual.

7.13. Resumen del seguimiento

Tras la finalización del proyecto, es momento de comparar el transcurso real del proyecto con la planificación realizada en el Capítulo 2.

El resumen de los trabajos realizados para este proyecto se encuentran en las Tablas 7.20 y 7.21. Como se puede observar, se ha mantenido la fechas de inicio con respecto a la planificación inicial y la fecha de fin del desarrollo coincide con la fecha de finalización prevista en la planificación, por lo que se ha producido un pequeño retraso. En cuanto a las horas totales que se estimó para llevar a cabo todos y cada uno de los sprints, superan las horas estimadas de trabajo en más de 140 horas. Afortunadamente, el total de horas de trabajo que han sido necesarias para finalizar el desarrollo del proyecto han sido menores, aunque esta cantidad supera a las 300 horas estimadas en la planificación. De estos datos se desprende que, en algunos sprints se realizó una planificación algo pesimista y que fue necesario realizar una sobreasignación de los recursos (en este caso, el único desarrollador) en algunos sprints para intentar cumplir el objetivo de 300 horas, aun sin poder cumplirlo finalmente.

HU	Tarea	T. estimado	T. empleado	Estado
-	Corrección error de CORS en la máquina virtual	30m	15m	Completado
-	Revisión de bugs y pequeños cambios en la aplicación	5h	7h 41m	Completado
-	Integración del servicio desarrollado por otro proyecto con la aplicación	5h	3h 51m	Completado
-	Integrar nueva versión del estilo de las pantallas	5h	12h 31m	Completado
3.3	Implementación chat grupal	15h	11h 48m	Completado
3.4	Implementación chat de sala	5h	3h 34m	Completado
3.3 y 3.4	Testing chat	15h	2h 3m	Completado
Total		50h 30m	41h 43m	7:7

Tabla 7.19: Seguimiento del sprint 10

Fecha inicio	Fecha fin desarrollo	Fecha fin memoria
19/01/2021	06/07/2021	17/07/2021

Tabla 7.20: Resumen de la calendarización del proyecto

Por todos estos datos que se desprenden de la comparativa, es necesario realizar un cálculo de los costes simulados del desarrollo del proyecto. Para calcular dichos costes se ha seguido el mismo procedimiento que para el cálculo del presupuesto (Capítulo 2). El resultado de dicho cómputo se presenta en la Tabla 7.22. La principal diferencia con el presupuesto reside en un incremento del coste laboral, derivado de la diferencia entre las horas de trabajo planificadas y las horas reales que ha llevado el desarrollo del proyecto.

En la Tabla 7.23 se muestra el coste real dado que no ha habido ningún coste laboral, y la máquina de despliegue ha sido una máquina virtual suministrada por la Escuela de Ingeniería Informática de la UVa, quedando solamente el coste de la amortización del ordenador del estudiante que realiza el TFG. Haciendo uso de las tablas de amortización [1], el coste de dicho equipo sería 800 * 25 % anual * 6/12 años de trabajo estimado, es decir, 100 €.

Para finalizar este resumen del seguimiento, es necesario realizar un apunte acerca del estado de completitud de las historias de usuario del *Product Backlog*. Aunque se han finalizado la mayor parte de las historias de usuario, aun quedan algunas historias de usuario poco prioritarias sin implementar. En concreto, estas historias hacen referencia a la adición de un chat para el moderador (historias 2.7 y 2.11), la capacidad del moderador para expulsar jugadores de un grupo (historia 2.12) y la gestión de la cuenta de un usuario registrado (historias 1.3, 1.4 y 1.5). El detalle de todas estas historias se encuentra en el *Product Backlog* final (Capítulo 2). Estas historias se abordarán en trabajos futuros.

Horas estimadas	**Horas desarrollo**	**Horas memoria**	**Tareas realizadas**
443h 20m	311h 23m	50h	77

Tabla 7.21: Relación de trabajo estimado y realizado para la finalización del proyecto

Tipo de coste	Coste estimado (€)
Laboral	6273
Material	801,55
Recursos	0
Total	7074,55

Tabla 7.22: Costes simulados del desarrollo del proyecto

Tipo de coste	Coste estimado (€)
Laboral	0
Material	100
Recursos	0
Total	100

Tabla 7.23: Costes reales del desarrollo del proyecto

Capítulo 8

Conclusiones y trabajo futuro

Una vez expuesto todos los aspectos concernientes al desarrollo de este proyecto, desde su fase inicial hasta la finalización del mismo, es momento de evaluar el cumplimiento de los objetivos marcados en el inicio.

Considero que se han cumplido todos los objetivos, pues no solo se ha desarrollado el front-end de la aplicación y se ha modificado el back-end, sino que se ha conseguido obtener una versión jugable del juego, con las principales características implementadas y con la adición de nuevas características no contempladas en el inicio y se ha desplegado dicha versión en una máquina virtual, por lo que ahora es accesible al público.

Además, en el plano personal, creo firmemente que este proyecto ha constituido una experiencia muy enriquecedora, pues no solo he adquirido amplios conocimientos técnicos en multitud de herramientas de uso muy común en la actualidad, sino que también he podido practicar algunas *Soft Skills*, como la capacidad de adaptación a los cambios o la capacidad de trabajo en equipo mediante la colaboración con otros proyectos. También considero que este proyecto ha sido todo un reto, por la gran complejidad de aprender a utilizar tantas herramientas desconocidas para mí en tan poco tiempo.

8.1. Líneas de trabajo futuras

A continuación, se realiza una propuesta de algunas líneas por las que se podría continuar el trabajo realizado en este proyecto:

- Desarrollo de las historias de usuario 1.3 a 1.5, 2.7, 2.11 y 2.12 que han quedado sin desarrollar (ver Sección 2.7).
- Activar la verificación del registro mediante correo electrónico en el back-end.

- Creación de un usuario con rol administrador que fuese el único capaz de realizar ciertas llamadas al back-end como, por ejemplo, las llamadas que realizan la población de la base de datos MongoDB.
- Adición de distintos personajes y habilidades de personaje para los miembros del grupo.
- Implementación de la asignación de los jugadores a los grupos de forma aleatoria.
- Mejora de la experiencia de usuario.
- Mejorar la robustez de la aplicación frente a fallos en la realización de peticiones HTTP.
- Implementar el registro en el back-end del número de cambios que realiza cada jugador en las respuestas y adición de este parámetro al cálculo de puntos.
- Añadir un registro en el back-end de las acciones que realiza el usuario y el tipo de las mismas.
- Modificar el mecanismo de reconexión de los moderadores para permitir el acceso a las salas de espera en el caso de partidas no iniciadas.
- Añadir un mecanismo de reconexión para los jugadores.
- Realizar pruebas de usabilidad con usuarios reales.

Bibliografía

[1] Agencia Tributaria. Tabla de coeficientes de amortización lineal. https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml. Accedido por última vez: 17/07/2021.

[2] Agile Alliance. Definition of done. https://www.agilealliance.org/glossary/definition-of-done. Accedido por última vez: 05/07/2021.

[3] Angular. Angular. https://angular.io/. Accedido por última vez: 19/01/2021.

[4] Angular. Lyfecycle hooks. https://angular.io/guide/lifecycle-hooks#onchanges. Accedido por última vez: 06/07/2021.

[5] Angular. Proxying to a backend server. https://angular.io/guide/build#proxying-to-a-backend-server. Accedido por última vez: 25/05/2021.

[6] Angular. Sharing data between child and parent directives and components. https://angular.io/guide/inputs-outputs. Accedido por última vez: 06/07/2021.

