Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

DESARROLLO DE UNA APLICACIÓN ANDROID PARA CROSSROADS 2.0, UN JUEGO EDUCATIVO PARA CONCIENCIAR SOBRE EL CAMBIO CLIMÁTICO.

Alumno: David Crespo Ríos

Tutores: Yania Crespo González-Carvajal

David Escudero Mancebo

...

Agradecimientos

A mi familia, por apoyarme durante estos cuatro años de carrera y en especial durante estos últimos meses.

A mi novia, por estar siempre a mi lado cuando lo necesitaba.

A mis amigos, por apoyarme también y ayudarme a desconectar cuando lo he necesitado.

A mis tutores Yania y David, por ayudarme a realizar este proyecto y con todos los problemas que han surgido durante la realización del mismo.

A Manuel Alda y María Robles por resolver todas las dudas que he tenido durante el desarrollo del proyecto.

Al Consejo Social por la beca de colaboración en tareas de investigación de los departamentos.

Gracias a todos.

Resumen

El cambio climático es uno de los mayores problemas a los que nos enfrentamos como sociedad desde hace algunos años. Es necesario actuar cuanto antes, porque de lo contrario podríamos llegar a un punto de no retorno. Este trabajo de fin grado tiene como objetivo la implementación de una aplicación móvil educativa gamificada para intentar concienciar a la sociedad, principalmente a los alumnos de Educación Secundaria y Bachillerato, y observar las consecuencias ecológicas y económicas que podrían tener en el futuro la toma de diversas decisiones.

Este proyecto se ha desarrollado con el lenguaje de programación *Kotlin*, utilizando la arquitectura *Model View ViewModel (MVVM)*, aplicando una adaptación del marco de trabajo *Scrum* y como resultado se ha obtenido una aplicación que se encuentra disponible en la *Play Store* bajo el nombre de "Crossroad2". Para ello, se ha creado el *front-end* y se ha utilizado el *back-end* existente previamente desarrollado en otros proyectos.

Abstract

Climate change is one of the biggest problems we have been facing as a society for some years now. It is necessary to act as soon as possible, because otherwise we could reach a point of no return. This final degree project aims to implement a gamified educational mobile application to try to raise awareness in society, mainly to students of Secondary Education and High School, and observe the ecological and economic consequences that could have in the future the taking of various decisions.

This project has been developed with the Kotlin programming language, using the Model View ViewModel (MVVM) architecture, applying an adaptation of the Scrum framework. The result of that is an application called "Crossroad2", which is available in the Play Store. To get this, the front-end has been created and the existing back-end, previously developed in other projects, has been used.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

El cambio climático es uno de los principales asuntos a tratar en estos últimos años. Cada vez es más importante concienciar sobre este tema debido a la importancia que tiene de cara al futuro. Es por ello que cada vez se le da mayor relevancia en los medios de comunicación y se toman más medidas para intentar frenar el calentamiento global, como por ejemplo, la prohibición de la venta de coches de gasolina en Europa a partir de 2035. Muchos expertos han señalado el año 2030 como la fecha límite para evitar una catástrofe global y creen que es necesario fijar el incremento máximo de la temperatura global a 1,5ºC. Pese a ello, en los Acuerdos de París se pactó restringirlo a 2ºC y vamos camino de que el aumento sea de 3ºC [62].

Existen proyectos que intentan sensibilizar a la población sobre este problema con el objetivo de frenar esta tendencia alcista. Uno de estos proyectos es LOCOMOTION [45], que tiene como objetivo la concepción y desarrollo de un IAM (Modelos de Evaluación Integrados) para tratar de evaluar la efectividad, viabilidad, costos y ramificaciones de diferentes opciones políticas sobre el medio ambiente.

Este proyecto surge como complemento a dos Trabajos de Fin de Grado (TFG) anteriores [44] [46] que se encuentran dentro de LOCOMOTION. En estos TFGs se desarrolló una aplicación web educativa gamificada llamada Crossroads [28] que utiliza el IAM desarrollado en MEDEAS [40], un proyecto anterior a LOCOMOTION. Esta aplicación es colaborativo-competitiva, y en ella varios participantes colaboran en equipo para consensuar la toma de decisiones a la vez que compiten con otros equipos.

1.2. Motivación

La idea de este proyecto, es la creación de un juego educativo para dispositivos *Android*. Como parte de un proyecto financiado por la Fundación Española de Ciencia y Tecnología (FECYT) se propone hacer más fácil la utilización del juego para usuarios individuales. La aplicación web colaborativo-competitiva se convierte en una aplicación móvil con la que el usuario juega de forma individual. Compite solo contra sí mismo por mejorar sus indicadores tomando otras medidas y/o proponiéndose otros objetivos. La finalidad es concienciar acerca de las consecuencias que tendrían la toma de ciertas decisiones en la evolución del medio ambiente (mediante la temperatura) y el desarrollo sostenible (mediante el Producto Interior Bruto per cápita). La mayoría de las personas han oído hablar sobre el cambio climático, pero no son conscientes del impacto que pueden tener las medidas políticas en él. Mediante esta aplicación se puede descubrir el nivel de aumento de la temperatura global y hacia dónde se encaminaría la economía mundial, según el grado de ecológicas que sean las decisiones. Este juego va dirigido a los adolescentes, y principalmente a los alumnos de Secundaria y Bachillerato, para concienciar acerca del calentamiento global y de los sacrificios que deberíamos hacer para frenar esta catástrofe climática.

1.3. Crossroads

Crossroads es un juego educativo para sensibilizar a la población sobre el cambio climático. Consiste en responder una serie de preguntas entre las que hay hipótesis, objetivos y decisiones políticas y a partir de las respuestas a esas cuestiones se muestran unas gráficas y puntuaciones para observar lo que sucedería en el futuro si se actuara de esa manera. Las gráficas se consiguen a partir de una predicción obtenida de ejecuciones de un simulador que implementa los IAMs desarrollados en los proyectos MEDEAS y, posteriormente, en LOCOMOTION. Estas ejecuciones se realizan con un simulador llamado **Vensim** [73], que usa un indicador para representar el calentamiento global mediante el incremento de la temperatura, y otro para representar el bienestar de la población en un entorno de desarrollo sostenible mediante el Producto Interior Bruto (PIB) per cápita.

El juego educativo se apoya en modelos de simulación basados en dinámica de sistemas que para una serie de entradas (las respuestas a las preguntas) calculan una predicción de un conjunto de variables (las gráficas).

Crossroads 2.0

Para hacer el juego más dinámico se han ejecutado previamente todas las combinaciones de respuestas a las preguntas y se tienen almacenados todos los resultados para evitar que se ejecute el modelo en cada partida, lo que ralentizaría bastante el juego, ya que cada simulación tarda en calcularse entre 5 y 10 minutos.

Actualmente existe una versión web en la que se puede jugar por equipos de manera *online* contra otros equipos, y se está desarrollando la versión para *iOS* para jugar de manera

individual al igual que el juego que se desarrollará en este Trabajo de Fin de Grado para *Android*.

1.4. Objetivos

Los objetivos que se propone cumplir al finalizar este proyecto son:

- Conseguir la aplicación Crossroads 2.0 para *Android* como producto final.
- Subir la aplicación a la tienda de aplicaciones de *Google* para que pueda ser descargado en cualquier dispositivo *Android* con una versión compatible.
- Haber desarrollado un proyecto con todas las fases, desde la planificación del mismo hasta el lanzamiento de la aplicación.
- Practicar el desarrollo de un proyecto aplicando una adaptación del marco de trabajo *Scrum*.
- Mejorar mis habilidades como desarrollador de software, y en especial con el lenguaje *Kotlin*.

1.5. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: En este capítulo se presenta el proyecto, explicando su origen y que objetivos se pretenden alcanzar.

Capítulo 2 Requisitos y planificación: Describe el marco de trabajo *Scrum* de manera teórica y su adaptación al proyecto. También se exponen los requisitos a desarrollar, el plan de riesgos y el presupuesto del proyecto.

Capítulo 3 Tecnologías utilizadas: Se explican todas las tecnologías utilizadas durante el desarrollo del proyecto.

Capítulo 4 Análisis: En este capítulo se exponen el modelo de dominio y el proceso del juego así como la explicación de la obtención de las puntuaciones y las gráficas.

Capítulo 5 Diseño: Se presentan las decisiones de diseño llevadas a cabo, los patrones utilizados, el diseño de la interfaz de usuario y los diagramas que explican la arquitectura de la aplicación

Capítulo 6 Implementación y pruebas: Se exponen los principales cambios realizados durante la implementación, y las pruebas desarrolladas y resultados obtenidos para cada una de ellas.

Capítulo 7 Seguimiento del proyecto: En este capítulo se presentan los avances del proyecto, lo que se ha implementado en cada *Sprint*, las dificultades que han surgido y el tiempo empleado en cada tarea.

Capítulo 8 Conclusiones: Se explican las conclusiones finales, los objetivos que se han cumplido y algunas propuestas para realizar en el futuro.

Anexo A Manuales: Incluye manuales de mantenimiento, de instalación, despliegue, y de uso.

Anexo B Enlaces adicionales: Incluye enlaces de interés sobre el proyecto, como el repositorio de código, el enlace de *Play Store* para descargar la aplicación y los ficheros para poblar la base de datos MongoDB para desplegar el *backend* en local.

Capítulo 2

Requisitos y Planificación

En este capítulo se explicará el marco de trabajo utilizado, como se ha adaptado particularmente a este proyecto, el *Product Backlog* inicial y final, la planificación inicial, el análisis de los riesgos identificados y el plan de presupuesto simulado y real.

2.1. Scrum

Scrum es un marco de trabajo que permite la colaboración entre equipos diferentes. Aunque incorpora una serie de reuniones y herramientas que ayudan a los equipos a estructurar y gestionar su trabajo, se considera una metodología ágil, por lo que tiene que cumplir el **Manifiesto para el Desarrollo Ágil de Software** [43]:

- Individuos e interacciones sobre los procesos y las herramientas.
- Software funcionando sobre la documentación extensiva.
- Colaboración con el cliente sobre la negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

Scrum se basa en el empirismo, es decir, se fundamenta más en la observación y la experimentación que en una planificación profunda y detallada desde el inicio, que son más propias de metodologías no ágiles. Esto no quiere decir que no haya ningún tipo de control, ya que este se basa en la transparencia, inspección y adaptación [42]:

- **Transparencia**: Consiste en hacer visibles los aspectos más importantes del proyecto a todos los implicados en él. Para ello se realizan reuniones diarias y de revisión del *Sprint* (más adelante en la subsección 2.1.1 se define qué es un *Sprint* y sus eventos asociados).

- **Inspección**: Se deben revisar los artefactos que se generan mediante *Scrum* para revisar si existen variaciones no deseadas. Para ello se dispone de cuatro reuniones diferentes: Reunión de Planificación de Sprint, *Daily meeting*, Revisión del Sprint y la Retrospectiva del Sprint.
- **Adaptación**: Consiste en realizar cambios en los productos tan pronto como sea posible para corregir las desviaciones observadas mediante la inspección.

Scrum puede aplicarse a todo tipo de proyectos, pero no todos requieren de su uso. En el caso que nos atañe, se ha elegido *Scrum* para llevarlo a cabo, porque un enfoque incremental es lo más adecuado teniendo en cuenta las características explicadas a continuación de acuerdo con [23]:

- La **complejidad** es alta. Llevar a cabo un proyecto por una sola persona y además siendo la primera vez que el alumno tiene que realizar un proyecto software completo, supone una gran dificultad.
- La **incertidumbre** es baja debido a que ya se han realizado pequeñas aplicaciones por el alumno, por lo que hay conocimiento de las principales tecnologías a utilizar, y la que se va a desarrollar ya tiene su versión web, aunque tiene diferencias, por lo que la incertidumbre sobre el producto a desarrollar es baja. También es pequeña sobre los recursos, ya que el único desarrollador es el propio alumno, y tendrá disponibilidad suficiente para desarrollar la aplicación. Tal vez, donde más incertidumbre pudiera haber es en el proceso, aunque al estar guiado por la tutora y haber llevado a cabo una práctica en la asignatura de Planificación y Gestión de Proyectos con *Scrum*, no debería suponer ningún problema.
- Hay una **fecha de entrega** que cumplir, por lo que es preferible realizar incrementos para que aunque no de tiempo a completar todo el proyecto poder entregar la mayor parte de este, que una metodología *One Shot* como por ejemplo, cascada.

A continuación, se explicarán los elementos más importantes de *Scrum*.

2.1.1. Eventos

El *Sprint* engloba a los demás eventos. En cada evento hay que inspeccionar y, si es necesario, adaptar los artefactos generados, cumpliendo así con dos de los tres fundamentos explicados anteriormente. El tercero, la transparencia, se sigue al realizar estos eventos. También son usados para crear una regularidad y minimizar la necesidad de reuniones no definidas. Estos eventos son [42]:

- **Sprint**: Contiene al resto de eventos y su finalidad es obtener un incremento, es decir, un producto o entregable. Tiene una duración fija establecida al inicio del proyecto, que es el tiempo en el que se puede producir el artefacto deseado. Los requisitos o historias que se van a llevar a cabo, también permanecen fijos desde el inicio del *Sprint*.

- **Sprint Planning**: Reunión al comienzo del *Sprint* para planificar el trabajo que se realizará en él. Participan todos los miembros del equipo de trabajo, y eligen que requisitos del *Product Backlog* se van a llevar a cabo en esa etapa.
- **Daily Scrum**: Reunión al inicio de cada día de 15 minutos de duración a la que asiste solo el equipo de desarrollo. El objetivo es poner en común los avances del día anterior, lo que se planea realizar hoy, y los impedimentos que podrían surgir para llegar al objetivo del *Sprint*. También se mejora la comunicación entre los miembros del grupo, la toma rápida de decisiones, y elimina la necesidad de realizar otras reuniones, aunque pueden reunirse más veces durante el día para adaptar o replanificar el trabajo.
- **Sprint Review**: Reunión que se realiza al final del *Sprint*, donde se valora si se han cumplido los objetivos propuestos y se presenta a los interesados en el proyecto el incremento realizado. El *Product Owner* anota las valoraciones de los *Stakeholders* y si es necesario actualiza el *Product Backlog* para el siguiente *Sprint*.
- **Sprint Retrospective**: En esta reunión se hace un balance de lo sucedido en el *Sprint* completado, y se valoran posibles mejoras para los siguientes. Se realiza después del *Sprint Review* y antes del *Sprint Planning* de la próxima etapa.

2.1.2. Roles

Los equipos de *Scrum* suelen tener un tamaño reducido, normalmente un *Product Owner*, un *Scrum Master* y entre 3 y 9 desarrolladores [42]:

- **Scrum Master**: Es el responsable de gestionar que el proceso *Scrum* se lleve a cabo correctamente. También debe eliminar los posibles impedimentos que puedan poner en riesgo el desarrollo de los *Sprints*.
- **Product Owner**: Se encarga de maximizar el valor del producto y actúa como intermediario con los *stakeholders* y *sponsors* del proyecto. Si actúa como representante del negocio, su trabajo adicionalmente aporta valor al producto.
- **Equipo de Desarrollo**: Se encargan de desarrollar el producto, organizándose para entregar un incremento al final de cada *Sprint*.

2.1.3. Artefactos

Los **artefactos** representan trabajo o valor. Son el producto resultante del uso de *Scrum* y garantizan la transparencia y el registro de la información obtenida en el proceso. Estos artefactos son:

- **Product Backlog**: Es una lista ordenada de los requisitos que se desea implementar en el producto. Esta lista es el resultado del trabajo del *Product Owner* junto con el cliente y otros *stakeholders*. Los requisitos reciben el nombre de historias de usuario,

y estas pueden variar a lo largo del proyecto para adaptarse a las características que deseen los interesados en el proyecto. El objetivo de este artefacto es el *Product Goal*, que es la finalidad a largo plazo del equipo *Scrum*.

- **Sprint Backlog**: Es el conjunto de determinadas historias de usuario del *Product Backlog* que se planean realizar por el equipo de desarrollo en cada *Sprint*. El objetivo es el *Sprint Goal*, que marca el fin a realizar en cada *Sprint*, y es flexible en cuanto a trabajo necesario para conseguirlo se refiere.
- **Incremento**: Es el resultado de cada *Sprint* que se entrega al cliente, aportando valor de negocio al producto que se está desarrollando. Cada incremento realizado por el equipo de desarrollo se une a los anteriores y es un paso más para acercarse al *Product Goal*. El trabajo realizado no se puede considerar parte de un incremento hasta que cumpla con el objetivo de este artefacto, la *Definition of Done*, que es la descripción formal de las medidas de calidad requeridas para el producto.

En la Figura 2.1 se puede observar un breve resumen del proceso seguido en *Scrum*, desde que el *Product Owner* realiza las historias de usuario para crear el *Product Backlog*, hasta que el entregable cumple la *Definition of Done* para obtener el incremento.