[7] Angular. Sharing sharing modules. https://angular.io/guide/sharing-ngmodules. Accedido por última vez: 06/07/2021.

[8] Angular. Tour of heroes app and tutorial. https://angular.io/tutorial. Accedido por última vez: 02/02/2021.

[9] Angular. Typeerror: Messageport was found in message but not listed in transferlist. https://github.com/angular/angular-cli/issues/20875. Accedido por última vez: 14/06/2021.

[10] Apache Software Foundation. Apache tomcat. http://tomcat.apache.org/. Accedido por última vez: 03/07/2021.

[11] Armin Ronacher. Flask. https://flask.palletsprojects.com/en/2.0.x/. Accedido por última vez: 10/07/2021.

[12] Baeldung. Do a simple http request in java. https://www.baeldung.com/java-http-request. Accedido por última vez: 06/07/2021.

[13] chartjs. Chart.js. `https://www.chartjs.org/`. Accedido por última vez: 03/07/2021.

[14] Cisco Systems. Webex by cisco. `https://www.webex.com/es/video-conferencing.html`. Accedido por última vez: 02/07/2021.

[15] Codegrip. Everything you need to know about code coverage. `https://www.codegrip.tech/productivity/everything-you-need-to-know-about-code-coverage/`. Accedido por última vez: 10/07/2021.

[16] Cypress. Code coverage. `https://docs.cypress.io/guides/tooling/code-coverage#E2E-code-coverage`. Accedido por última vez: 07/07/2021.

[17] Cypress. Cypress. `https://www.cypress.io/`. Accedido por última vez: 10/02/2021.

[18] Cypress. exec. `https://docs.cypress.io/api/commands/exec.html#Syntax`. Accedido por última vez: 21/02/2021.

[19] Cypress. Plugins. `https://docs.cypress.io/plugins/index.html`. Accedido por última vez: 24/02/2021.

[20] Damilare A. Adedoyin. Fixing cors errors with angular cli proxy. `https://levelup.gitconnected.com/fixing-cors-errors-with-angular-cli-proxy-e5e0ef143f85`. Accedido por última vez: 25/05/2021.

[21] David Álvarez Antelo and Iñigo Capellán-Pérez 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, 2019.

[22] Davide Bellone. How to run google chrome without cors. `https://www.code4it.dev/blog/run-google-chrome-without-cors`. Accedido por última vez: 25/06/2021.

[23] Docker Inc. Docker. `https://www.docker.com/`. Accedido por última vez: 10/07/2021.

[24] Douglas Wilson, fengmk2, Andrey Sidorov y otros. mysql. `https://github.com/mysqljs/mysql#readme`. Accedido por última vez: 24/02/2021.

[25] Eli Grey. Filesaver.js. `https://github.com/eligrey/FileSaver.js`. Accedido por última vez: 24/05/2021.

[26] Free Software Foundation. Gnu general public license v3. `https://www.gnu.org/licenses/gpl-3.0.html`. Accedido por última vez: 09/07/2021.

[27] GitLab Inc. Gitlab. `https://about.gitlab.com/`. Accedido por última vez: 02/07/2021.

[28] GitLab Inc. Gitlab ci/cd. `https://docs.gitlab.com/ee/ci/`. Accedido por última vez: 02/07/2021.

[29] Gustavo B. ¿qué es apache? descripción completa del servidor web apache. `https://www.hostinger.es/tutoriales/que-es-apache/`. Accedido por última vez: 03/07/2021.

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

[31] Ionos. Patrón observer: ¿en qué consiste este patrón de diseño? `https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-patron-observer/`. Accedido por última vez: 08/07/2021.

[32] Javainuse. Spring boot security example - refresh expired json web token. `https://www.javainuse.com/webseries/spring-security-jwt/chap7`. Accedido por última vez: 24/05/2021.

[33] Javier Santos Pascualena. ¿cuánto cuesta contratar un trabajador?).

[34] JetBrains. Intellij idea. `https://www.jetbrains.com/es-es/idea/`. Accedido por última vez: 03/07/2021.

[35] Ken Schwaber y Jeff Sutherland. La guía de scrum. `https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf`. Accedido por última vez: 14/02/2021.

[36] LOCOMOTION H2020. Low-carbon society: an enhanced modelling tool for the transition to sustainability. `https://www.locomotion-h2020.eu/about-project/overview/`. Accedido por última vez: 01/07/2021.

[37] Lucas Matías González Calderón. Diseño de la interacción y desarrollo del back-end de crossroads 2.0, un juego educativo para concienciar sobre el cambio climático. . Accedido por última vez: 10/07/2021.

[38] Martin Fowler. Pageobejct. `https://martinfowler.com/bliki/PageObject.html`. Accedido por última vez: 10/07/2021.

[39] Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson, 2008. Consultado por última vez: 09/07/2021.

[40] Microsoft. Patrón model-view-viewmodel. `https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm`. Accedido por última vez: 08/07/2021.

[41] Microsoft. Precios de máquinas virtuales linux. `https://azure.microsoft.com/es-es/pricing/details/virtual-machines/linux/`. Accedido por última vez: 17/07/2021.

[42] Microsoft. Typescript. `https://www.typescriptlang.org/`. Accedido por última vez: 03/07/2021.

[43] Microsoft. Visual studio code. `https://code.visualstudio.com/`. Accedido por última vez: 02/07/2021.

[44] MongoDB Inc. Mongodb. `https://www.mongodb.com/es`. Accedido por última vez: 03/07/2021.

[45] Mozilla. Control de acceso http (cors). `https://developer.mozilla.org/es/docs/Web/HTTP/CORS`. Accedido por última vez:20/02/2021.

[46] MvvmCross. Data binding. `https://www.mvvmcross.com/documentation/fundamentals/data-binding`. Accedido por última vez: 08/07/2021.

[47] Nacho Blanco. ¿qué patron usa angular? mvc o mvvm. `https://openwebinars.net/blog/que-patron-usa-angular-mvc-o-mvvm/`. Accedido por última vez: 08/07/2021.

[48] Naciones unidas. Cambio climático. `https://www.un.org/es/global-issues/climate-change`. Accedido por última vez: 10/07/2021.

[49] Object Management Group. What is uml. `https://www.uml.org/what-is-uml.htm`. Accedido por última vez: 02/07/2021.

[50] OpenJS Foundation. Node.js. `https://nodejs.org/es/`. Accedido por última vez: 02/07/2021.

[51] Oracle Corporation. Mysql. `https://www.mysql.com/`. Accedido por última vez: 03/07/2021.

[52] Osman Cea. Inyección de componentes y directivas en angular. `https://medium.com/angular-chile/inyecci%C3%B3n-de-componentes-y-directivas-en-angular-6ae75f64be66`. Accedido por última vez: 10/07/2021.

[53] Overleaf. Overleaf, online latex editor. `https://www.overleaf.com/`. Accedido por última vez: 02/07/2021.

[54] PrimeNG. The most powerful angular ui component library. `https://www.primefaces.org/primeng/`. Accedido por última vez: 25/05/2021.

[55] PrimeNG. Overlaypanel. `https://www.primefaces.org/primeng/showcase/#/overlaypanel`. Accedido por última vez: 25/05/2021.

[56] PrimeNG. Paginator. `https://www.primefaces.org/primeng/showcase/#/paginator`. Accedido por última vez: 25/05/2021.

[57] PrimeNG. Table. `https://www.primefaces.org/primeng/showcase/#/table`. Accedido por última vez: 25/05/2021.

[58] Programador Web Valencia. Testing end-to-end (e2e). `https://programadorwebvalencia.com/cursos/testing/e2e/`. Accedido por última vez: 03/07/2021.

[59] Project Management Institute. *A guide to the project management body of knowledge (PMBOK guide)*. Project Management Institute, 1987. Consultado por última vez: 22/05/2021.

[60] Rocket.Chat Technologies Corp. Rocket.chat - la plataforma de comunicación definitiva. `https://rocket.chat/es/`. Accedido por última vez: 02/07/2021.

[61] Rubén Pahino. ¿qué son spring framework y spring boot? tu primer programa java con este framework. https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx. Accedido por última vez: 03/07/2021.

[62] Shirivo. ¿modelo – vista – vista modelo? ¿eso es un patrón? https://shirivo.wordpress.com/2015/06/30/modelo-vista-vista-modelo-eso-es-un-patron/. Accedido por última vez: 08/07/2021.

[63] Stackoverflow. Angular material: Sasserror: Invalid css after "@include mat": expected 1 selector or at-rule, was ".core();". https://stackoverflow.com/questions/67622281/angular-material-sasserror-invalid-css-after-include-mat-expected-1-select. Accedido por última vez: 14/06/2021.

[64] Stackoverflow. Gitlab continuous integration npm background process. https://stackoverflow.com/questions/38634979/gitlab-continuous-integration-npm-background-process. Accedido por última vez: 24/05/2021.

[65] Stackoverflow. Spring boot cors filter - cors preflight channel did not succeed. https://stackoverflow.com/questions/36809528/spring-boot-cors-filter-cors-preflight-channel-did-not-succeed/36821971. Accedido por última vez: 06/07/2021.

[66] Techiediaries. Fixing cors issues in your front-end angular 7/8 app with angular cli proxy configuration. https://www.techiediaries.com/fix-cors-with-angular-cli-proxy-configuration/. Accedido por última vez: 25/05/2021.

[67] The LaTeX Project. Latex – a document preparation system. https://www.latex-project.org/. Accedido por última vez: 02/07/2021.

[68] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2019-2020 (Mención Ingeniería de Software). https://alojamientos.uva.es/guia_docente/uploads/2019/545/46976/1/Documento.pdf. Accedido por última vez: 02/02/2021.

[69] Vincent Driessen. A successful git branching model. https://nvie.com/posts/a-successful-git-branching-model/. Accedido por última vez: 02/07/2021.

[70] Visual Paradigm. Visual paradigm. https://www.visual-paradigm.com/. Accedido por última vez: 02/07/2021.

[71] W3Schools. Node.js mysql. https://www.w3schools.com/nodejs/nodejs_mysql.asp. Accedido por última vez: 24/02/2021.

[72] Wikipedia. Single-page application. https://es.wikipedia.org/wiki/Single-page_application. Accedido por última vez: 03/07/2021.

[73] Yanina Muradas. Qué es npm y para qué sirve. https://openwebinars.net/blog/que-es-node-package-manager/. Accedido por última vez: 02/07/2021.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

En esta sección se detallan los requisitos necesarios para desplegar y poner en marcha la aplicación de *Crossroads*.

A.1.1. Despliegue del back-end

Requisitos

Para poder poner en marcha el servidor de la aplicación, es necesario instalar las siguientes dependencias:

- **Java.** Serán necesarias dos versiones, la 8 y la 11, aunque la versión que debe estar activa es la 11.
- **MySQL Server.** Versión 5.7.3x. Además, durante la instalación (o al finalizarla) hay que establecer una contraseña para el usuario root. Dicha contraseña debe ser "admin".
- **MongoDB.** Instalar la versión 4.4.1. Como en el caso anterior, hay que establecer una contraseña para el usuario root (ésta vuelve a ser "admin").
- **Apache Tomcat.** Será necesario descargar la versión 9.0.24.
- **Maven.** En el caso de Maven, con instalar la última versión que esté disponible será suficiente.

Despliegue

Una vez satisfechas las dependencias, es momento de realizar el despliegue del servidor. Para ello, hay que seguir los siguientes pasos:

1. Crear en MySQL y MongoDB una base de datos llamada **crossroads**.
2. Ir a la carpeta en la que se ha clonado el proyecto y ejecutar el siguiente comando: `mvn spring-boot:run`. De esta forma, se habrá puesto en marcha la aplicación y se habrán creado en las bases de datos las tablas y colecciones correspondientes. Si se quiere ejecutar en segundo plano, el comando que habría que ejecutar sería `nohup mvn spring-boot:run &`
3. Poblar la base de datos MySQL con el archivo *seed.sql* que se encuentra en la carpeta **deploy** del proyecto.
4. Ejecutar las llamadas a la API **uploadPatterns** y **uploadGraphics**. Para ello, será necesario descargar los ficheros que se encuentran en Google Drive (Anexo B). La primera de las llamadas utiliza como body de la petición HTTP el contenido del fichero *patterns.json*, mientras que la segunda requiere enviar una petición con los ficheros *mutual_information.txt*, *description_scenario.txt*, *dataTemp.txt*, *dataPIB.txt* y *dataToPattern.csv* en el body de la misma. NOTA: para poder ejecutar dichas llamadas a la API, será necesario tener un usuario registrado y logeado en la aplicación y añadir en las cabeceras de la petición el token de autenticación que devuelve la petición de iniciar sesión en la respuesta.

Para simplificar los pasos 3 y 4, se han añadido a la carpeta deploy del proyecto dos scripts que realizan dichos pasos de forma automática (sólo en Linux). El primero de los scripts se llama *reset_db.sh* y se utiliza, en conjunto con el fichero *seed.sql* para borrar los datos de la base de datos MySQL (si los hubiere) y poblarla de nuevo con el script sql. El segundo script se llama `upload_patterns_and_graphics.py` y se utiliza para realizar las llamadas a la API del paso 4, sin necesidad de realizar inicio de sesión. Para ejecutar el script python que realiza el paso 4, es necesario copiar todos los ficheros de Google Drive en la propia carpeta deploy, ejecutar el comando `upload_patterns_and_graphics.py` y seleccionar la llamada a la API que quiere ejecutarse. Nota: se recomienda cambiar el orden de los pasos 3 y 4 en el caso de querer usar el script python, pues este script resetea la base de datos MySQL al hacer la petición de inicio de sesión.

A.1.2. Despliegue del front-end

Requisitos

Para desplegar el código del front-end, primero hay que instalar las siguientes dependencias:

- **Node.js.** Instalar la versión 12.20.2.
- **NPM.** Instalar la versión 6.14.11
- **Angular.** Instalar la versión 12.

Despliegue

Para poder realizar el despliegue, es necesario seguir los siguientes pasos:

1. Colocarse en la carpeta del proyecto recién clonado y ejecutar el comando `npm install`. Esto descargará todos los módulos de los que depende el front-end.
2. Para utilizar un servidor de desarrollo, ejecutar el comando `ng serve`. Este comando realiza la compilación del código y levanta un servidor en `http://localhost:4200`.
3. Si por el contrario, queremos utilizar un servidor para producción, ejecutar el comando `npm run build`. Este comando genera en la carpeta dist del proyecto una carpeta con todos los artefactos de la aplicación, desde el propio código hasta los ficheros de recursos.

A partir de este punto, sólo faltaria indicar al servidor donde están los artefactos de la página web. Como ese proceso depende en gran medida del servidor que se quiera utilizar, indicaremos solo los pasos que se han seguido para realizar el despliegue en un servidor Apache en Linux:

1. Colocarse en el directorio
   `/etc/apache2/sites-available`
   y editar el fichero
   `000-default.conf`.
   Añadir a ese fichero, dentro de las etiquetas *VirtualHost* el fragmento de código de la Figura A.1.
2. Copiar el fichero `.htaccess` de la carpeta deploy en la carpeta que ha generado el comando `npm run build` dentro de la carpeta dist.
3. Copiar la carpeta entera en el directorio `/var/www/html`.
4. Reiniciar el servidor Apache con el comando `sudo systemctl restart apache2`.