Figura 2.1: Proceso de *Scrum* [18]

2.2. Adaptación de Scrum al proyecto

Para llevar a cabo este Trabajo de Fin de Grado, es necesario adaptar *Scrum*, principalmente por la diferencia de personal (alumno y tutora) comparado con un *Scrum Master*, un *Product Owner* y de 3 a 9 desarrolladores como sería habitual en un proyecto real que use este marco de trabajo.

En cuanto a los roles, el alumno asumirá la función de *Product Owner* y del equipo de desarrollo y la tutora se encargará de las labores del *Scrum Master*, revisando que todo el proceso se siga correctamente. Todas estas tareas han sido explicadas en la subsección 2.1.2.

La tutora también desarrollará en papel de cliente, ya que ha sido quien ha propuesto la idea del proyecto al alumno, y forma parte del equipo de *Crossroads*.

Debido al reparto de roles, todos los artefactos serán realizados por el alumno, dado que es el *Product Owner* quien realiza el *Product Backlog*, y es el equipo de desarrollo el que selecciona las historias de usuario recogidas en el *Sprint Backlog* y también lleva a cabo el incremento.

La duración de *Sprint* elegida es de dos semanas, como se explicará en la sección 2.5. Los eventos que se realizarán son los propios de cualquier proyecto que siga *Scrum*, con la salvedad de que no habrá *Daily Scrum* debido a que el equipo de desarrollo está formado por una sola persona y carece de sentido. En su lugar, se harán *Weekly Scrum*, ya que tendrán lugar una vez por semana, junto con la tutora, para compartir los avances del proyecto, realizar la inspección por si hubiera variaciones sobre el plan inicial, y adaptar en caso de que fuera necesario. Las otras tres reuniones, *Sprint Planning*, *Sprint Review* y *Sprint Retrospective* se producirán cada dos semanas, para finalizar el *Sprint* actual, y dar comienzo al siguiente. Todos los eventos se realizarán los jueves por la tarde.

2.3. Product Backlog inicial

La existencia de esta aplicación en versión Web, aunque aún en fase de pruebas y con ciertas diferencias como la posibilidad de jugar en modo multijugador, funcionalidad que no se implementará en *Android*, ha ayudado a obtener una primera versión de los requisitos que se desean desarrollar.

Como se ha comentado anteriormente, en *Scrum*, estos requisitos reciben el nombre de historias de usuario y tienen la forma:

"Como <*stakeholder*> quiero <*objetivo/comportamiento*> para <*motivo/razón/valor*>".

Los *Stakeholders* son todas las partes interesadas en el proyecto. Estos son:

- **Usuario final**: Aquel que quiere aprender sobre las consecuencias de las decisiones políticas en el cambio climático y en el desarrollo sostenible.
  El usuario final sería una persona acostumbrada a usar aplicaciones móviles, de cualquier edad y género, aunque preferentemente será un usuario joven. Estas personas tendrán que saber hablar español y tener curiosidad acerca de como pueden afectar las decisiones tomadas por los gobiernos al calentamiento global y al desarrollo sostenible.
- **Equipo de Scrum**
  - **Scrum Master**: La tutora.
  - **Product Owner**: El alumno.
  - **Equipo de desarrollo**: El alumno.
- **Cliente**: Equipo de *Crossroads*, representado por la tutora.

Inicialmente, las historias son demasiado generales y grandes como para desarrollarlas en un *Sprint* cada una, por lo que reciben el nombre de **épicas**, que son historias de usuario que se descomponen en otras más pequeñas para ser gestionadas con los principios y técnicas ágiles de estimación y seguimiento cercano [65].

A continuación, se muestra el *Product Backlog* inicial como épicas en la Tabla 2.1 y una subdivisión en historias en las Tablas 2.2 a 2.4.

Product Backlog Inicial	
	Épica
1	Como equipo de desarrollo quiero adaptar la aplicación al *backend* existente para integrar la *App Android* junto con la versión *Web* e *iOS*.
2	Como usuario final quiero recibir información para aprender a jugar correctamente.
3	Como usuario final quiero tomar una serie de medidas políticas para reducir el calentamiento global de manera sostenible.
4	Como usuario final quiero ver los resultados de las medidas tomadas para saber como afectarían al futuro en cuanto a calentamiento global y desarrollo sostenible.
5	Como usuario final quiero que la aplicación se ajuste a la pantalla del móvil para favorecer la usabilidad.
6	Como equipo de desarrollo quiero que la aplicación quede documentada para favorecer su mantenimiento en el futuro.

Tabla 2.1: Product Backlog Inicial

La definición de las épicas 1, 5 y 6, son similares a la definición de un requisito no funcional y se irán cubriendo mediante tareas al desarrollar el resto de historias de usuario.

Épica 2	
ID	**Historia**
2.1	Como usuario final quiero poder ver un vídeo explicativo para aprender a jugar.
2.2	Como usuario final quiero poder leer información sobre el proyecto para entender el contexto.
2.3	Como usuario final quiero poder diferenciar entre los diferentes tipos de cuestiones que serán hipótesis, objetivos y medidas políticas para entender los diferentes tipos de preguntas.
2.4	Como usuario final quiero ser advertido de las preguntas cuya respuesta no podré modificar para pensarlo más detenidamente.
2.5	Como usuario final quiero tener la opción de recibir ayuda en cada pregunta para poder tomar mejores decisiones.
2.6	Como usuario final quiero saber cómo se calculan los resultados para entender las puntuaciones obtenidas y mejorar en las siguientes rondas.
2.7	Como usuario final quiero recibir recomendaciones para poder mejorar mis resultados en la siguiente ronda.

Tabla 2.2: Historias de usuario de la Épica 2

Épica 3	
ID	**Historia**
3.1	Como usuario final quiero que la aplicación me permita marcar solamente una de las opciones disponibles para responder a cada pregunta.
3.2	Como usuario final quiero poder avanzar en las preguntas de la ronda actual para contestar a las siguientes cuestiones.
3.3	Como usuario final quiero poder retroceder en las preguntas de la ronda actual para revisar o modificar las respuestas.
3.4	Como usuario final quiero finalizar la ronda de preguntas cuando haya acabado para ver los resultados obtenidos.
3.5	Como usuario final quiero comprobar mis resultados al final de cada ronda para ver las consecuencias de las medidas tomadas.
3.6	Como usuario final quiero modificar las respuestas para avanzar a la siguiente ronda y poder mejorar los resultados.

Tabla 2.3: Historias de usuario de la Épica 3

Épica 4	
ID	**Historia**
4.1	Como usuario final quiero ver los resultados de cada ronda para obtener un *feedback* sobre las decisiones que he tomado en cada una de ellas.
4.2	Como usuario final quiero ver los resultados globales al acabar las 3 rondas para ver en qué turno lo he hecho mejor.

Tabla 2.4: Historias de usuario de la Épica 4

2.4. Product Backlog final

Durante el transcurso del proyecto el *Product Backlog* ha sufrido algunas modificaciones. Las más importantes han sido las siguientes:

- Introducción de la **Épica 7** y la **Historia de usuario 7.1** 2.9 debido al cambio de orden de las preguntas sufrido durante el desarrollo de la aplicación.
- División de la **Historia de Usuario 3.6** 2.7 en cuatro nuevas historias ya que abarcaba demasiada funcionalidad.
- Introducción de la **Historia de usuario 4.3** 2.8.

Product Backlog Final	
	Épica
1	Como equipo de desarrollo quiero adaptar la aplicación al *backend* existente para integrar la *App Android* junto con la versión *Web* e *iOS*.
2	Como usuario final quiero recibir información para aprender a jugar correctamente.
3	Como usuario final quiero tomar una serie de medidas políticas para reducir el calentamiento global de manera sostenible.
4	Como usuario final quiero ver los resultados de las medidas tomadas para saber como afectarían al futuro en cuanto a calentamiento global y desarrollo sostenible.
5	Como usuario final quiero que la aplicación se ajuste a la pantalla del móvil para favorecer la usabilidad.
6	Como equipo de desarrollo quiero que la aplicación quede documentada para favorecer su mantenimiento en el futuro.
7	Como equipo de desarrollo quiero diseñar e implementar una aplicación que facilite el mantenimiento.

Tabla 2.5: Product Backlog Final

2.5. Planificación inicial

Los *Sprints* tendrán una duración fija de dos semanas. Debido a la necesidad de compatibilizar el desarrollo de este proyecto con las prácticas de empresa, que empezarán a mediados del primer *Sprint*, las horas de trabajo semanales serán entre 15 y 20, por lo que a cada *Sprint* se dedicarán entre 30 y 40 horas, resultando unas 35 horas de media. Se planea realizar **8 *Sprints*** para llevar a cabo toda la funcionalidad más **2 *Sprints* extra** al final por si fuera necesario para completar las tareas que no hubiera dado tiempo a desarrollar en los 8 *Sprints* previstos.

Previamente ha sido necesario un *Sprint* 0, en el que se han empleado 33 horas en realizar toda la preparación necesaria, como se explicará en la Sección 7.2.1, antes de poder empezar a desarrollar el proyecto. Esta etapa fue de duración distinta a las demás, ya que solo abarcó 8 días en vez de 2 semanas.

Por lo tanto, la planificación inicial es de 280 horas para los 8 *Sprints* planeados, tomando 35 horas de media por etapa. Teniendo en cuenta las 33 horas empleadas para la realización del *Sprint* 0, el total son **313 horas**, ampliables a 345 o 380 horas, en caso de necesitar uno o ambos *Sprints* extra. Como se explica en la guía docente de la asignatura [71], el Trabajo de Fin de Grado tiene una carga de 12 ECTS, que corresponden a 300 horas por lo que se cumplen las horas requeridas.

Por último, se han escogido los puntos de historia como medida de estimación del esfuerzo que deberá realizar el equipo de desarrollo para desarrollar cada historia de usuario y poder así, establecer los *Sprint Backlogs*. Se utilizará una escala lineal del 1 al 10, siendo cada punto de historia equivalente a 5 horas de trabajo.

Épica 2	
ID	**Historia**
2.1	Como usuario final quiero poder ver un vídeo explicativo para aprender a jugar.
2.2	Como usuario final quiero poder leer información sobre el proyecto para entender el contexto.
2.3	Como usuario final quiero poder diferenciar entre los diferentes tipos de cuestiones que serán hipótesis, objetivos y medidas políticas para entender los diferentes tipos de preguntas.
2.4	Como usuario final quiero ser advertido de las preguntas cuya respuesta no podré modificar para pensarlo más detenidamente.
2.5	Como usuario final quiero tener la opción de recibir ayuda en cada pregunta para poder tomar mejores decisiones.
2.6	Como usuario final quiero saber como se calculan los resultados para entender las puntuaciones obtenidas y mejorar en las siguientes rondas.
2.7	Como usuario final quiero recibir recomendaciones para poder mejorar mis resultados en la siguiente ronda.

Tabla 2.6: Historias de usuario de la Épica 2

En la tabla 2.10 se muestra la planificación inicial:

Calendario de Sprints		
Sprint	**Fecha inicio**	**Fecha fin**
Sprint 0	02/02/2022	10/02/2022
Sprint 1	10/02/2022	24/02/2022
Sprint 2	24/02/2022	10/03/2022
Sprint 3	10/03/2022	24/03/2022
Sprint 4	24/03/2022	07/04/2022
Sprint 5	07/04/2022	21/04/2022
Sprint 6	21/04/2022	05/05/2022
Sprint 7	05/05/2022	19/05/2022
Sprint 8	19/05/2022	02/06/2022
Sprint extra 1	02/06/2022	16/06/2022
Sprint extra 2	16/06/2022	30/06/2022

Tabla 2.10: Planificación inicial de *Sprints*

2.6. Plan de riesgos

Dentro de la fase de planificación, uno de los puntos más importantes es la gestión y documentación de los posibles riesgos que se pueden manifestar en el proyecto. Siempre hay que tener en cuenta que pueden surgir eventos imprevistos, por ello hay que establecer mecanismos para reducir la posibilidad de que ocurra una amenaza, y procedimientos para que tengan el menor impacto posible en caso de que ocurran.

Épica 3	
ID	**Historia**
3.1	Como usuario final quiero que la aplicación me permita marcar solamente una de las opciones disponibles para responder a cada pregunta.
3.2	Como usuario final quiero poder avanzar en las preguntas de la ronda actual para contestar a las siguientes cuestiones.
3.3	Como usuario final quiero poder retroceder en las preguntas de la ronda actual para revisar o modificar las respuestas.
3.4	Como usuario final quiero finalizar la ronda de preguntas cuando haya acabado para ver los resultados obtenidos.
3.5	Como usuario final quiero comprobar mis resultados al final de cada ronda para ver las consecuencias de las medidas tomadas.
3.6	Como usuario final quiero ver el resumen de las respuestas dadas en la ronda anterior para poder modificar las que se deseen en la ronda actual.
3.7	Como usuario final quiero modificar las respuestas dadas en la ronda anterior poder mejorar los resultados en la ronda actual.
3.8	Como usuario final quiero avanzar a la siguiente ronda para poder volver a realizar el cuestionario.
3.9	Como usuario final quiero terminar la ronda actual desde la pantalla de resumen de las respuestas para avanzar a la ronda siguiente.

Tabla 2.7: Historias de usuario de la Épica 3

Un **riesgo** [23] es un acontecimiento que, si sucede, tiene un impacto positivo o negativo en los objetivos del proyecto. Estos involucran tanto la causa como el efecto de este. Es importante destacar que los riesgos están relacionados con potenciales contratiempos que se pueden ocasionar en el futuro, no con los problemas actuales.

Según el origen del riesgo, se pueden clasificar en cuatro categorías [23]:

- **Actores**: Todas las personas involucradas en el desarrollo del proyecto.
- **Tareas** en las que se divide un proyecto.
- **Estructura** de gestión y de sistema, incluyendo las que afectan a la planificación y control del proyecto.
- **Tecnologías** utilizadas en el desarrollo del proyecto.

Una vez identificados los riesgos, para documentarlos se analizarán los siguientes factores relativos a cada uno [23]:

- **Probabilidad**: Posibilidad de que se materialice el riesgo.
- **Impacto**: Consecuencias del riesgo en caso de materializarse. Pueden ser, por ejemplo, económicas, legales o reputacionales.

Épica 4	
ID	**Historia**
4.1	Como usuario final quiero ver los resultados de cada ronda para obtener un *feedback* sobre las decisiones que he tomado en cada una de ellas.
4.2	Como usuario final quiero ver los resultados globales al acabar las 3 rondas para ver en que turno lo he hecho mejor.
4.3	Como usuario final quiero ver las respuestas de cada ronda desde la pantalla de resultados finales para poder analizar comparativamente cómo he obtenido esas puntuaciones.

Tabla 2.8: Historias de usuario de la Épica 4

Épica 7	
ID	**Historia**
7.1	Como equipo de desarrollo quiero que sea fácil cambiar el orden de las preguntas.

Tabla 2.9: Historias de usuario de la Épica 7

Tanto la probabilidad como el impacto se miden en 3 niveles para poder clasificar y priorizar los riesgos a los que más expuestos se esté. Estos niveles son bajo, medio y alto.

También es necesario elaborar unos planes de actuación para estar preparados por si el riesgo sucede [23]:

- **Plan de mitigación**: Programa para que, si el riesgo ocurre, tenga menos impacto en el proyecto. Estas acciones tienen un coste extra en el proyecto independientemente de que el riesgo se materialice o no.
- **Plan de contingencia**: Acciones que se llevan a cabo si el riesgo sucede para intentar que su impacto sea el menor posible. Solo ocasionarán coste si se materializa.

Para analizar y priorizar los riesgos, se evaluará su posición en la matriz de impacto mostrada en la Figura 2.2. De esta forma se podrá conocer el estado de cada riesgo y monitorizarlos, aplicando los planes de mitigación y contingencia cuando sea necesario.

Si el nivel resultante del riesgo es **bajo**, su estado será abierto, es decir, existe la amenaza, pero no es necesario actuar. Si es **medio**, se debe aplicar el plan de mitigación ya que es posible que el riesgo se materialice y provoque un impacto no deseado. Y por último si el nivel es **alto**, hay que mitigar lo posible y preparar el plan de contingencia debido a su alta probabilidad de ocurrencia y su gran impacto.

En las Tablas 2.11 a 2.22 se exponen los riesgos analizados. Para cada uno de ellos se muestra título, descripción, probabilidad, impacto, plan de mitigación y plan de contingencia. Algunos de estos riesgos están inspirados en el Top 10 de riesgos en el desarrollo de Boehm [25] y [69].