A.1.3. Despliegue del servicio de recomendación

Requisitos

Para desplegar este servicio, tan solo es necesario tener instalado docker y la versión 3.8 de python. El resto de dependencias se instalarán durante el proceso de creación de la imagen docker que contendrá este servicio.

```
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/crossroads-frontend
    <Directory "/var/www/html/crossroads-frontend">
            Options Indexes FollowSymLinks
            AllowOverride All
            Require all granted
    </Directory>


    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
```

Figura A.1: Fragmento de código del fichero 000-default.conf con las líneas que hay que añadir

Despliegue

Para poder levantar este servicio en una máquina Linux, es necesario seguir los siguientes pasos:

1. Descargar el código del servicio desde Google Drive (ver Anexo B), descomprimir la carpeta comprimida y situar el directorio actual en el directorio recién descomprimido.

2. Ejecutar el comando `sudo docker build --tag recomendador_docker .` para crear una imagen docker llamada `recomendador_docker`.

3. Una vez creada la imagen docker, utilizar el comando `sudo docker run -p 8000:5000 recomendador_docker`, donde `8000` es el puerto de la máquina al que se realizarán las peticiones y `5000` es el puerto en el que se ejecuta el servidor docker. Para ejecutar el comando anterior en segundo plano, hay que añadir la opción `-d`.

A.2. Manual de programación

A.2.1. Back-end

Para obtener una guía con el esquema de la API REST basta con poner en marcha la aplicación (comando `mvn spring-boot:run`) y visitar la URL `http://localhost:8080/swagger-ui.html#/`. Ahí se encuentra la documentación generada por Swagger acerca de la API. Contiene el esquema de todos los métodos de la API, indicando en cada caso los parámetros que necesita y la respuesta que devuelve cada método.

A.2.2. Front-end

Ejecución de los tests

Para ejecutar la batería de tests e2e de la aplicación es necesario colocar como directorio actual la carpeta del proyecto y ejecutar el comando `node_modules/.bin/cypress open`. Este comando abre una interfaz gráfica en la que aparecen todos los archivos que contienen casos de prueba de la aplicación. Si se hace click en cada uno de los ficheros, se ejecutan los tests asociados a ese fichero. También hay una opción para ejecutar todos los tests de todos los ficheros.

Si se quieren ejecutar los tests sin necesidad de una interfaz gráfica, ejecutar el comando `node_modules/.bin/cypress run`.

Para ver el análisis de la cobertura, hay que acceder al directorio `coverage/lcov-report` que Cypress ha creado en el directorio raíz del proyecto. El informe se encuentra en el archivo `index.html`.

A.3. Manual de usuario

En esta sección se presenta en detalle una guía de uso para los usuarios del sistema. La guía está dividida en dos partes, una destinada para el moderador y otra para los jugadores. El objetivo de esta guía es enseñar a los usuarios a jugar una partida.

A.3.1. Manual del moderador

En la Tabla A.1 se encuentra el detalle del funcionamiento de cada pantalla, así como la navegación que se produce entre ellas.

Página	Descripción
Home (Figura A.2)	Esta es la pantalla principal de la aplicación. Contiene tres enlaces que permiten la navegación a tres páginas distintas. El primero de ellos, realiza la navegación a la pantalla de unión de los jugadores a una sala (Figura A.11), el segundo de ellos navega a la pantalla de inicio de sesión (si el usuario no la ha iniciado. Figura A.4) o a la pantalla de creación de sala (si hay una sesión iniciada. Figura A.6) y el tercero de ellos navega a la pantalla de registro, es decir, a la Figura A.3. Además, contiene un botón en la esquina superior izquierda que permite la navegación a la pantalla de menú (Figura A.5) si la sesión está iniciada o a la propia pantalla Home si la sesión no ha sido iniciada. Este botón aparece en todas las pantallas del moderador, con el mismo comportamiento.

Registro (Figura A.3)	En esta pantalla el usuario tiene que introducir sus datos en el formulario para poder crear la cuenta. Al situar el cursor por encima de cada uno de los campos del formulario, aparece un mensaje con el formato de los datos que hay que introducir en dicho campo. En la parte inferior de la página, se encuentra un botón para enviar el formulario y un enlace para acceder a la pantalla de inicio de sesión (Figura A.4). Si el resultado del registro es correcto, se inicia la sesión automáticamente, mostrándose la página de menú (Figura A.5).
Inicio de sesión (Figura A.4)	Página de inicio de sesión. Al igual que en la pantalla de registro, el usuario tiene que rellenar un formulario con los datos de su cuenta. También contiene dos enlaces, uno para realizar la recuperación de los datos de la cuenta y otro para navegar a la pantalla de registro (Figura A.3). Si los datos de la cuenta coinciden con los de una cuenta registrada, se inicia la sesión y se muestra la pantalla de menú (Figura A.5).
Menú (Figura A.5)	Página que contiene todas las acciones que puede realizar un usuario fuera de la partida. A la hora de jugar una partida, la opción principal se encuentra en el botón "Crear sala", pues al hacer click en dicho botón, se inicia el proceso de creación de sala, mediante el mostrado de la página de la Figura A.6. Otras opciones importantes son "Buscar sala", que muestra un histórico de las salas que ha creado en usuario y la opción "Cerrar sesión".
Creación de sala (Figura A.6)	Página de creación de la sala. En ella, el moderador tiene que seleccionar el nombre de la sala, el número de rondas, el número de grupos y el tamaño del grupo. Al hacer click en el botón "Crear sala", se validarán los datos y se mostrará la sala de espera (Figura A.7). Si por el contrario, el moderador selecciona el botón "Cancelar", se volverá a la pantalla de menú.
Sala de espera del moderador (Figura A.7)	A través de esta página, el moderador puede ver los datos de la sala de partida que ha creado. Además, mediante los dos desplegables de la parte inferior, puede monitorizar los jugadores que van entrando en la sala y el grupo al que se van uniendo. Al hacer click en el botón de la parte inferior ("Empezar partida"), la aplicación comprueba que todos los grupos tengan, al menos, un jugador, dando comienzo así a la primera ronda de la partida y mostrando al moderador la sala de control de la partida.
Sala de control del moderador (Figura A.8)	A través de esta pantalla, el moderador puede controlar todos los aspectos de la partida. Todas las acciones que puede realizar el moderador se encuentran en los botones de esta página, aunque las opciones "EXPULSAR" y "PRESENTACIÓN" no están implementadas. El botón "SIGUIENTE RONDA" permite la finalización de la ronda actual y, si quedan más rondas por jugar, inicia una nueva ronda, mientras que el botón "TERMINAR", finaliza la partida.

Sala de control del moderador. Monitorización de grupo (Figura A.9)	Si en la sala de control seleccionamos alguno de los botones correspondientes a los grupos, el sistema mostrará en el cuadro central una tabla con el detalle de los datos de todos y cada uno de los jugadores pertenecientes al grupo que se acaba de seleccionar. El checkbox que aparece al final de cada fila es para seleccionar un jugador y realizar alguna acción sobre él, aunque de momento no hay ninguna implementada.
Sala de control del moderador. Monitorización de ronda (Figura A.10)	Si por el contrario, seleccionamos uno de los botones correspondientes a las rondas, el sistema mostrará en el cuadro central los datos relativos al estado de la ronda, así como una tabla en la que se muestra en estado de cada grupo durante esa ronda. La aplicación solo permite mostrar los datos de las rondas que se han iniciado, mostrando un mensaje de error en el caso en el que se quiera ver los datos de una ronda.

Tabla A.1: Tabla con el manual de usuario para el moderador

A.3.2. Manual del jugador

De manera similar al apartado anterior, se presenta una explicación del funcionamiento de cada pantalla en la Tabla A.2.

Página	**Descripción**
Home (Figura A.2)	Esta es la pantalla principal de la aplicación. Contiene tres enlaces que permiten la navegación a tres páginas distintas. El primero de ellos, realiza la navegación a la pantalla de unión de los jugadores a una sala (Figura A.11), el segundo de ellos navega a la pantalla de inicio de sesión (si el usuario no la ha iniciado. Figura A.4) o a la pantalla de creación de sala (si hay una sesión iniciada. Figura A.6) y el tercero de ellos navega a la pantalla de registro (Figura A.3). Además, contiene un botón en la esquina superior izquierda que permite la navegación a la pantalla de menú (Figura A.5) si la sesión está iniciada o a la propia pantalla Home si la sesión no ha sido iniciada. Este botón aparece en todas las pantallas del moderador, con el mismo comportamiento.
Unirse a sala (Figura A.11)	Página a través de la cual, el jugador se une a una sala. Para ello, tiene que introducir en el formulario el código de la sala y hacer click en el botón entrar. Si la sala está en espera, el jugador podrá entrar y se le mostrará un formulario para que introduzca sus datos de perfil (Figura A.12).

Perfil jugador (Figura A.12)	Formulario en el que el jugador tiene que introducir sus datos de perfil para la partida. Al igual que en el resto de formularios de la aplicación, se muestra un mensaje con el formato de la entrada de cada campo al pasar el cursor por encima del campo en cuestión. Una vez introducidos los datos del perfil, se mostrará una pantalla con información acerca del juego (Figura A.13).
Información sobre el juego (Figura A.13)	Pantalla de información sobre el juego. Al seleccionar la opción "Ya sé jugar" se cargará la sala de espera de los jugadores (Figura A.14).
Sala de espera del jugador (Figura A.14)	Esta es la sala de espera para los jugadores. En ella, cada jugador puede mirar los grupos que hay y los jugadores que hay en cada grupo mediante los despleglables. En cuanto al selector de personajes que se sitúa en la parte izquierda todavía está sin implementar, por lo que el jugador no tiene que realizar ninguna acción sobre él. Una vez seleccionado el grupo al que quiere unirse mediante el primer desplegable, el jugador tiene que hacer click en el botón "Unirse al grupo". Si el grupo tiene espacio de sobra, el jugador se habrá unido de forma adecuada, por lo que le tocará esperar a que el moderador inicie la partida, momento en el que se cargarán las preguntas que tendrá que contestar (Figura A.15).
Formulario de preguntas (Figura A.15)	Esta página incluye el formulario de preguntas que el jugador tiene que responder junto con su grupo para elaborar una propuesta. Al hacer click en el botón "Siguiente" para enviar la respuesta, se solicitará al jugador un argumento que sustente su elección. La página también permite navegar hacia atrás en el formulario mediante el botón "Anterior". Una vez respondidas todas las preguntas, el jugador accederá a una página en la que podrá ver las respuestas que ha seleccionado y las que han seleccionado el resto de miembros del grupo (Figura A.19)
Formulario de preguntas. Argumentación (Figura A.16)	Al enviar la respuesta a una pregunta del formulario, salta un pop-up que solicita la introducción de un argumento. Si no se introduce dicho argumento, la respuesta no se va a guardar.
Formulario de preguntas. Aparición del argumento en el chat (Figura A.17)	Al enviar un argumento para una respuesta, el argumento se añade a la lista de mensajes del chat de grupo para que el resto de miembros del grupo pueda verlo y facilitar así el debate.
Chat. Pop-up con los datos del mensaje (Figura A.18)	El chat también permite enviar mensajes sin tener que responder a una pregunta. Para ello hay que introducir el cuerpo del mensaje y darle al botón "Publicar". Como respuesta a dicha acción, el sistema mostrará por pantalla un pop-up en el que el usuario tendrá que seleccionar el tipo de mensaje y la pregunta y la respuesta a la que hace referencia el mensaje que acaba de introducir.

Comprobación de conflictos. No hay conflictos (Figura A.19)	En esta pantalla se muestra el estado del formulario de cada jugador, con las respuestas que ha seleccionado cada uno y la existencia o no de conflicto en cada pregunta. También se muestra el estado en el que se encuentra cada jugador en un momento dado. El jugador tendrá que esperar para ver los resultados de ronda (Figura A.22) o bien a que no haya conflictos, o bien a que el moderador finalice la ronda y se resuelvan los conflictos de forma automática.
Comprobación de conflictos. Hay conflicto (Figura A.20)	Si después de que los jugadores vayan introduciendo respuestas a las preguntas surge algún conflicto, dicho conflicto se mostrará en esta pantalla mediante el icono correspondiente. Para poder resolver el conflicto, el jugador puede hacer click en el icono de la pregunta con conflicto, lo que abrirá la página de resolución de conflictos para esa pregunta (Figura A.21)
Resolución de conflicto (Figura A.21)	En esta pantalla se muestra el formulario para la pregunta en la que se quiere resolver el conflicto. Además, se ofrece información acerca de los argumentos y los votos que ha recibido cada opción de la pregunta. Una vez modificada la respuesta (o cancelada la acción) se produce la navegación a la pantalla anterior.
Resultados de ronda (Figura A.22)	Página que muestra los resultados de la ronda para el grupo. Incluye las gráficas con la evolución del PIB y la temperatura como consecuencia de las medidas seleccionadas. También incluye un mensaje de recomendación para mejorar los resultados obtenidos y las puntuaciones que ha conseguido el grupo en esa ronda. El jugador permanecerá en esta pantalla hasta que el moderador inicie una nueva ronda o hasta que este finalice la partida, momento en el cual se mostrarán los resultados finales de la partida (Figura A.24).
Respuesta del grupo en Resultados de ronda (Figura A.23)	Si en la pantalla de resultados de ronda hacemos click en el icono de la lupa, se mostrará un pop-up que contiene las respuestas de la propuesta que ha enviado el grupo.
Resultados finales (Figura A.24)	En esta pantalla se muestran los resultados finales de la partida para el grupo. Incluye las gráficas de la ronda en la que el grupo obtuvo mejor puntuación en el baremo “Equilibrio”. También incluye un podio con los tres mejores grupos en cada una de las puntuaciones.
Gráficas del mejor grupo (Figura A.25)	Si el jugador quiere ver las gráficas correspondientes a los resultados finales del grupo que ha quedado mejor, basta con hacer click en el botón con forma de lupa en la pantalla de resultados finales. Esta acción mostrará un desplegable con las dos gráficas de dicho grupo.