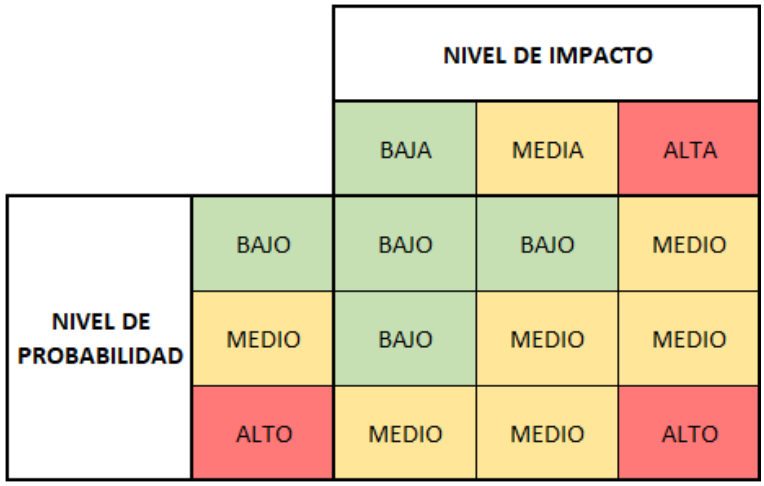

Figura 2.2: Matriz de Impacto (Elaboración propia)

Riesgo R01	
Título	Mala estimación del tiempo.
Descripción	Elaboración de una planificación temporal muy optimista que se traduce en un incremento del tiempo de realización de las tareas y el incumplimiento del desarrollo de todas las historias planeadas para el Sprint.
Probabilidad	Media
Impacto	Medio
Plan de Mitigación	▪ Tener en cuenta estimaciones de actividades anteriores. ▪ Sobrestimar o añadir un pequeño margen de tiempo para retrasos imprevistos. ▪ No incluir funcionalidad innecesaria en el Sprint.
Plan de Contingencia	▪ Reprogramar las tareas menos esenciales en Sprints posteriores.

Tabla 2.11: Riesgo R01: Mala estimación del tiempo

Riesgo R02	
Título	Miembro del equipo de desarrollo enfermo.
Descripción	Indisposición por parte del alumno de seguir realizando las actividades planeadas y por consiguiente, posible retraso en el Sprint.
Probabilidad	Media
Impacto	Medio
Plan de Mitigación	▪ Planificación de dos Sprints extra para llevar a cabo tareas que no se hayan podido realizar por enfermedad.
Plan de Contingencia	▪ Replanificación de las tareas afectadas para el siguiente Sprint. ▪ Si no se puede recuperar la demora en las siguientes etapas, usar los Sprints extra. ▪ Retraso de la fecha de finalización del proyecto.

Tabla 2.12: Riesgo R02: Miembro del equipo de desarrollo enfermo

Riesgo R03	
Título	Aumento de la pandemia y de las restricciones.
Descripción	Aumento de la incidencia de la pandemia por COVID-19 y endurecimiento de las restricciones sociales y de movilidad.
Probabilidad	Media
Impacto	Bajo
Plan de Mitigación	▪ Estar acostumbrado al uso de herramientas de comunicación telemáticas para el diálogo entre el alumno y la tutora.
Plan de Contingencia	▪ Uso de herramientas de comunicación telemática para poder proseguir con el correcto desarrollo del proyecto.

Tabla 2.13: Riesgo R03: Aumento de la pandemia y de las restricciones

Riesgo R04	
Título	Falta de conocimiento de las tecnologías a utilizar.
Descripción	Desconocimiento de las herramientas, programas y lenguajes a utilizar, como por ejemplo Latex, Overleaf, GitLab, Android Studio y Kotlin.
Probabilidad	Baja
Impacto	Alto
Plan de Mitigación	▪ Realización de tutoriales y cursos de aprendizaje para el manejo de las tecnologías que se desconozcan.
Plan de Contingencia	▪ Búsqueda de soluciones para cada problema concreto que aparezca durante la realización del proyecto. ▪ Sustituir estas herramientas por otras conocidas, como Word o Java.

Tabla 2.14: Riesgo R04: Falta de conocimiento de las tecnologías a utilizar

Riesgo R05	
Título	Falta de comunicación entre el alumno y la tutora.
Descripción	Escasez de reuniones de seguimiento del proyecto causando retrasos y errores en el proyecto.
Probabilidad	Baja
Impacto	Alto
Plan de Mitigación	▪ Uso del marco de trabajo *Scrum* para tener reuniones de seguimiento semanales y al inicio y al comienzo de los Sprints.
Plan de Contingencia	▪ Concertar reuniones fuera de planificación para un mayor entendimiento entre alumno y tutora. ▪ Planificar en Sprints posteriores las tareas que hayan sido retrasadas.

Tabla 2.15: Riesgo R05: Falta de comunicación entre el alumno y la tutora

Riesgo R06	
Título	Modificación de requisitos ya desarrollados.
Descripción	Una vez hecha la planificación inicial, el cliente modifica parte de la funcionalidad que ya está desarrollada.
Probabilidad	Media
Impacto	Medio
Plan de Mitigación	▪ Realizar reuniones con el cliente (la tutora), para intentar aclarar al máximo los requisitos que se van a desarrollar en cada *Sprint* y reducir el riesgo de que sean modificados más adelante.
Plan de Contingencia	▪ Priorizar las tareas más importantes, sacrificando si fuera necesario la funcionalidad residual, es decir, funcionalidad que requiere mucho trabajo y no será muy utilizada. ▪ Utilizar los Sprints extras reservados para imprevistos.

Tabla 2.16: Riesgo R06: Modificación de requisitos ya desarrollados

Riesgo R07	
Título	Requisitos no definidos correctamente.
Descripción	El *Backlog* inicial no ha sido revisado por todos los *stakeholders* y algunas historias no reflejan los intereses de los involucrados en el proyecto.
Probabilidad	Media
Impacto	Medio
Plan de Mitigación	▪ Revisión de los requisitos iniciales por parte de todos los stakeholders del proyecto.
Plan de Contingencia	▪ Modificación de aquellas historias mal especificadas y si habían sido desarrollados, adaptación de la funcionalidad a los nuevos requisitos. ▪ Replanificación de las tareas y si es necesario, uso de las etapas reservadas para imprevistos.

Tabla 2.17: Riesgo R07: Requisitos no definidos correctamente

Riesgo R08	
Título	Diseño inadecuado.
Descripción	Errores en la fase de diseño, que, debido a una falta de revisión, pasan desapercibidos, manifestándose en fases posteriores del proyecto.
Probabilidad	Baja
Impacto	Medio
Plan de Mitigación	▪ Revisar detenidamente el diseño antes de realizar la implementación para no retrasar la detección de errores a fases posteriores y tener que desechar trabajo ya realizado.
Plan de Contingencia	▪ Rehacer el diseño. ▪ Replanificar el proyecto.

Tabla 2.18: Riesgo R08: Diseño inadecuado

Riesgo R09	
Título	Desarrollo de la funcionalidad incorrecta.
Descripción	Los requisitos están bien definidos pero han sido mal entendidos y no se ha desarrollado la funcionalidad esperada.
Probabilidad	Baja
Impacto	Alto
Plan de Mitigación	▪ Asegurarse de que los requisitos han sido bien entendidos y, si es necesario, realizar una reunión con el cliente (la tutora) para aclarar las historias de usuario que sean necesarias.
Plan de Contingencia	▪ Corregir la funcionalidad errónea ▪ Replanificar las tareas retrasadas debido al desarrollo del requisito corregido.

Tabla 2.19: Riesgo R09: Desarrollo de la funcionalidad incorrecta

Riesgo R10	
Título	Desarrollo de la interfaz de usuario equivocada.
Descripción	La interfaz desarrollada carece de usabilidad para el usuario final y dificulta el correcto desarrollo del juego, disminuyendo así la experiencia de usuario.
Probabilidad	Baja
Impacto	Medio
Plan de Mitigación	▪ Seguir patrones de usabilidad. ▪ Realizar pruebas con posibles usuarios de la aplicación.
Plan de Contingencia	▪ Rediseñar la interfaz con los consejos dados por los usuarios que han realizado las pruebas.

Tabla 2.20: Riesgo R10: Desarrollo de la interfaz de usuario equivocada

Riesgo R11	
Título	Centrarse en funcionalidad compleja y poco útil.
Descripción	Implementación de requisitos que requieren mucho esfuerzo y no tienen mucha importancia para el cliente o apenas van a ser usados por el usuario final.
Probabilidad	Media
Impacto	Medio
Plan de Mitigación	▪ Priorizar las historias correctamente para que lo más necesario esté desarrollado y lo menos importante se implemente al final si hay tiempo para ello. ▪ Revisar en las reuniones con el cliente (la tutora) si la funcionalidad prevista para realizar en el siguiente Sprint es prioritaria.
Plan de Contingencia	▪ Finalizar la actividad lo más rápido posible y si es necesario dividir en varias subtareas para finalizar cuanto antes la actual y replanificar las demás tareas, aunque genera sobrecostes.

Tabla 2.21: Riesgo R11: Centrarse en funcionalidad compleja y poco útil

Riesgo R12	
Título	Fallo del ordenador.
Descripción	Reinstalación del BackEnd y todos los programas necesarios para realizar el proyecto por fallo del ordenador.
Probabilidad	baja
Impacto	Alto
Plan de Mitigación	▪ Realizar las actualizaciones cuando se esté seguro de que todos los nuevos progresos están guardados en la nube. ▪ Realizar las menores actualizaciones posibles hasta la finalización del proyecto.
Plan de Contingencia	▪ Reinstalar todos los programas necesarios, y tener una guía con los pasos necesarios para desplegar el *backend*, ya que es lo más costoso y complicado.

Tabla 2.22: Riesgo R12: Reinstalación del BackEnd por fallo del ordenador

2.7. Plan de presupuestos

Otro de los temas que se deben analizar en la fase de planificación antes de empezar a desarrollar el proyecto es la gestión de los costes. Debido a que este es un Trabajo de Fin de Grado y no es un proyecto real, no tiene todos los costes que se ocasionan en una empresa. Por ello, se presentarán dos presupuestos de manera estimada, que se compararán con los costes finales una vez concluido el proyecto en la Sección 7.3.3:

- **Presupuesto simulado**: Se analizarán los gastos ocasionados suponiendo que fuera un proyecto auténtico en una organización.
- **Presupuesto real**: Estimación de los costes adaptada al caso real que se está desarrollando.

2.7.1. Presupuesto simulado

A la hora de realizar un proyecto, hay diversos recursos que deben ser adquiridos y asignados a las tareas que hay que realizar. Los más importantes de todos ellos son los recursos de personal, entre los que podríamos destacar el gestor de proyecto y el Desarrollador *Android*. Después de analizar dos fuentes [68] [36] que realizan una media basándose en diferentes salarios obtenidos, y haciendo el promedio de los sueldos ofrecidos por estas, se puede estimar que la remuneración anual media de un *Project Manager* en España es de

38.500€. Al no especificar, se supone que es el salario bruto, porque la deducción del IRPF a aplicar sobre el sueldo, va en función de la situación personal y familiar de cada individuo.

En cuanto al *Product Owner*, la media resultante de las dos páginas examinadas [35] [38] es de 42.659€ brutos al año.

Por último, Desarrollador *Android*, haciendo la media obtenida consultando las mismas fuentes [67] [37], su sueldo sería de aproximadamente 31.004€ brutos. Según el Estatuto de Trabajadores, el máximo de horas laborales anuales en 1.826 horas, pero los convenios suelen establecer entre 1750 y 1800 horas. Suponiendo 1775 horas trabajadas al año, el Gestor del Proyecto cobra 21,69€/hora, el *Product Owner* 24,03€/hora y el Desarrollador *Android* 17,47€/hora. Como el Trabajo de Fin de Grado está planificado para ser realizado en 313 horas y 4 meses, siempre que no sea necesario usar los *Sprints* extra (Sección 2.5), el coste total del desarrollador sería de **5467,18€**. En cuanto a las horas empleadas por el *Project Manager* y el *Scrum Master* en el proyecto, habría que tener en cuenta la duración de las reuniones [1] y el número de Sprints. El *Daily Meeting* tiene una duración máxima de 15 minutos, la reunión de planificación del *Sprint*, es de máximo 2 horas por cada semana de duración del *Sprint*. Al tratarse de *Sprints* de 2 semanas, serían aproximadamente 4 horas por etapa. La reunión de revisión es de 1 hora por semana de *Sprint*, esto son 2 horas por *Sprint*. Y por último, la reunión de retrospectiva tiene una duración aproximada de 2 horas por etapa. En total, la extensión de estas reuniones es de aproximadamente 8 horas por *Sprint*, por lo que multiplicándolo por las 8 etapas planificadas, da un total de 64 horas y por tanto el coste del *Scrum Master* y el *Product Owner* serían de **1388,17€** y **1.538,13€** respectivamente a lo largo del proyecto.

El desarrollador necesitará equipamiento hardware y software para llevar a cabo su trabajo. Para mayor similitud con el proyecto real que se está desarrollando, se realizará el presupuesto de los mismos equipos para el contexto simulado y real. Se utilizará un *Lenovo ideapad 310* cuya vida útil es de aproximadamente cuatro años y un coste de 650€, por lo que serían 13,54€/mes. Como dispositivo móvil, se empleará un *Xiaomi Redmi Note 7*, con una duración estimada de dos años y un precio de 199€, es decir, 8,29€/mes. También se utilizará un ratón *Lenovo 300* para mayor comodidad del desarrollador con un precio de 9€ y una vida útil de un año, es decir, 0.75€/mes.

En cuanto al software a utilizar, será necesaria una licencia de ***Windows 10*** [51], con un precio de 145€. Como es un pago único, este gasto se incluirá dentro de los costes indirectos, ya que se tiene que repartir entre todos los proyectos que se desarrollen en la empresa.

También se necesitará adquirir la licencia de ***Astah Professional*** [21] para su uso. El precio de esta es de 54€ para 6 meses, tiempo suficiente para llevar a cabo el proyecto, lo que resultaría en 9€/mes si este durara 6 meses, pero como está previsto realizarlo en 4 meses, supone un gasto de 13,5€/mes.

Android Studio [7] no requiere de ninguna licencia para utilizarlo, pero si el objetivo final del proyecto es crear una aplicación y comercializarla, se necesitará adquirir la licencia de desarrollador de *Android* de *Google* para poder publicar el *Software* en la tienda de aplicaciones ***Play Store*** [39], lo que tiene un coste único de 25$, es decir, 24.39€. Este coste se repartiría entre todas las aplicaciones *Android* que se desarrollasen en la empresa, por lo que se incluirá en los costes indirectos.

Las bases de datos a utilizar son MySQL y MongoDB como se explicará en el Capítulo 3. Debido a que la licencia de **MySQL** consta de un único pago de 2.000€ que se repartiría entre todos los proyectos de la organización y **MongoDB** tiene una opción para aplicaciones pequeñas (hasta 1 TB de almacenamiento) en la que se paga solamente 0,30€ por cada millón de lecturas, ambas se incluirán dentro de los costes indirectos del proyecto.

También, se necesitaría adquirir un *Software* de control de versiones, en este caso se ha elegido ***GitLab*** [32], con un precio de 19$/mes, unos 17,70€/mes.

Para la documentación se ha utilizado el editor de LaTeX **Overleaf** [59], que tiene un coste de 5€ al mes.

Para realizar los bocetos de la aplicación, se ha utilizado la plataforma **Miro** [52], que incluye una opción de uso gratuito.

Por último, habrá que adquirir una herramienta para la comunicación telemática entre los integrantes del proyecto. Se utilizará ***Teams*** [50], cuyo precio más económico sería 5,10€/mes, correspondiente al paquete básico de *Microsoft 365* para empresas.

Por otro lado tenemos las sobrecargas o costes indirectos. Son constantes y se distribuyen entre todos los proyectos de la empresa y se estiman como un 15 % del presupuesto total de cada proyecto. Este dinero es destinado a pagar gastos como luz, agua, gas o Wi-Fi, además del alquiler de la oficina u otros departamentos como el legal o el de recursos humanos. También se incluirán aquí las licencias de *Windows* 10 y *Google Play Store* comentadas anteriormente.

En la tabla 2.23, se suman todos estos costes, y se muestra el presupuesto total simulado para llevar a cabo el proyecto:

Presupuesto simulado del proyecto			
Elemento	**Coste**	**Cantidad**	**Coste total**
Desarrollador Android	17,47€/hora	313 horas	5467,18€
Scrum Master	21,69€/hora	64 horas	1.388,17€
Product Owner	24,03€/hora	64 horas	1.538,13€
Ordenador	13,54€/mes	4 meses	54,16€
Smartphone	8,29€/mes	4 meses	33,16€
Ratón	0,75€/mes	4 meses	3€
Licencia Astah	13,5€/mes	4 meses	54€
GitLab	17,70€/mes	4 meses	70,80€
Overleaf	5€/mes	4 meses	20€
Teams	5,10€/mes	4 meses	20,40€
Subtotal			8.649€
Costes indirectos	15 % del proyecto		1.297,35€
Total			**9.946,35€**

Tabla 2.23: Presupuesto simulado del proyecto

2.7.2. Presupuesto real

En el caso real en el que se está trabajando, muchos de los costes se omiten debido a que todas las herramientas *Software* que se utilizan son gratis al disponer de licencias académicas. *Astah*, *Teams*, *GitLab* proporcionan concesiones gratuitas para los estudiantes. La única licencia de pago sería la de desarrollador de *Android* de *Google* para realizar publicaciones en *Play Store*.