Tabla A.2: Tabla con el manual de usuario para el jugador

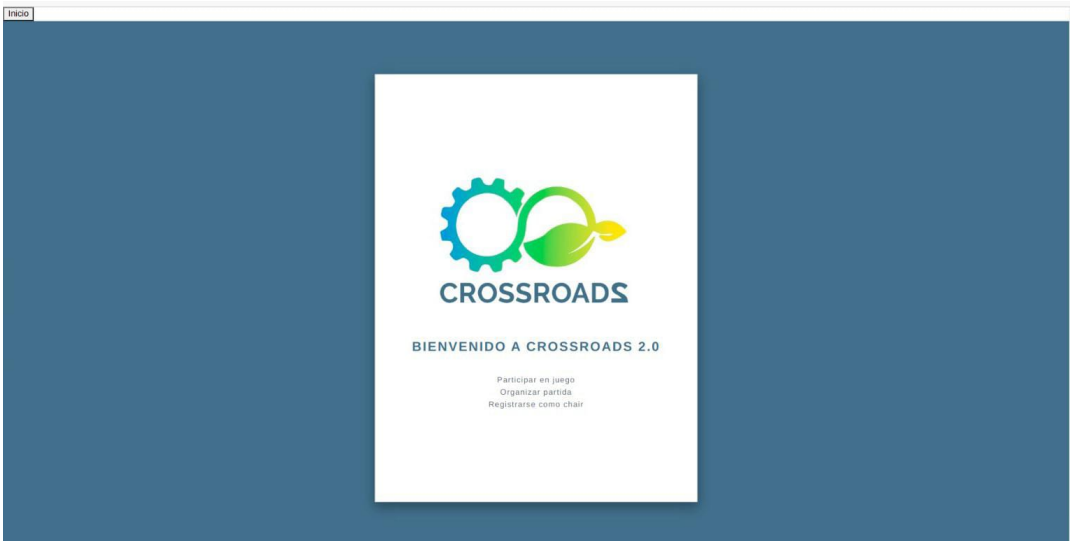

Figura A.2: Imagen de la pantalla principal de la aplicación

Figura A.3: Imagen de la pantalla de registro de la aplicación

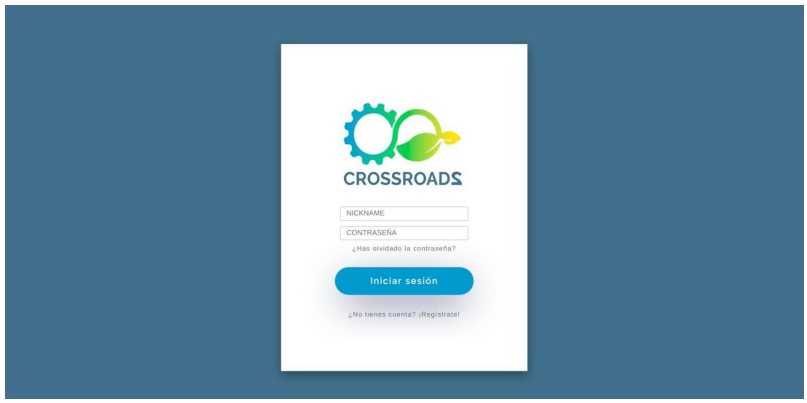

Figura A.4: Imagen de la pantalla de inicio de sesión de la aplicación

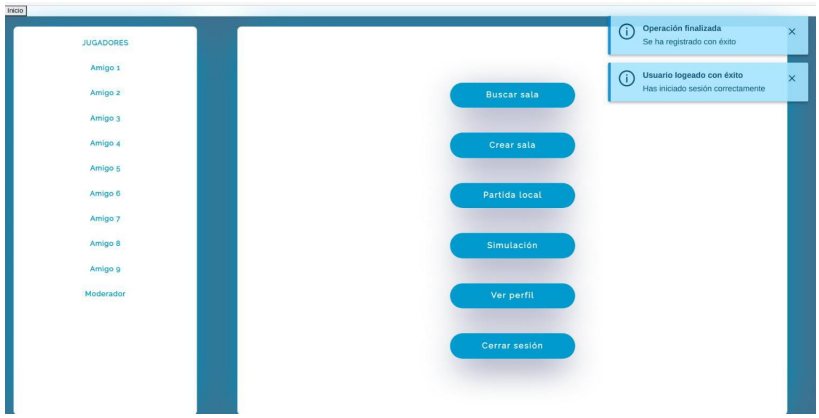

Figura A.5: Imagen del menú de la aplicación

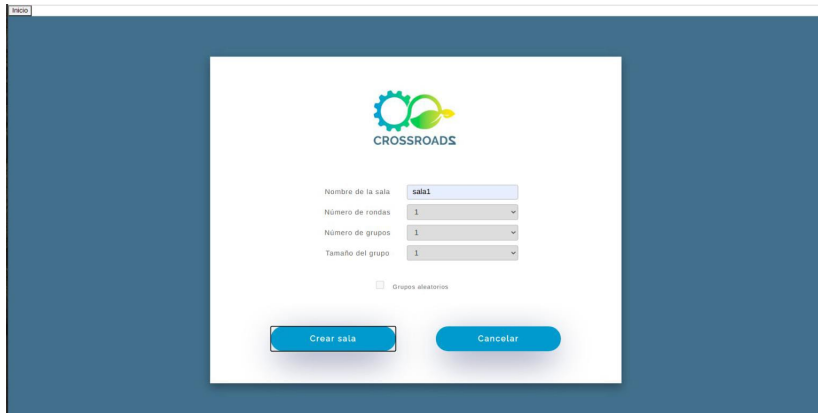

Figura A.6: Imagen del menú de la pantalla de creación de una sala

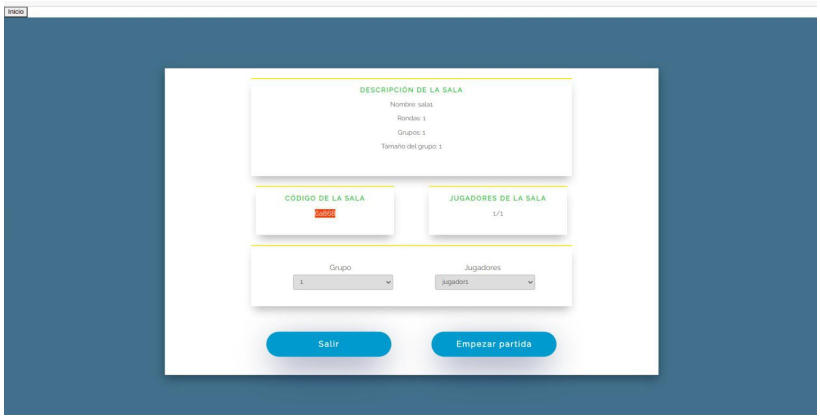

Figura A.7: Imagen del menú de la pantalla de creación de una sala

Figura A.8: Imagen de la pantalla de monitorización de la partida

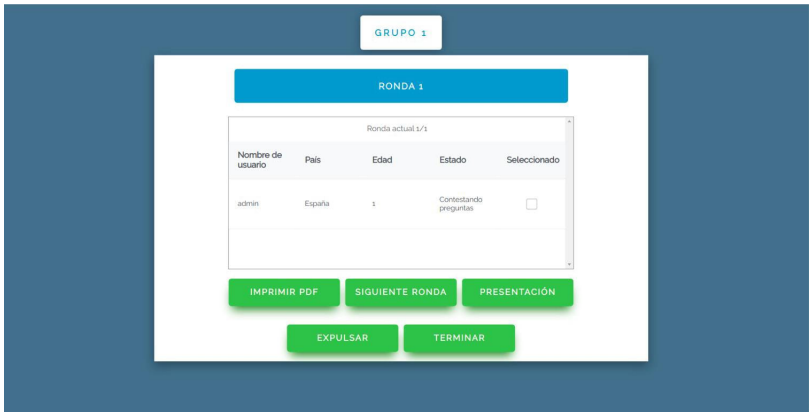

Figura A.9: Imagen de la pantalla de monitorización de la partida cuando el moderador selecciona al grupo 1