Tampoco se producirán costes por sueldos de los empleados, ni por el alquiler de una oficina, ya que se desarrollará al completo desde casa. Por ello no se añadirán los costes indirectos, puesto que el único gasto que habría en este caso sería el de la electricidad, que no se tendrá en cuenta.

Los gastos por uso de **ordenador**, ***Smartphone*** y **ratón** son idénticos a los mencionados en el apartado anterior, por lo que el monto total es de **114,71€**, como se puede ver en la Tabla 2.24.

Presupuesto real del proyecto			
Elemento	**Coste/mes**	**Cantidad**	**Coste total**
Ordenador	13,54€/mes	4 meses	54,16€
Smartphone	8,29€/mes	4 meses	33,16€
Ratón	0,75€/mes	4 meses	3€
Licencia Google	-	-	24,39€
Total			**114,71€**

Tabla 2.24: Presupuesto real del proyecto

Capítulo 3

Tecnologías utilizadas

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

3.1. Android

Android [16] es un sistema operativo basado principalmente en Linux. Es un *software* de código abierto y gratuito. Es el más usado del mundo, con un 70.52 % de los usuarios móviles [60], es decir, aproximadamente, 2500 millones de usuarios.

El producto final del desarrollo con *Scrum* de este proyecto, será una aplicación *Android* para dispositivos con la **API 28** de *Android* o superior, es decir, móviles con el Sistema Operativo **Android 9.0 Pie** o superior. En junio de 2022, el 75 % de dispositivos *Android* son compatibles con esta versión, como se observa en la Figura 3.1. Se considera que esta API es lo suficientemente actual como para poder desarrollar la aplicación con la tecnología vigente, y abarca a más de 7 de cada 10 dispositivos, cifra que seguramente sea incluso mayor en los usuarios objetivos, ya que la gente joven suele actualizar sus móviles con mayor frecuencia debido a su gran uso día a día.

El entorno de desarrollo de *Android* es **Android Studio** [7] en la versión 2021.1.1, que es la más reciente en el momento de creación del proyecto.

3.2. Kotlin

En cuanto al lenguaje utilizado para desarrollar la aplicación, se ha elegido **Kotlin** [12] en la versión 1.6.10. Es un lenguaje creado por *JetBrains*, y cada vez es más utilizado para el

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99,9%
4.3 Jelly Bean	18	99,7%
4.4 KitKat	19	99,6%
5.0 Lollipop	21	98,6%
5.1 Lollipop	22	98,1%
6.0 Marshmallow	23	95,6%
7.0 Nougat	24	91,7%
7.1 Nougat	25	89,1%
8.0 Oreo	26	86,7%
8.1 Oreo	27	83,5%
9.0 Pie	28	75,1%
10. Q	29	58,9%
11. R	30	35,0%

Figura 3.1: Dispositivos compatibles con cada versión de *Android* [13]

desarrollo en *Android* con más de un 75 % de desarrolladores [12], por encima de *Java*, que era el lenguaje utilizado antes de la llegada del primero.

Se decidió usar *Kotlin* debido al conocimiento básico adquirido por el interés del alumno por las aplicaciones móviles previamente, para poder seguir ganando práctica en este lenguaje, y por ser utilizado también en las prácticas curriculares.

Para el desarrollo de esta aplicación se han decidido utilizar los componentes y librerías explicados en las siguientes subsecciones.

3.2.1. Fragments

Los **Fragments** [14] son elementos reutilizables para desarrollar la interfaz de usuario de la aplicación. En la Figura 3.2 se puede ver el ciclo de vida de los fragmentos, con sus estados

y métodos que se ejecutan en cada situación. Los fragmentos no pueden existir por si solos, si no que tienen que estar contenidos en una actividad o en otro fragmento. Una actividad puede contener a uno o más fragmentos.

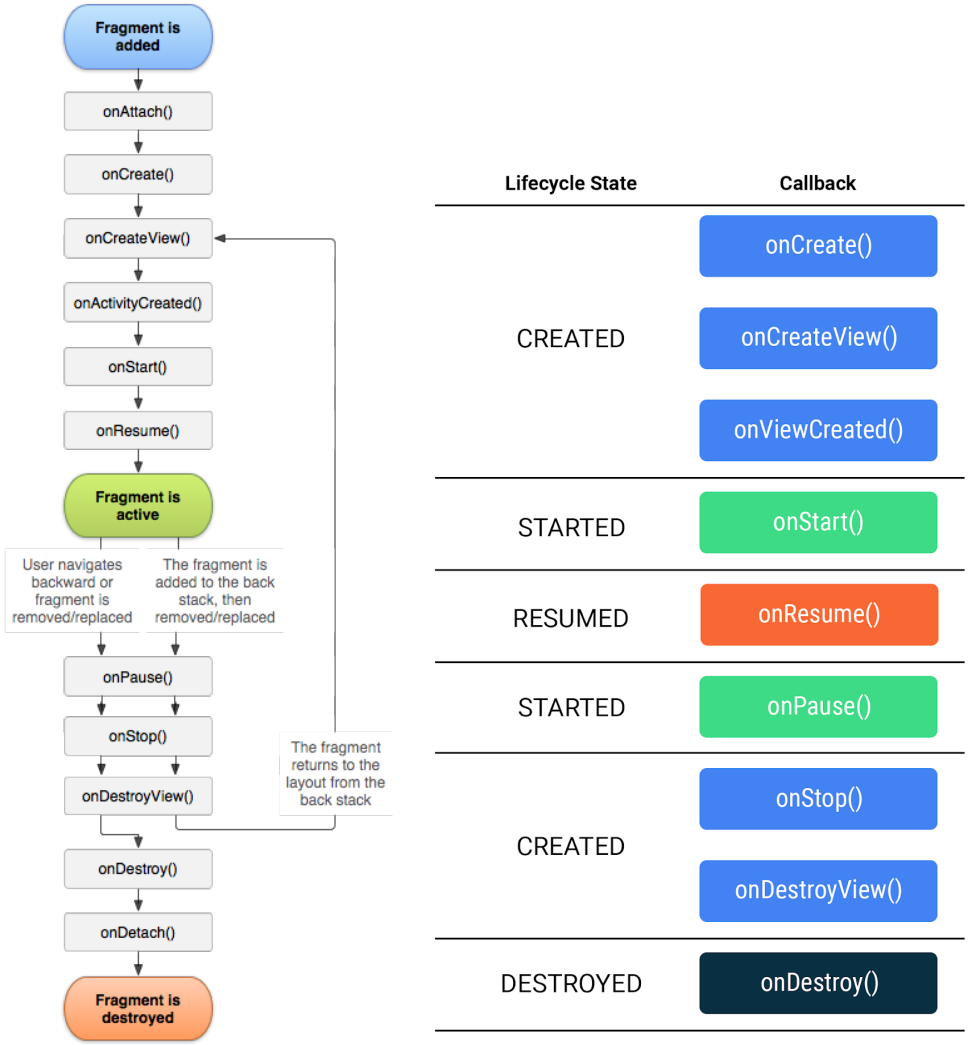

Figura 3.2: Ciclo de vida de un Fragmento [5, 14]

Cuando se navega hacia otro fragmento o actividad, el fragmento actual se coloca en la pila, se pasa por todos los métodos hasta *onDestroyView()*, y si posteriormente se vuelve a visualizar este fragmento, se ejecutan los métodos desde *onCreateView()* hasta *onResume()*.

3.2.2. Navigation

El componente **Navigation** [15] facilita la navegación entre diferentes fragmentos. El archivo *nav_graph.xml* contiene el **gráfico de navegación**, mediante el cual se puede ver de manera visual la navegación entre todos los fragmentos que conforman la aplicación (Figura 3.3).

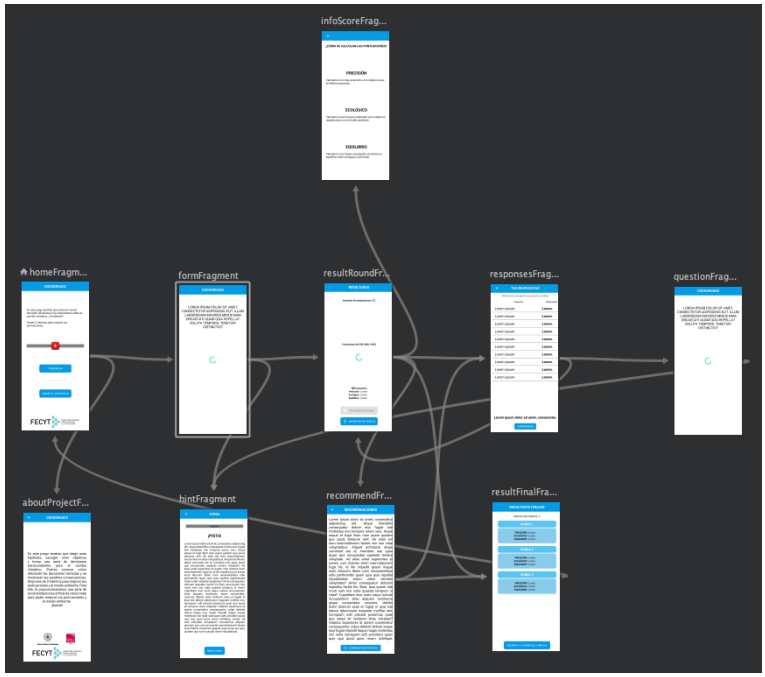

Figura 3.3: Gráfico de navegación (Elaboración propia)

Los elementos principales del gráfico de navegación son:

- **NavHost**: Es el contenedor que muestra los destinos de cada fragmento en el gráfico de navegación.
- **NavController**: Es el objeto que gestiona la navegación de un *NavHost*.

También permite el envío de datos entre destinos de forma segura mediante el complemento *SafeArgs* explicado en la siguiente subsección.

3.2.3. Safe Args

Safe Args [10] es un componente de *Navigation* que otorga seguridad de tipo al paso de datos entre destinos al navegar.

Al implementar este elemento, se crean diversas clases internas entre las que destacan:

- Una clase por cada destino o fragmento en el que se origina la acción con el nombre de la clase seguido de la palabra "Directions".
- Una clase por cada acción o navegación a realizar. Por defecto su nombre se forma de la siguiente manera:

  "action" + FragmentoOrigen + "To" + FragmentoDestino
- Una clase para el fragmento de recepción para recuperar los datos enviados a este destino cuyo nombre es el fragmento seguido de "Args".

3.2.4. Retrofit

Retrofit [66] es un cliente *REST* de tipo seguro para *Android* desarrollado por *Square*. Es una librería para realizar peticiones a APIs en *Android*. Facilita el consumo de *APIs* y la conversión de la respuesta al objeto de *Java* o *Kotlin* deseado mediante la factoría *Gson*. Se puede realizar cualquier tipo de petición, y también posibilita la adición de cabeceras.

3.3. Room

Room [9] es una librería de base de datos local proporcionada por la arquitectura de *Android*. Es una librería de persistencia que permite acceder a la base de datos mediante *SQLite*. Sirve para almacenar localmente los datos obtenidos de la base de datos remota mediante las peticiones a APIs, y ahorrar tiempo al no tener que repetir estas operaciones que son mucho más costosas que la obtención de estos datos del almacenamiento local.

En la Figura 3.4 se observan las relaciones entre los componentes de *Room*.

3.4. Advanced REST Client

Advanced REST Client [3] es un cliente REST que permite realizar llamadas a *APIs* para realizar pruebas. Mediante esta herramienta podemos enviar y recibir datos de la *API* de la misma manera que sucedería con la aplicación real.

3.5. Despliegue del Backend

Para el despliegue en local del *backend* existente previamente desarrollado por el equipo de *Crossroads*, fue necesaria la instalación de las siguientes tecnologías utilizadas para acceder

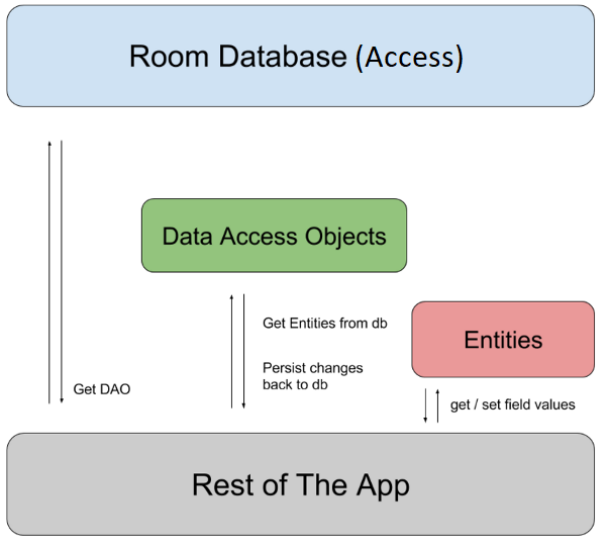

Figura 3.4: Relaciones entre los componentes de Room [9]

a toda la información almacenada de manera persistente y para poder introducir los nuevos datos generados. Este *backend* se utilizó durante el desarrollo de la aplicación antes de su lanzamiento final con el *backend* definitivo.

3.5.1. MySQL

MySQL [55] es un sistema de administración de base de datos SQL (Lenguaje de Consultas Estructurado) de código abierto, que ha sido desarrollado, distribuido y mantenido por *Oracle*. Dichas bases de datos son relacionales, es decir, cumplen el modelo relacional, cuyo objetivo es evitar la duplicidad de registros y garantizar la integridad referencial.

Esta base de datos guarda toda la información relativa a la partida, sala, jugador, moderador, respuestas etc., es decir, todos los datos menos los necesarios para construir las gráficas.

3.5.2. MongoDB

MongoDB [53] es un gestor de bases de datos NoSQL (No Solo SQL) orientado a documentos y de código abierto que almacena información de forma flexible y con un formato muy similar a *JSON* mediante un esquema dinámico.

Esta base de datos es utilizada por el *backend* para almacenar la información de las simulaciones de las predicciones de la temperatura y el Producto Interior Bruto (PIB) para realizar las gráficas, como se explica en la subsección 4.4.

3.5.3. Maven

Maven [49] es un *Software* de gestión de proyectos que permite compilar, realizar informes y documentación de un proyecto. En este caso, se ha utilizado para compilar el *backend*.

3.5.4. Tomcat

Apache Tomcat [19] es un contenedor de servlets con el objetivo de manipular proyectos WAR (Archivo Web). En este proyecto se utiliza como servidor para desplegar el *backend*.

3.5.5. Java

Java [57] es el lenguaje de programación utilizado para desarrollar el *backend* con ayuda del *framework Spring Boot*. Para desplegarlo fue necesario descargar la versión 11 de Java.

3.6. Herramientas para la gestión del proyecto

3.6.1. GitLab

GitLab [32] es la herramienta elegida para gestionar y realizar diversas tareas del proyecto y del repositorio. Es una plataforma *DevOps*, lo que significa que proporciona a los desarrolladores la posibilidad de realizar operaciones con el fin de acelerar el desarrollo ágil del proyecto.

Los principios de DevOps son [34]:

- Automatización del ciclo de vida del desarrollo *Software*.
- Colaboración y comunicación.
- Mejora continua y reducción del gasto (tiempo y dinero).
- Enfoque en las necesidades del usuario con bucles de retroalimentación cortos.

En cuanto a las labores llevadas a cabo en *GitLab*, se encuentran el control de versiones y el tablero de *Scrum*, llamado *Issue Board*, que se explicarán en las siguientes subsecciones.

3.6.2. Git

Git [31] es un sistema de control de versiones distribuido, gratuito y de código abierto diseñado para gestionar proyectos, de tal forma que se pueda tener un seguimiento del progreso realizado. También incluye la posibilidad de dividir el proyecto en **ramas**, para avanzar a partir de la rama principal (normalmente llamada *master*) o a partir de otra sub-rama, y continuar el desarrollo sin afectar al código funcional y estable.

En este proyecto se sigue la metodología de **rama por tarea**, es decir, por cada tarea a desarrollar se crea una nueva rama que, cuando esté finalizada se unirá a la que contiene la versión estable del código, en este caso la rama *develop*. La diferencia entre *master* y *develop* es que esta última contiene el código de tareas finalizadas, pero la aplicación no es completamente funcional, mientras que *master* solamente contiene el código que esté completamente finalizado para que la aplicación pueda ser utilizada.

3.6.3. Issue Board

GitLab Issue Board [33] es una herramienta para la gestión de proyectos software utilizada para organizar el seguimiento de las tareas a realizar de manera visual. Consiste en una disposición de columnas personalizable, en las que se puede añadir tantas como se desee. Para el seguimiento de este proyecto se utilizarán las columnas mostradas en la Figura 3.5.

3.7. Herramientas para análisis, diseño y documentación

3.7.1. Astah

Astah [21] es una herramienta de modelado UML (Lenguaje de Modelado Unificado) creado por la compañía *Change Vision*. Ha sido utilizada para desarrollar los diagramas necesarios de la fase de diseño 5.

3.7.2. Visual Paradigm

Visual Paradigm [74] es una herramienta de modelado UML para la gestión y diseño de sistemas como *Astah*, pero con más posibilidades a la hora de realizar diagramas de clases, ya que permiten las relaciones ternarias. Se utilizó principalmente por esta razón, aunque finalmente no ha sido necesario usar este tipo de relaciones.

Se ha empleado para realizar el modelo de dominio y la máquina de estados de la fase de análisis 4.

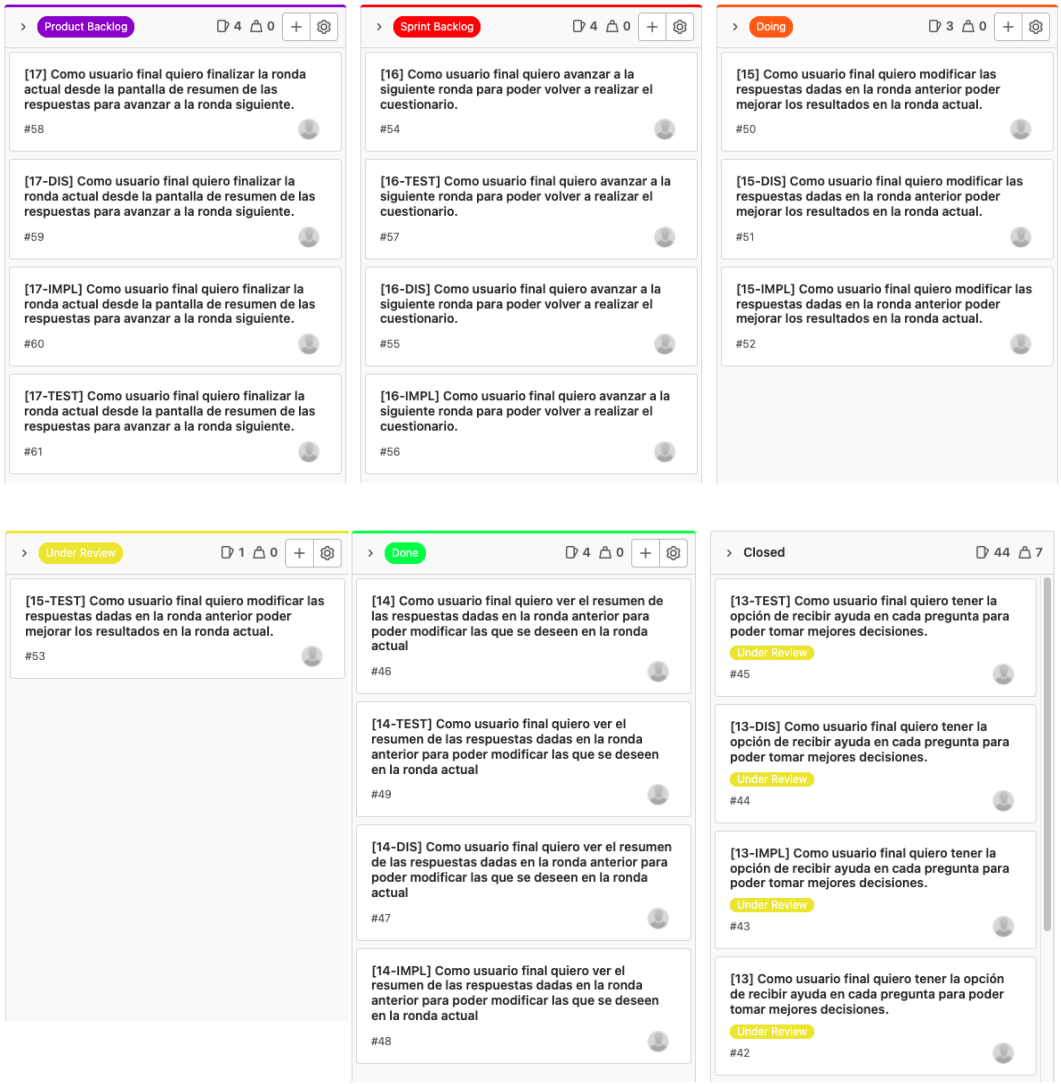

Figura 3.5: Issue Board GitLab Crossroads (Elaboración propia)

3.7.3. Miro

Miro [48] es una plataforma *online* que permite, entre otras cosas, realizar bocetos del diseño de aplicaciones móviles. Proporciona una pizarra para que diversos miembros de un equipo trabajen de forma colaborativa, aunque en este caso se utiliza para desarrollar los bocetos de la interfaz de usuario de manera individual.

3.7.4. Overleaf

Overleaf [58] es un editor de texto LaTeX colaborativo. LaTeX es un sistema de composición de textos, formado por macros de TeX, que es un sistema tipográfico. Ha sido utilizado para realizar la memoria del Trabajo de Fin de Grado. Overleaf tiene diversas herramientas como la posibilidad de comprobar el historial de versiones, hacer revisiones con comentarios y compilar el código para producir un PDF, a mayores de la funcionalidad propia de LaTeX.

3.8. Herramientas para la comunicación

3.8.1. Webex

Para las reuniones de seguimiento del proyecto con la tutora, es decir, Sprint Planning, Sprint Review, Sprint Retrospective y Weekly Scrum, se ha utilizado la plataforma **Webex** [76]. Es una plataforma de videoconferencias, con posibilidad de compartir pantalla para poder mostrar los progresos y revisar las dudas surgidas.

3.8.2. Telegram

Para dudas más breves y para avisar si hay que modificar el horario de la reunión semanal, se ha utilizado **Telegram** [70], que es una aplicación de mensajería instantánea.

3.8.3. Vysor

Vysor [75] es un programa para visualizar la pantalla del dispositivo móvil en el ordenador, y de esta forma poder compartir los avances de la aplicación con la tutora durante las reuniones de seguimiento. Ofrece la posibilidad de conectarse al móvil mediante cable o de forma inalámbrica al tener el ordenador y el *Smartphone* conectados a la misma red WiFi.

Capítulo 4

Análisis

En este capítulo se detalla la fase de análisis del proyecto, que consiste principalmente en el desarrollo del modelo de dominio del sistema, la máquina de estados para analizar y documentar el proceso del juego, y la explicación de cómo se elaboran la puntuación y las gráficas obtenidas al final de cada ronda.

4.1. Modelo de dominio

A partir de las historias de usuario mostradas en el *Backlog* inicial 2.1, se han desarrollado los requisitos de información a partir de los cuales se ha creado el modelo de dominio mostrado en la Figura 4.1. Estos requisitos de información en forma de historias de usuario se pueden observar en la Tabla 4.1.

El sistema está compuesto por un **jugador**, que durante 3 **rondas**, selecciona una **opción** para cada una de las 14 **preguntas** que forman el formulario. En cada pregunta puede consultar las **pistas** disponibles para ellas. Una vez finalizada la ronda, se obtiene una **puntuación**, compuesta por puntos según la precisión, ecología y equilibrio alcanzados, y valores para realizar una gráfica de temperatura y otra de PIB anual per cápita. También recibe una **recomendación** para intentar mejorar las puntuaciones en las siguientes rondas.

4.2. Proceso del juego

Para representar los diferentes estados por los que va pasando el jugador a lo largo del juego, se ha realizado una **máquina de estados** (Figura 4.2). En ella se pueden ver además de los estados en los que puede encontrarse el jugador, los eventos que le llevan a esas situaciones, y las condiciones para permanecer en un estado o avanzar al siguiente.

Requisitos de información	
ID	**Requisito**
RI01	Como equipo de desarrollo quiero guardar los siguientes datos de un **jugador**: identificador, nombre de usuario, edad y país, para distinguir a los usuarios y poder diferenciarlos según diferentes parámetros.
RI02	Como equipo de desarrollo quiero guardar los siguientes datos de una **pregunta**: identificador, texto de la pregunta y tipo, para poder mostrar las preguntas en el cuestionario.
RI03	Como equipo de desarrollo quiero guardar los siguientes datos de una **opción**: identificador, etiqueta y texto de la opción, para poder mostrar las opciones en el cuestionario.
RI04	Como equipo de desarrollo quiero guardar los siguientes datos de una **pista**: identificador y texto, para poder mostrar las pistas de cada pregunta.
RI05	Como equipo de desarrollo quiero guardar los siguientes datos de la **puntuación**: puntos de precisión, ecología y equilibrio y valores para realizar las gráficas de temperatura y PIB, para poder mostrar los resultados de cada ronda.
RI06	Como equipo de desarrollo quiero guardar los siguientes datos de una **recomendación**: identificador y texto, para poder mostrar las recomendaciones de cada ronda.

Tabla 4.1: Requisitos de información

En la figura anterior se puede apreciar que el jugador obtiene información sobre el juego hasta que decida iniciar partida. Una vez comience, responderá preguntas y avanzará a las siguientes hasta la pregunta 14. En todas estas cuestiones, podrá recibir ayuda si así lo desea. Cuando haya respondido a todas, visualizará los resultados de la ronda y recomendaciones para mejorar sus puntuaciones. Posteriormente, se avanzará a la ronda siguiente. Este proceso se repetirá 3 rondas, y después se mostrarán los resultados finales.

4.3. Obtención de la puntuación

Una vez respondidas todas las preguntas, se muestran tres puntuaciones relacionadas con las preguntas respondidas:

- **Precisión**: Se valora la proximidad a los objetivos propuestos. Se mide la distancia entre el valor objetivo y el último valor al final de la simulación tanto de la temperatura como del PIB. Este valor se combina por medio de una operación con un rango establecido como aceptable para obtener la puntuación final de la variable precisión.
- **Ecológico**: Se valora si los objetivos propuestos son respetuosos con el medio ambiente. Se calcula un margen de mejora entre la temperatura objetivo propuesta y el valor final obtenido en la simulación. Estos dos valores se combinan mediante una operación usando también el margen de mejora hallado anteriormente para saber como de respetuoso con el medio ambiente es el resultado.

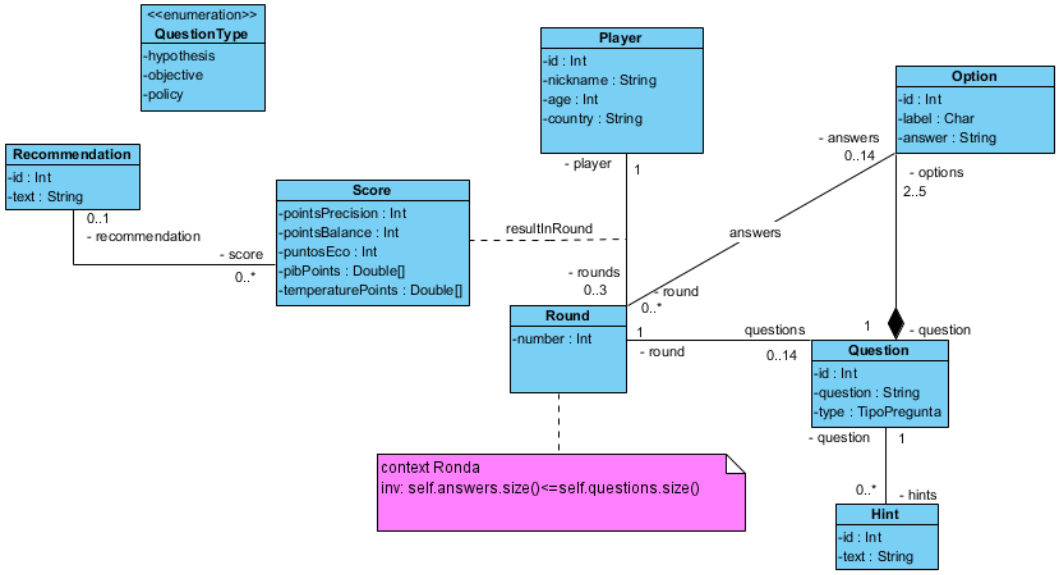

Figura 4.1: Modelo de Dominio

- **Equilibrio**: Se valora si hay un equilibrio entre la ecología y la economía. Se calcula el margen de mejora de los valores finales de las simulaciones de temperatura y PIB junto con los rangos asociados para medir el grado de equilibrio entre los resultados de economía y de medio ambiente.

Estas puntuaciones tienen un valor comprendido entre 0 y 100. Cuanto mejor se haya hecho, mayor será el valor obtenido en cada una de ellas.

4.4. Obtención de las gráficas

Al finalizar cada ronda, el usuario verá los resultados de las medidas tomadas en forma de gráficas. Esas gráficas representan las predicciones en cuanto a **temperatura** y a **Producto Interior Bruto anual per cápita** como indicador de bienestar. A su vez, en cada gráfica se representan dos líneas, una para simbolizar el objetivo propuesto y otra para la simulación creada a partir de las respuestas a las preguntas.

Las líneas rojas representan los puntos obtenidos al realizar la simulación de la temperatura y del PIB. Estos datos se obtienen a partir de las respuestas dadas por el jugador para las preguntas de hipótesis y decisiones políticas.

Las líneas rectas azules representan el valor de la temperatura y del PIB objetivo, obtenido a partir de las respuestas a las preguntas de tipo objetivo.

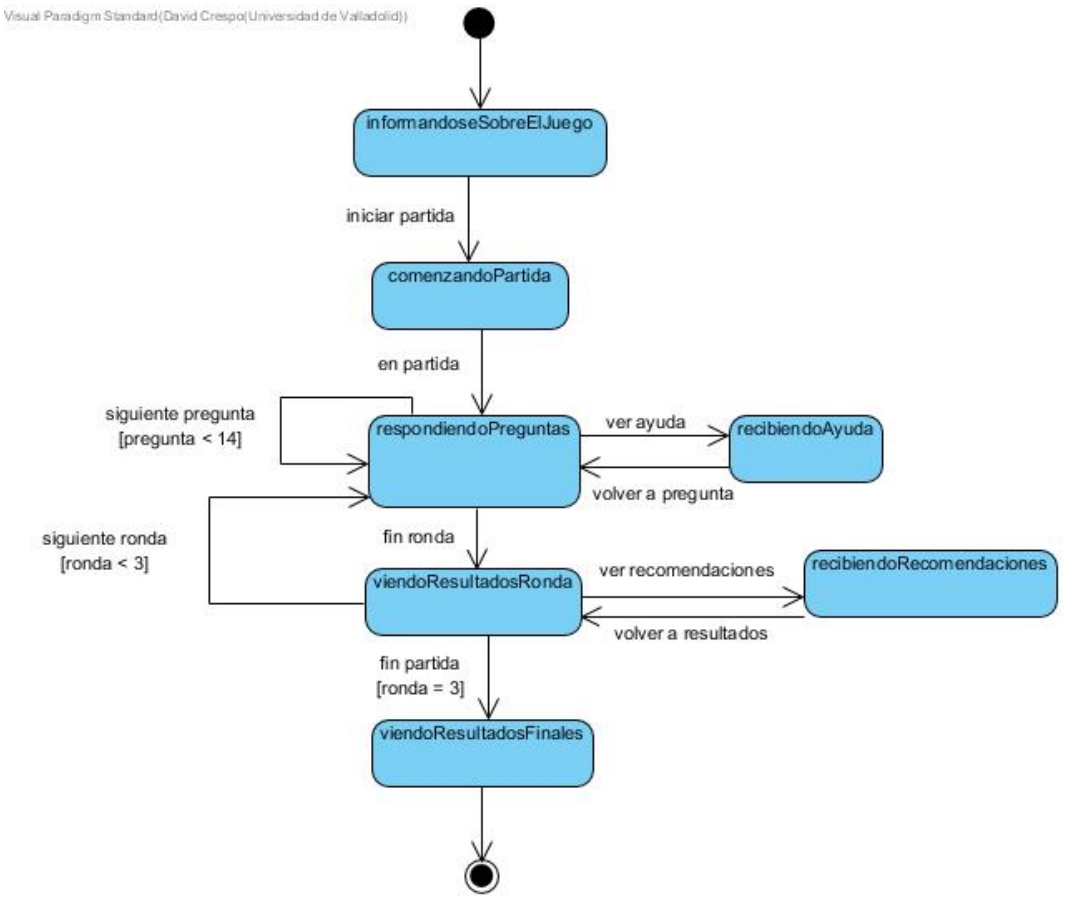

Figura 4.2: Análisis de la interacción del usuario con el sistema mediante una máquina de estados.