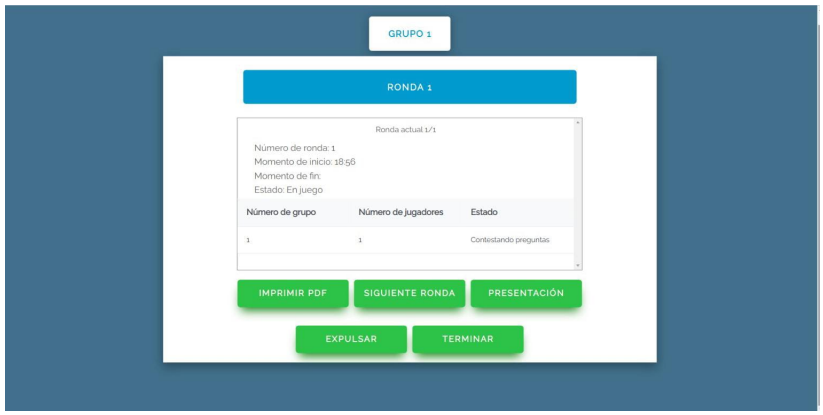

Figura A.10: Imagen de la pantalla de monitorización de la partida cuando el moderador selecciona la ronda 1

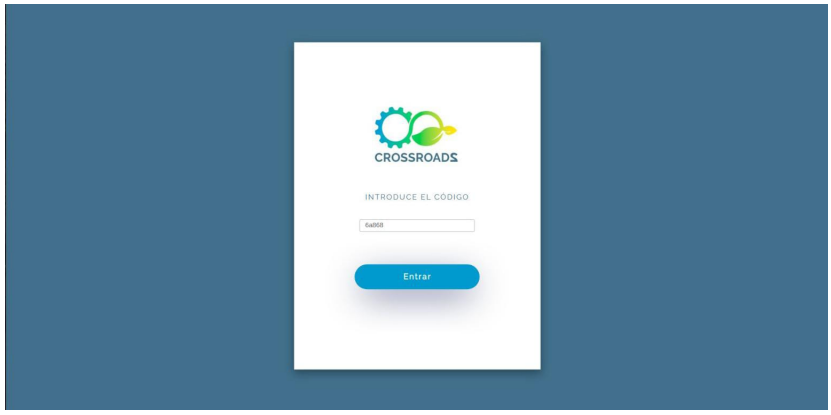

Figura A.11: Imagen de la pantalla de unión a la sala para los jugadores

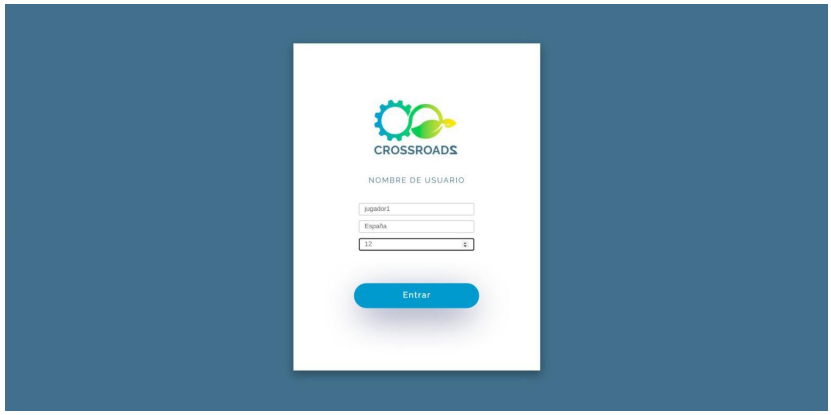

Figura A.12: Imagen de la pantalla en la que el jugador introduce sus datos

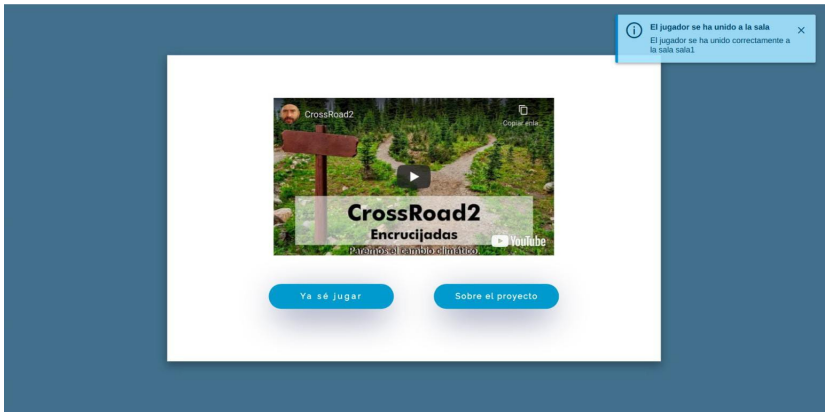

Figura A.13: Imagen de la pantalla de información sobre el juego para los jugadores

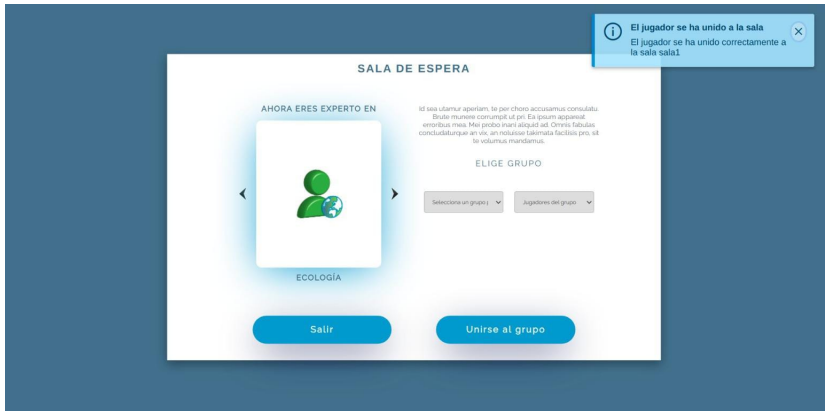

Figura A.14: Imagen de la sala de espera de los jugadores

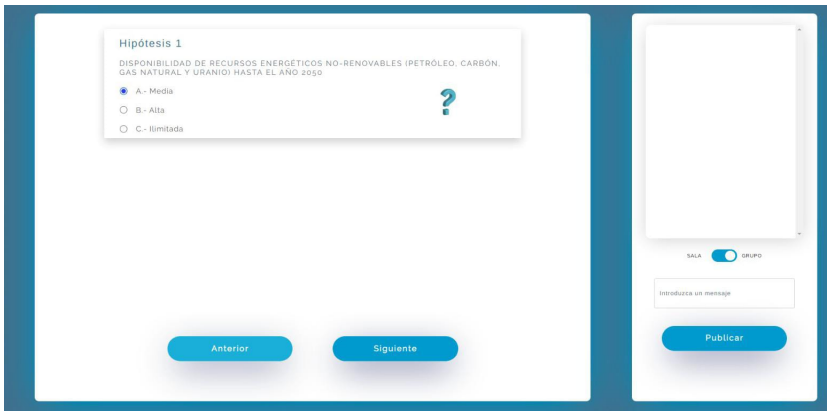

Figura A.15: Imagen de la pantalla de las preguntas que el jugador tiene que responder

Figura A.16: Imagen de la pantalla de preguntas con el pop-up que solicita el argumento