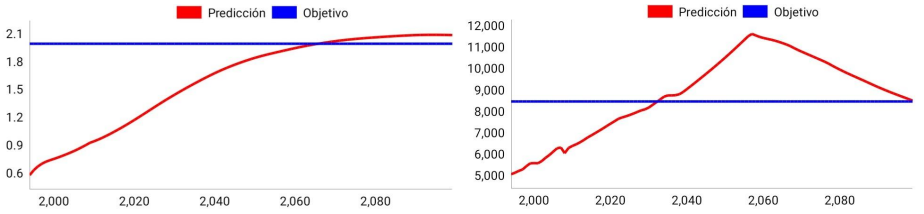

Figura 4.3: Gráficas de temperatura y PIB Anual per cápita

Capítulo 5

Diseño

5.1. Decisiones de diseño

En esta sección se resumen las decisiones de diseño tomadas para abordar la solución:

- **API 28**: Como se comenta en la sección 3.1 de forma más detallada, se eligió esta versión mínima de *Android*, debido a que el 75 % de los dispositivos son compatibles con esta versión, y esta cifra podría ser incluso más alta entre los jóvenes, que son los usuarios objetivos ya que suelen actualizar sus dispositivos móviles más frecuentemente que los adultos debido a la importancia que tienen para ellos. Esta versión es adecuada para reducir el posible riesgo de no poder utilizar tecnología lo suficientemente actual, ya que toda la que se ha necesitado usar ha estado disponible para este nivel de API.

- **Kotlin**: Es el lenguaje recomendado por *Google* para el desarrollo *Android* ya que es más moderno y conciso que *Java*. Como se explica en la sección 3.2, es el lenguaje más usado por los desarrolladores *Android*, y debido a la experiencia previa del alumno y su uso en las prácticas de empresa, se decidió optar por este lenguaje.

- **XML**: Para realizar el diseño de la interfaz se ha utilizado el lenguaje de etiquetas XML. La alternativa era *Jetpack Compose*, pero debido a que aún tiene muchos elementos en fase experimental, la necesidad de aprender esta librería y la mayor dificultad para encontrar documentación, se prefirió usar la primera opción para reducir el riesgo relacionado con la falta de conocimiento de las tecnologías a utilizar (Riesgo 2.14).

- **Arquitectura single-activity**: Consiste en construir la aplicación a partir de una sola actividad o el menor número de actividades posibles, con los fragmentos alojados dentro de ellas. Desde la celebración de **Google I/O** 2018, que es un evento organizado por *Google*, es recomendado que se siga esta arquitectura [24]. En este proyecto se ha utilizado una sola actividad, con todos los *fragments* albergados en ella. También se ha utilizado **Navigation Component** para la navegación entre los fragmentos y el

Plugin Safe Args para el envío de argumentos de forma segura entre ellos. En las secciones 3.2.2 y 3.2.3 se explican los detalles del *Navigation Component* y *Safe Args* respectivamente.

- **Animaciones**: Para añadir fluidez a las transiciones entre fragmentos, se decidió añadir un desplazamiento de salida y uno de entrada al fragmento saliente y al entrante respectivamente.
- **Toolbar**: Se ha decidido incorporar en todos los *fragments* un *toolbar* o barra de herramientas, con el título de la aplicación o del fragmento y una flecha para navegar a la pantalla anterior en caso de que sea posible.
- **Room**: Se decidió utilizar debido a que las llamadas a la base de datos remota se producen de manera asíncrona y no se puede devolver el resultado de esa llamada como resultado del método de manera convencional. Las opciones encontradas fueron la implementación de *Callbacks* o el uso de *Room*. Se optó por la segunda debido a que es la recomendación de *Android* en el punto 5 de su guía básica [6] y porque permite acceder a los datos desde cualquier lugar de la aplicación mediante los repositorios en lugar de enviar todos los datos continuamente entre los *fragments*.
- **Corrutinas**: Para realizar las peticiones a APIs, se ha hecho uso de las corrutinas. Las corrutinas [8] sirven para delegar tareas asíncronas que podrían bloquear el hilo principal en otros hilos, liberando de responsabilidad al primer hilo, que es el encargado de mostrar la interfaz de usuario. Esto aporta mayor ligereza a la aplicación, obteniendo más rápidamente la respuesta de las consultas a la API, y menos fugas de memoria debido a que se establece el alcance de las operaciones ejecutadas mediante corrutinas.
- **Base de datos remota**: Para el almacenamiento remoto persistente se optó por utilizar el *Backend* existente [44] [46] [27] previamente desarrollado en otros proyectos. Para ello, fue necesario adaptarlo, ya que está pensado para realizar partidas multijugador por equipos y con varios equipos por juego. Principalmente hubo que crear un identificador de grupo (de manera estática ya que no es necesario que cambie para cada partida), registrar el moderador previamente en la base de datos y crear un jugador. El jugador es quien en realidad juega, siendo el único jugador del grupo ficticio creado, y el único grupo en juego. El moderador es necesario porque es el encargado en el *backend* existente de iniciar la partida, avanzar de ronda o finalizar la partida.

5.2. Patrón arquitectónico MVVM

Model View ViewModel [20] (MVVM) es un patrón utilizado para favorecer el desacoplamiento, separando lo máximo posible la lógica de la interfaz de usuario. El propósito de esto es minimizar los problemas durante el desarrollo, facilitar el mantenimiento y la escalabilidad de la aplicación y permitir la reutilización de código.

Con este patrón, la aplicación tiene tres componentes principales [20]:

- **View**: Es la interfaz de usuario (IU), formada por las activities, los fragments, los archivos XML y elementos auxiliares para mostrar los datos como por ejemplo las clases adaptadoras para las listas. Estas clases deben ejecutar la menor lógica posible, siendo su mayor responsabilidad capturar eventos recibidos por parte del usuario y mostrar los datos, pero no la manera en la que se obtienen esos datos.

- **ViewModel**: Es el encargado de obtener los datos y enviárselos a las vistas. Siguiendo el patrón observador, cada vista se suscribe a los datos deseados de su respectivo *view model*, y estos cuando hay un cambio, lo notifican a la vista.

- **Model**: Contiene la lógica de negocio. Es el componente en la que se almacena los datos en modelos o clases de datos, obtenidos al realizar llamadas a la base de datos o a un servicio web.

Para observar los datos obtenidos por los *view models*, se utiliza **Live Data** [11]. Es una clase contenedora de datos, observable y optimizada para los ciclos de vida de las actividades o fragmentos de la aplicación. Esta optimización asegura que los únicos datos que se actualizan para ser observados, son los de los componentes de la aplicación cuyo ciclo de vida se encuentra en estado activo.

En la Figura 5.1 se muestra las relaciones entre cada componente de los que forman el patrón MVVM. El acoplamiento es mínimo ya que el componente *View* solo "conoce" al *ViewModel*, y este solo "conoce" al *Model*.

Dado que mediante *Retrofit* se realizan consultas a la *API*, y también se dispone de la base de datos local accesible mediante *Room*, en esta aplicación el modelo esta constituido por ambas partes, accediendo a la primera mediante el paquete `services` y a la segunda a través del paquete `dao` como se explicará en la sección 5.4.

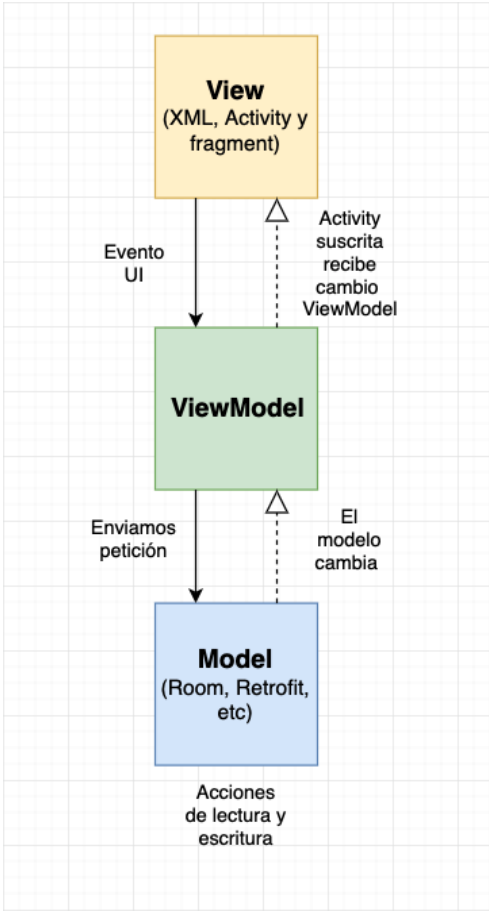

Figura 5.1: Relaciones entre los componentes del patrón MVVM [20]

5.3. Patrones de diseño

A continuación se exponen los distintos patrones de diseño utilizados durante el desarrollo del proyecto. En cada sección, se explicará brevemente en que consisten cada uno de ellos y cual ha sido su uso en la aplicación.

5.3.1. Patrón Observador

El **Patrón Observador** [63] permite crear un mecanismo de subscripción mediante el cual, cualquier interesado puede ser notificado al producirse cambios en el objeto observado.

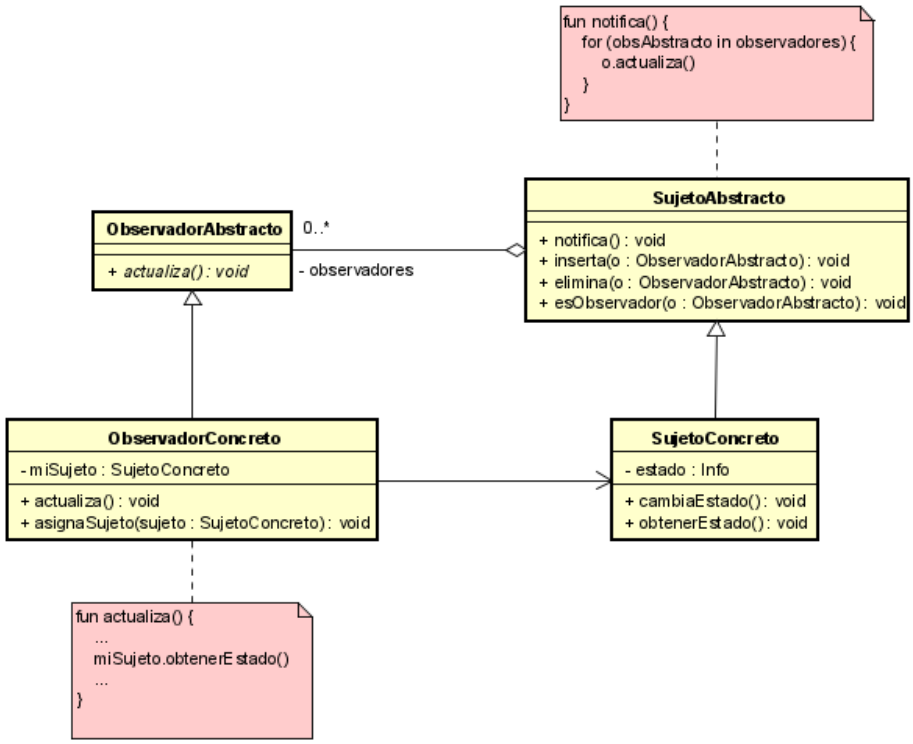

Figura 5.2: Patrón Observador

En este proyecto este patrón cobra gran importancia, ya que uno de los pilares de la arquitectura MVVM, es que la Vista observa los datos necesarios del *ViewModel* y cambia al ser notificada de los cambios de estado producidos en estos objetos observados.

5.3.2. Método Factoría

El **método factoría** [26] nos permite crear un objeto mediante una clase denominada la clase factoría, que oculta los detalles de creación del objeto.

En este proyecto se utiliza para crear *ViewModels* mediante una clase Factoría que permite pasarle argumentos al *ViewModel*. Los *ViewModels* se deben crear mediante la clase *ViewModelProvider* para controlar los cambios del ciclo de vida de la vista a la que están asociados, y esta clase por defecto no admite el paso de parámetros.

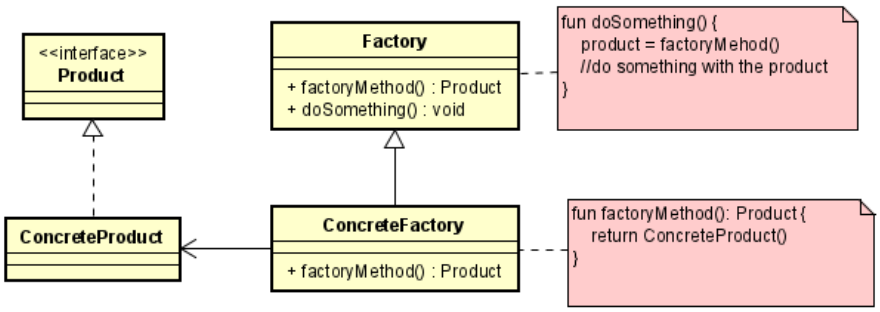

Figura 5.3: Método Factoría

5.3.3. Patrón Adaptador

El patrón **Adaptador** [77] se usa para convertir una interfaz en otra, de forma que se pueda utilizar la primera mediante la segunda. El adaptador es un objeto que ayuda a transformar una interfaz para que pueda ser utilizada.

En la Figura 5.4 se muestra el patrón Adaptador. La clase *Objetivo* define la interfaz esperada por el cliente. La clase *Adaptada* es la clase que se necesita adaptar y *Adaptadora* adapta *Adaptada* a la interfaz *Objetivo*.

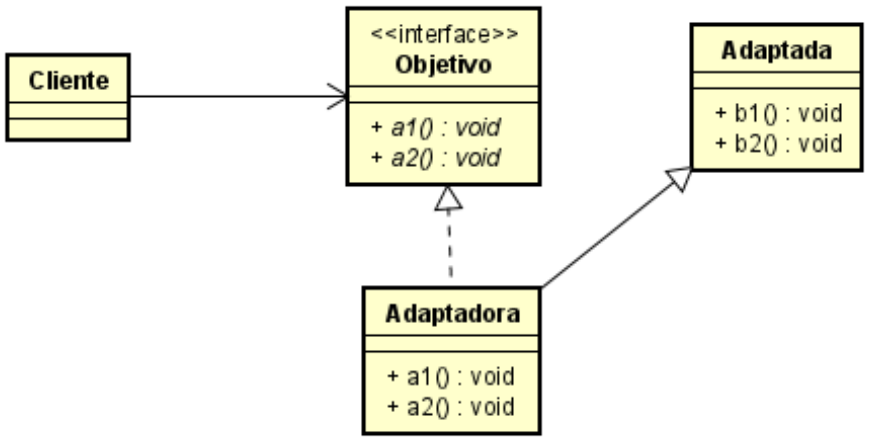

Figura 5.4: Patrón Adaptador

Este patrón es utilizado para mostrar listas dinámicas mediante el elemento *RecyclerView* de *Android*. Se crea una clase adaptadora que extiende *RecyclerView.Adapter* para convertir los objetos que se quieren mostrar en elementos de la lista.

5.3.4. Patrón Singleton

El patrón **Singleton** [64] o instancia única, es un patrón de diseño que permite que una clase tenga una única instancia y proporciona un sólo punto de acceso global a esta instancia. La Figura 5.5 muestra el diagrama UML básico de una clase *Singleton*.

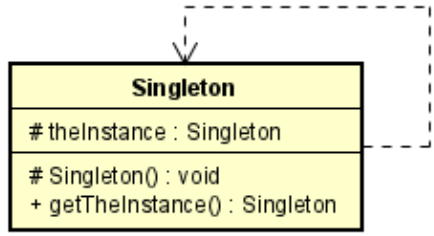

Figura 5.5: Patrón Singleton

Fue usado para los repositorios encargados de realizar todas las peticiones a APIs y de gestionar la base de datos local, y para obtener la referencia a base de datos local.

En el caso de los repositorios fue utilizado debido a que inicialmente al instanciar un repositorio, se ejecutaba un método encargado de eliminar los posibles datos que hubiera de partidas anteriores en la base de datos local, por lo que si un mismo repositorio es usado por dos o más *View Models*, el segundo haría que los datos guardados por el primero fueran eliminados.

Finalmente, todos los datos anteriores son borrados únicamente en el lanzamiento de la aplicación, por lo que ya no sería necesario el uso del patrón *Singleton* para ellos, pero se mantuvieron así ya que pueden utilizar este patrón porque no necesitan ser *Thread safe*, es decir, no se lanzan peticiones simultáneamente a ningún repositorio y por tanto no necesitan ser objetos separados.

La clase de acceso a la base de datos local también es *Singleton* porque sólo se necesita una instancia de ella, y tener más de una podría llevar a tener fugas de memoria [22].

5.3.5. Patrones DAO y DTO

El **patrón DTO** [29] se utiliza para enviar y recibir datos entre el cliente y el servidor o entre diversas capas de la aplicación mediante un objeto de transferencia de datos.

El **patron DAO** [29] es el encargado de obtener o enviar esos datos. Almacena en la clase *DAO* las operaciones *CRUD* necesarias (*create*, *read*, *update*, *delete*), separando así la lógica de acceso a datos de la lógica de negocio.

En este proyecto se usan para transferir datos entre la implementación del acceso a la fuente de datos local mediante *Room* y los *ViewModels*.

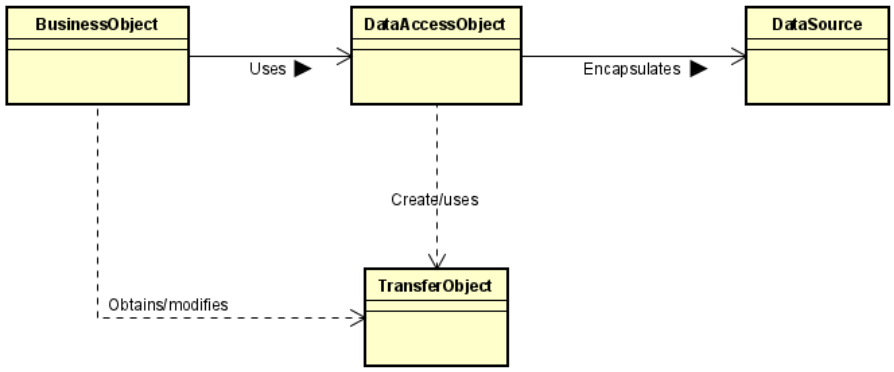

Figura 5.6: Patrón DAO DTO

Los *DTOs* que se obtienen son de tipo *entity*. No son *business entities* porque no tienen una funcionalidad de negocio, solamente son representaciones de datos. No se traen los objetos de las bases de datos en crudo para posteriormente transformarlos a objetos del modelo, sino que se hace una conversión automática a *entity*. A bajo nivel sí que se obtienen los objetos de datos en *JSON*, pero se convierten automáticamente a *entities*.

En cuanto a los DAOs, se encuentran localizados en el paquete "dao" dentro de "db". También se dividen en distintas clases de acuerdo al tipo de datos solicitados, como se muestra en la Sección 5.4.

5.4. Diseño arquitectónico

La aplicación está estructurada en los 5 paquetes principales que se explican a continuación:

- **db**: Este paquete contiene todas las clases necesarias para almacenar y obtener los datos guardados en la base de datos local accesible mediante *Room*. Se divide en los siguientes sub-paquetes:
  - **converter**: En él se encuentran las clases que se encargan de convertir tipos de datos no primitivos a *String* y viceversa. Es necesario ya que en *Room* los datos se guardan como *JSONs*, y por ello al querer obtenerlos o guardarlos hay que convertirlos.
  - **dao**: Contiene las interfaces para realizar las operaciones CRUD (*Create, Read, Update y Delete*) con los datos almacenados en *Room*. Este paquete permite el acceso al Modelo local y por tanto representa el Modelo de la arquitectura MVVM.
  - **repository**: Las clases repositorio proporcionan un acceso a los datos de *Room*. Son los encargados de comunicarse con los daos para acceder a la información guardada.

    - **database**: Contiene la clase necesaria para obtener una conexión con la base de datos.

- **entity**: Engloba las clases de las entidades del dominio en forma de *POJOs*. Son las representaciones de los datos, sin lógica de negocio. Actúan como *DTOs*.

- **services**: En este paquete se encuentran las interfaces necesarias para realizar las consultas a la *API*. Permite el acceso al Modelo remoto, y representa el Modelo de la arquitectura MVVM.

- **viewmodels**: Contiene los *ViewModels* y las factorías necesarias para crear algunos de ellos. Es la capa ViewModel de la arquitectura MVVM.

- **views**: Este paquete contiene las vistas y elementos relacionados que conforman la interfaz de la aplicación. Es la capa View de la arquitectura MVVM. Se divide en:

    - **activities**: Contiene la única actividad que forma la aplicación, encargada de alojar a todos los fragmentos, como se ha comentado en la sección de decisiones de diseño 5.1.
    - **fragments**: Contiene los fragmentos que forman las diferentes pantallas de la aplicación.
    - **uicomponents**: Contiene las clases auxiliares para elaborar las vistas. En este caso únicamente se han necesitado adaptadores para crear las listas de elementos.

A continuación se muestran los diagramas de la arquitectura de la aplicación. En la Figura 5.7 se muestra el diagrama de *Decomposition&Uses Style* que representa el estilo de descomposición modular y uso de la arquitectura general, incluyendo todos los paquetes que conforman la aplicación y las relaciones entre ellos. En él se observa como el paquete entity actúa un paquete sumidero de soporte, ya que es accedido desde todos los otros paquetes. Los paquetes restantes son accedidos solo por el que se encuentra en el nivel inmediatamente superior.

En las figuras 5.8 a la 5.12 se observan los diagramas de *Decomposition&Uses Style* que representan el estilo de descomposición modular y uso de cada paquete descrito en la arquitectura general.

5.5. Diseño de la interfaz de usuario

Durante el desarrollo de la aplicación, al inicio de la realización de cada historia de usuario se han ido realizando los diseños de la interfaz de cada pantalla cuando correspondía con una vista nueva. De la Figura 5.13 a la 5.25 se muestran estos bocetos.

Los dos últimos bocetos tienen el título de la pantalla en la *Toolbar*, es decir, la barra superior de color azul. Sólo está en las dos últimas figuras debido a que fue una decisión de diseño final, pero posteriormente se aplicó a todos los diseños.