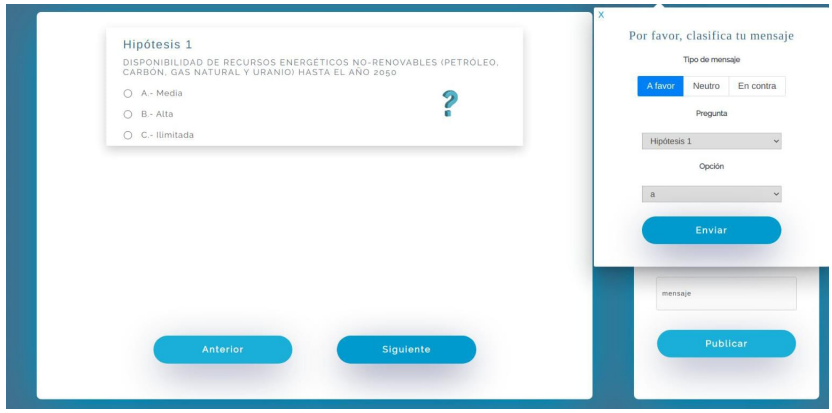

Figura A.17: Imagen de la pantalla de preguntas con el argumento de la primera pregunta volcado en el chat

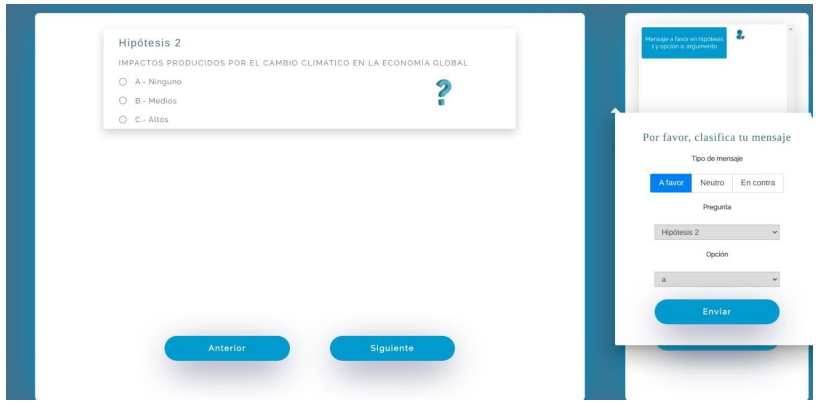

Figura A.18: Imagen de la pantalla de preguntas con el pop-up de la clasificación del mensaje

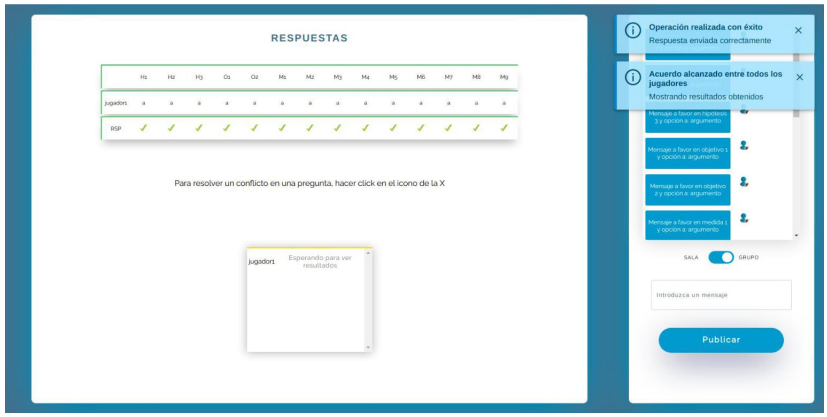

Figura A.19: Imagen de la pantalla de comprobación de conflictos (sin conflictos)

Figura A.20: Imagen de la pantalla de comprobación de conflictos (con conflicto en la pregunta 1)

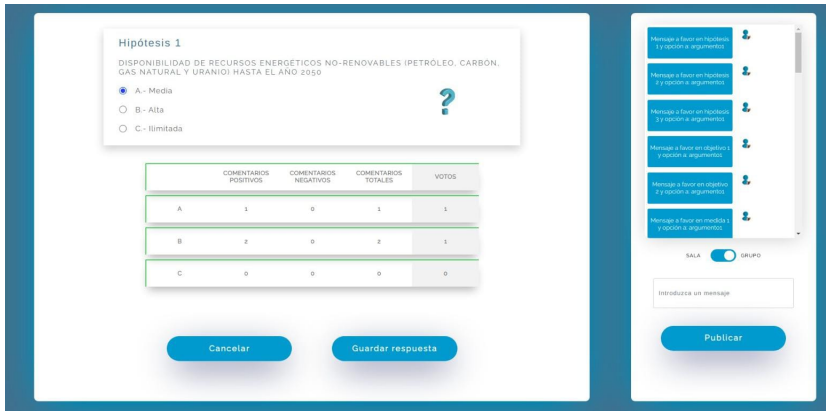

Figura A.21: Imagen de la pantalla de resolución de conflictos para la pregunta 1

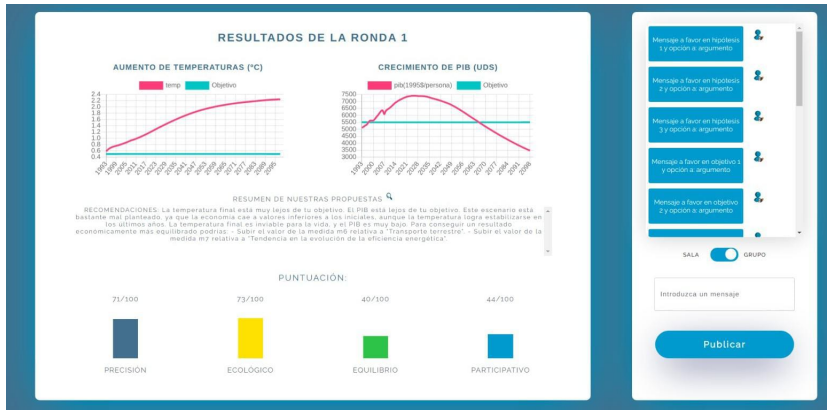

Figura A.22: Imagen de la pantalla de resultados para la ronda 1

Figura A.23: Imagen de la pantalla de resultados de ronda (con las respuestas del grupo)

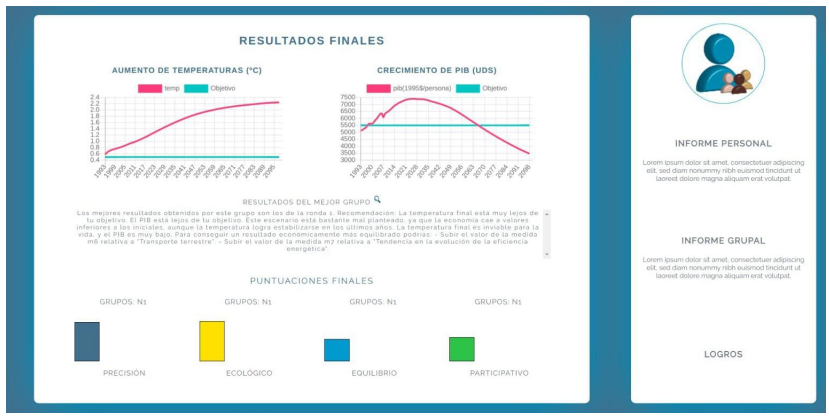

Figura A.24: Imagen de la pantalla de resultados finales

Figura A.25: Imagen de la pantalla de resultados finales con el pop-up de las gráficas del mejor grupo

Apéndice B

Resumen de enlaces adicionales

Los principales enlaces de interés de este proyecto son los siguientes:

- Repositorio con el código del back-end:
  `https://gitlab.inf.uva.es/manalda/crossroads-backend`
- Repositorio con el código del front-end:
  `https://gitlab.inf.uva.es/manalda/crossroads-frontend`.
- Ficheros para poblar la base de datos MongoDB:
  `https://drive.google.com/file/d/1xCL3O_hujNYx-HUSZuhWmrC6P1AxyVQ9/view`
- Código del servicio de recomendación:
  `https://gitlab.inf.uva.es/adrmanz/RecomendadorCrossRoads`