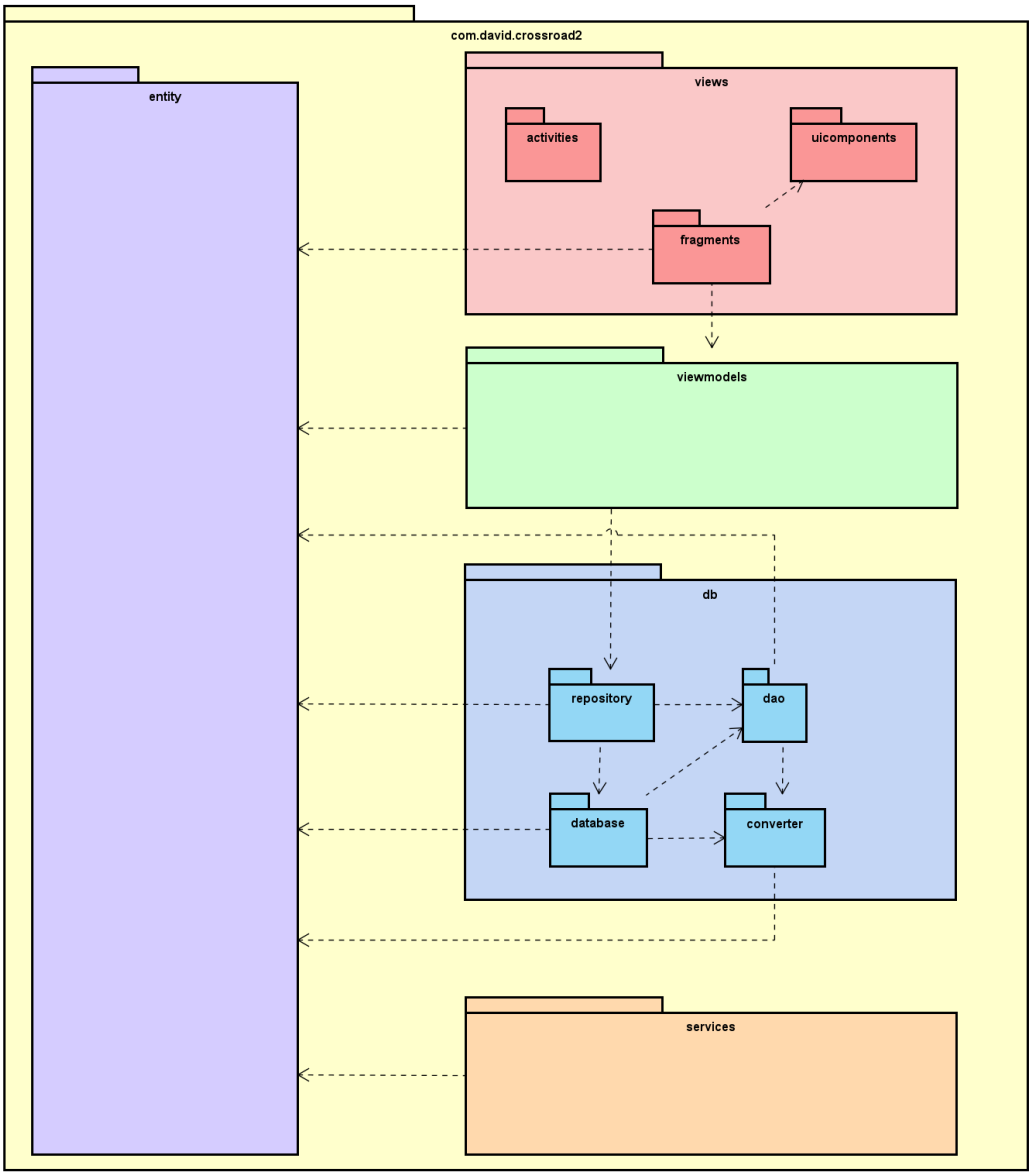

Figura 5.7: Decomposition&Uses Style general

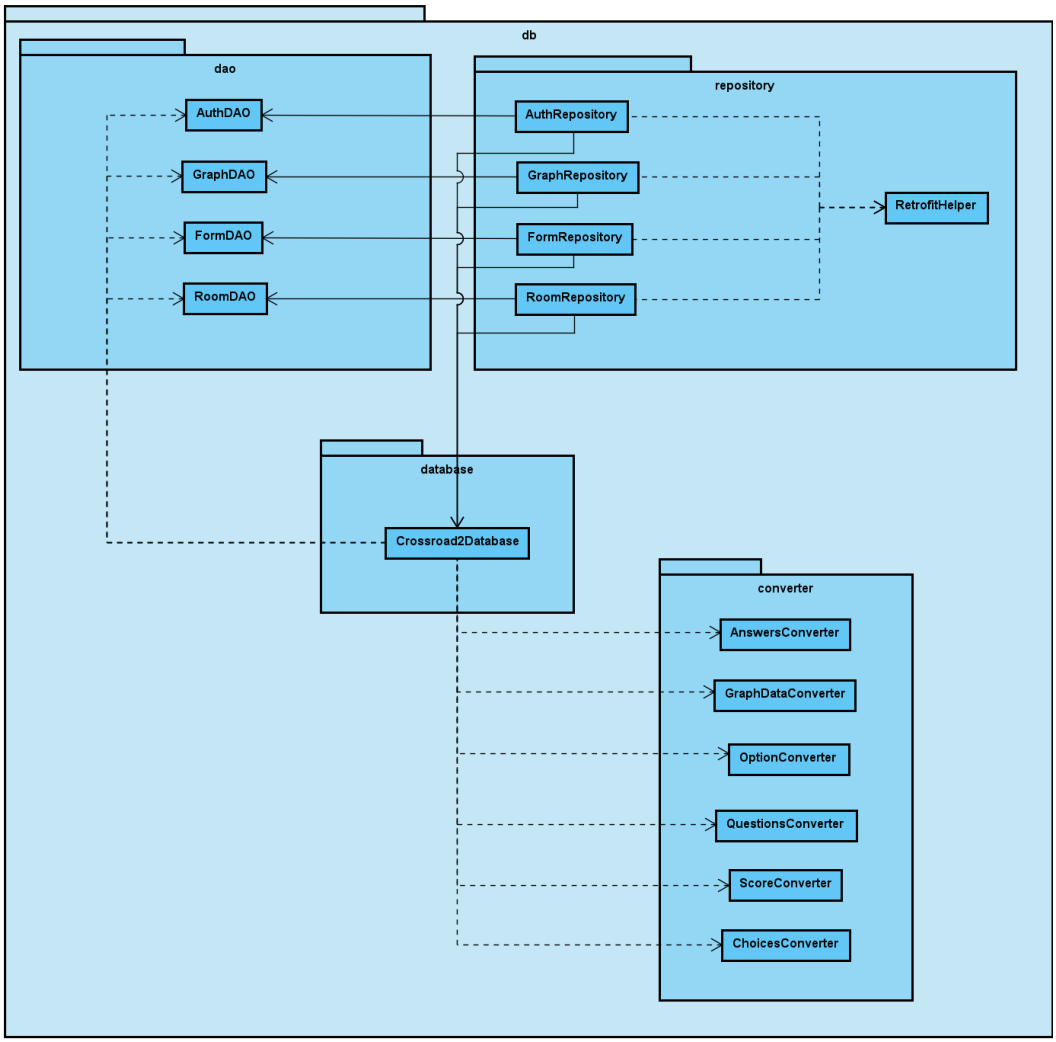

Figura 5.8: Decomposition&Uses Style del paquete **db**

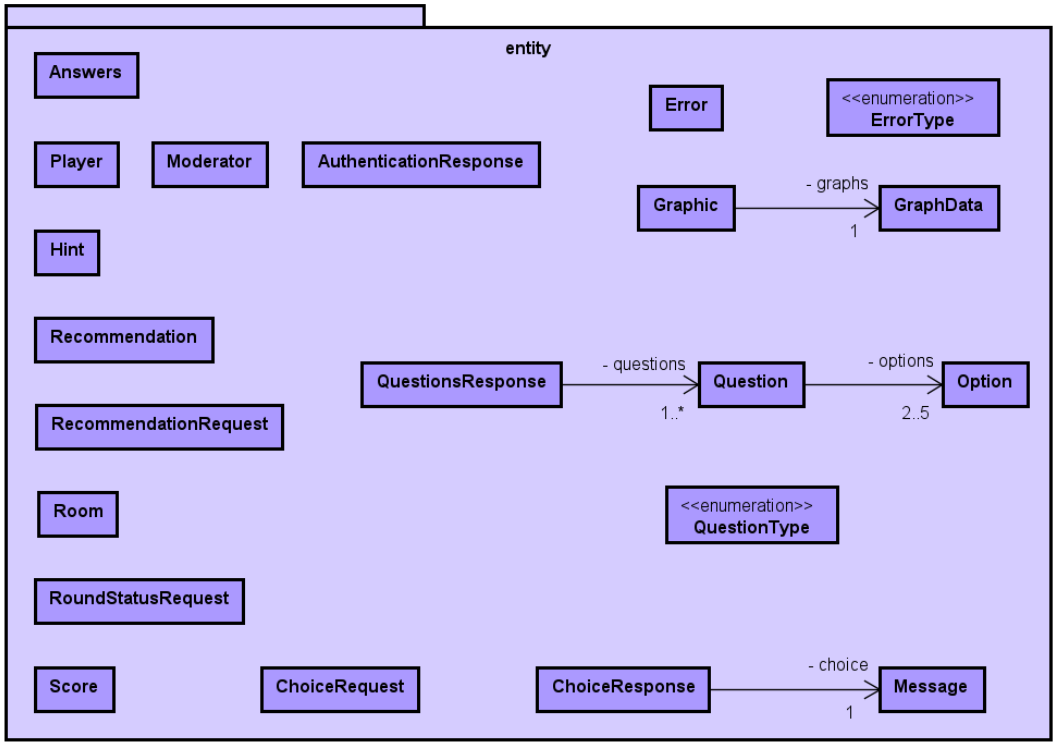

Figura 5.9: Decomposition&Uses Style del paquete **entity**

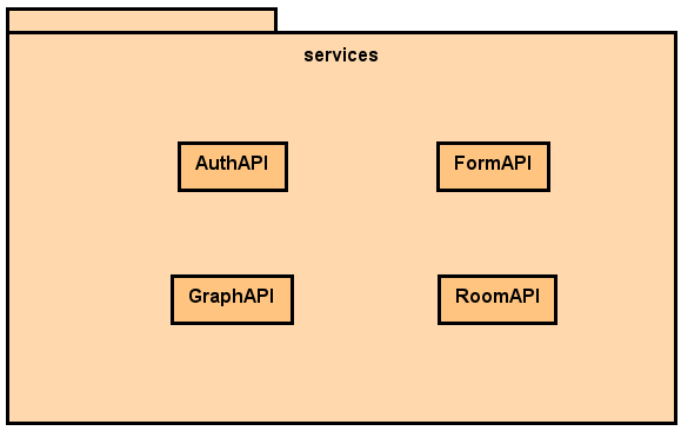

Figura 5.10: Decomposition&Uses Style del paquete **services**

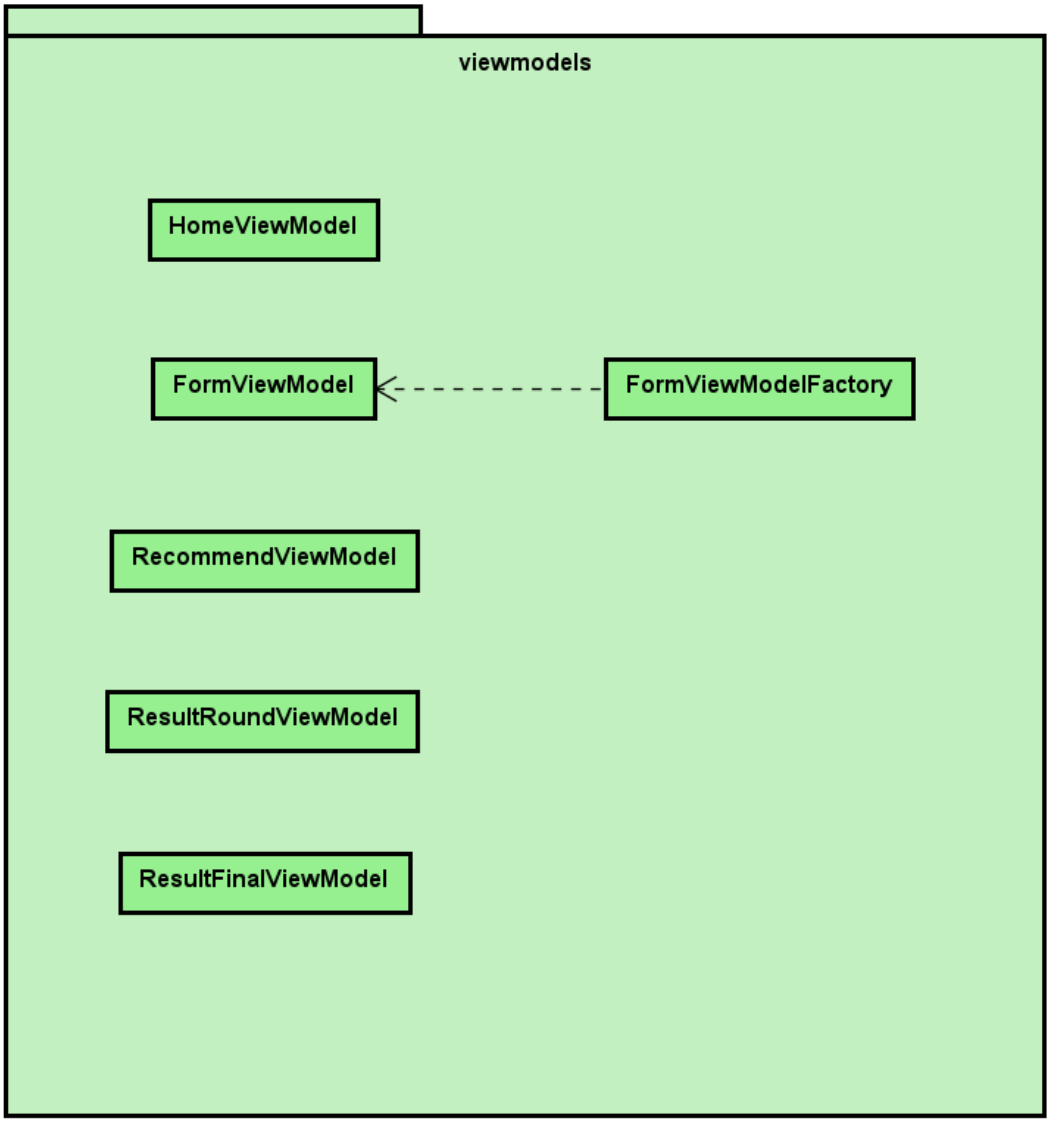

Figura 5.11: Decomposition&Uses Style del paquete **viewmodels**

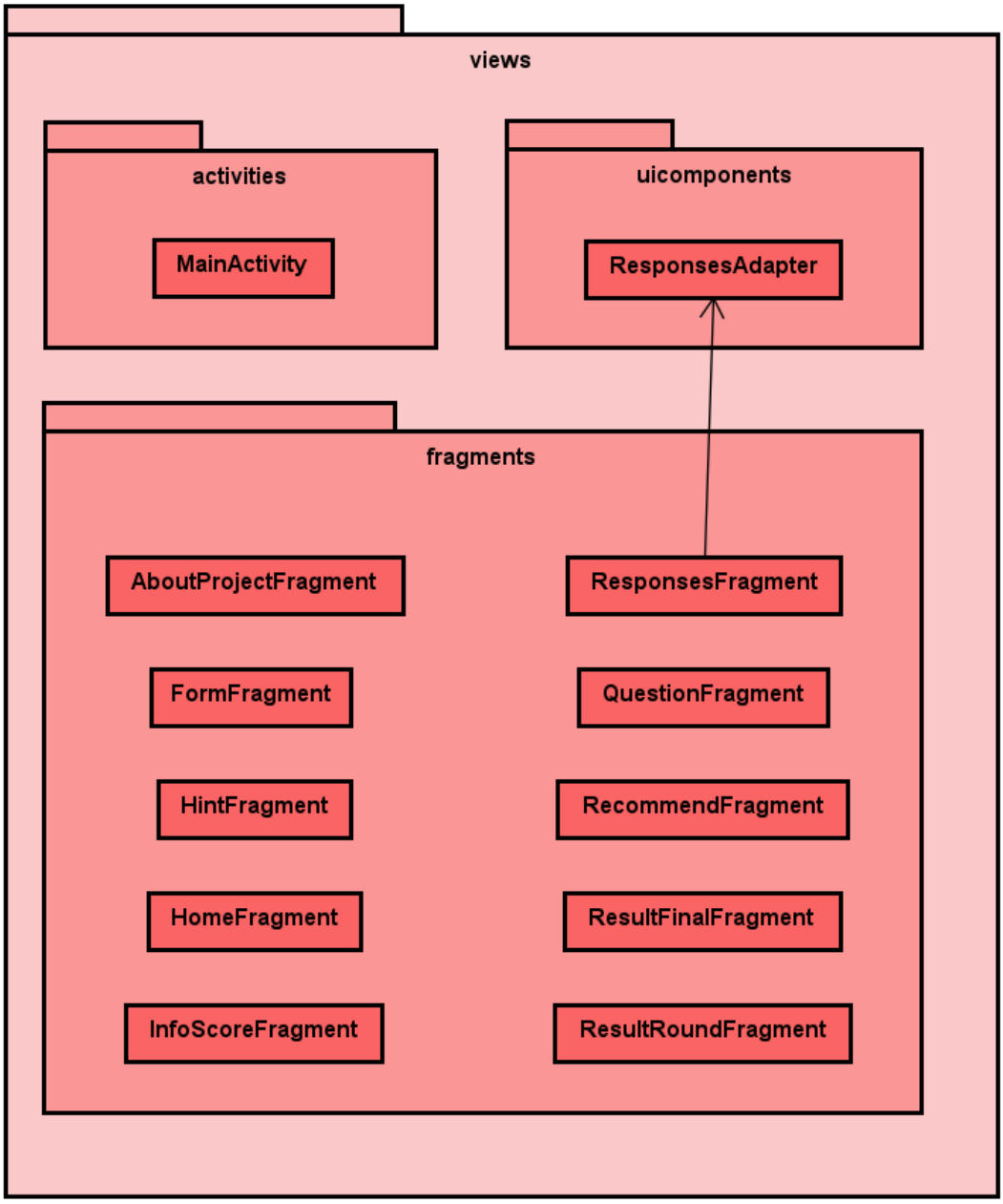

Figura 5.12: Decomposition&Uses Style del paquete **views**

Figura 5.13: *Splash Screen*

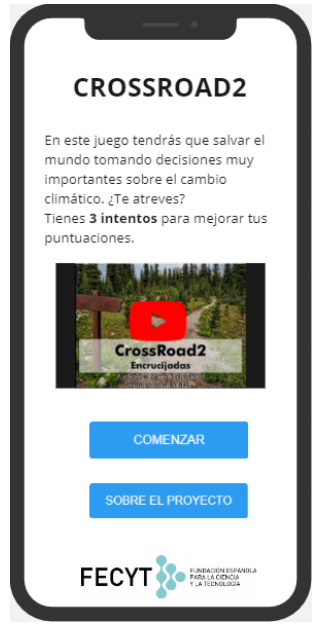

Figura 5.14: Pantalla inicial de Crossroads2

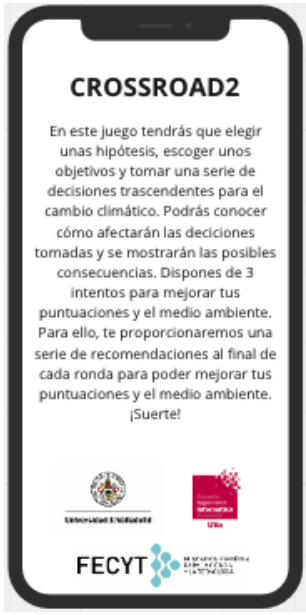

Figura 5.15: Información sobre el proyecto

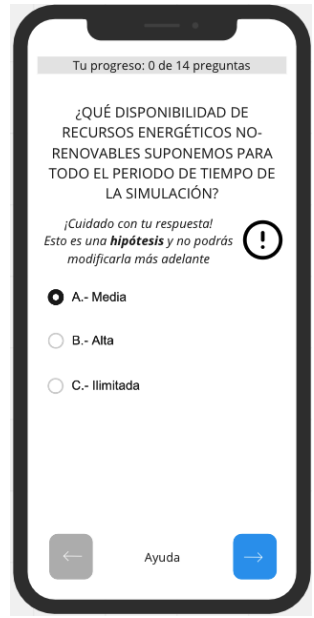

Figura 5.16: Pantalla inicial del formulario

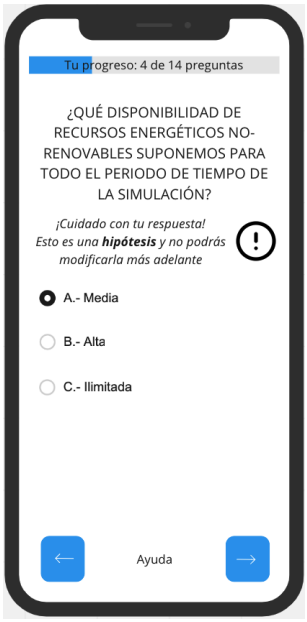

Figura 5.17: Pantalla intermedia del formulario

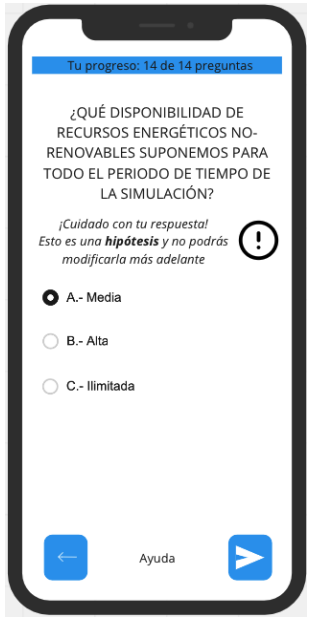

Figura 5.18: Pantalla final del formulario

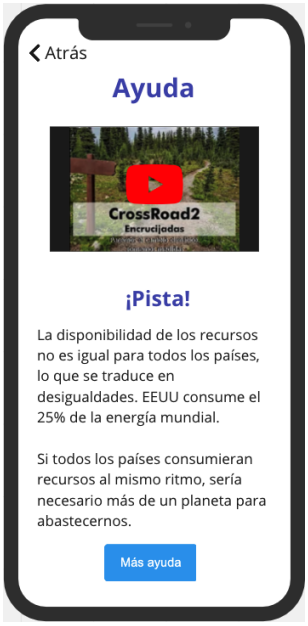

Figura 5.19: Pantalla de ayuda en cada pregunta

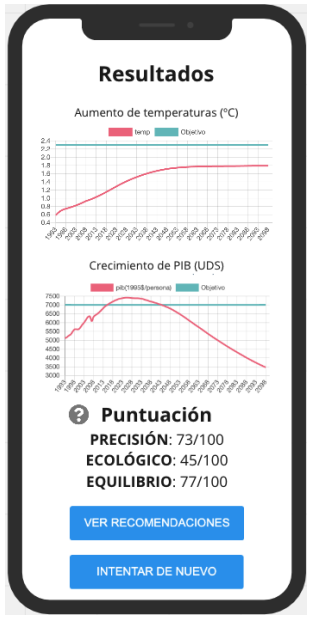

Figura 5.20: Resultados de ronda

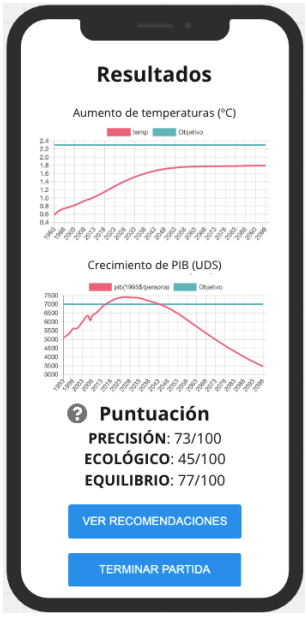

Figura 5.21: Resultados de ronda (última ronda)

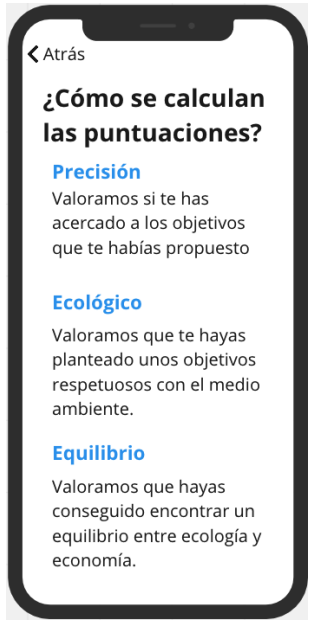

Figura 5.22: Información sobre el cálculo de las puntuaciones

Figura 5.23: Recomendaciones sobre los resultados de la ronda

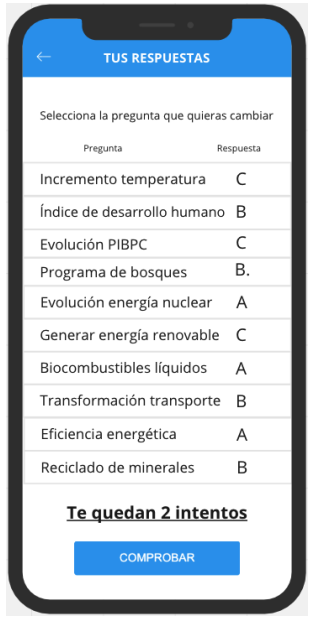

Figura 5.24: Respuestas de la ronda

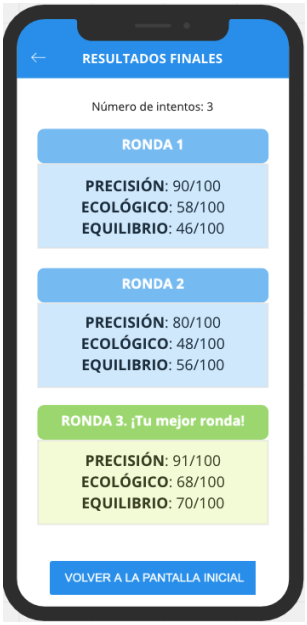

Figura 5.25: Resultados finales

5.6. Diseño de la comunicación entre objetos

Se ha elegido la historia de usuario **HU3.1: Como usuario final quiero poder marcar solo una de las opciones disponibles para responder a cada pregunta** como historia más representativa para explicar el diseño de la comunicación entre objetos. Esta abarca desde que el usuario pulsa en el botón de comenzar la partida (Figura 5.14) hasta que se muestra la primera pregunta (Figura 5.16). También se explica el funcionamiento de todo lo que se puede hacer desde la pantalla del formulario, es decir, seleccionar una opción, avanzar y retroceder en el cuestionario, y abrir la vista de ayuda (Figura 5.19).

Algunos subdiagramas no se mostrarán en este documento para no alargarlo excesivamente, pero todos ellos se pueden ver en el archivo *Astah* del repositorio de *GitLab* que se encuentra en el Apéndice B.

En las Figuras 5.26 y 5.27 se muestra la **secuencia principal**. En ella se crean todos los *listeners* necesarios, se inicializan algunas variables y se crea el *ViewModel*. Posteriormente se realiza la primera llamada a la *API*, que es la creación de la sala en la que se desarrolla la partida, y la observación de los datos por parte de la vista, a la espera de que el *ViewModel* le notifique que ha habido cambios. Se hace una distinción entre si es la primera vez que se accede al *Fragment* del formulario, o por el contrario se regresa a él después de haber abierto

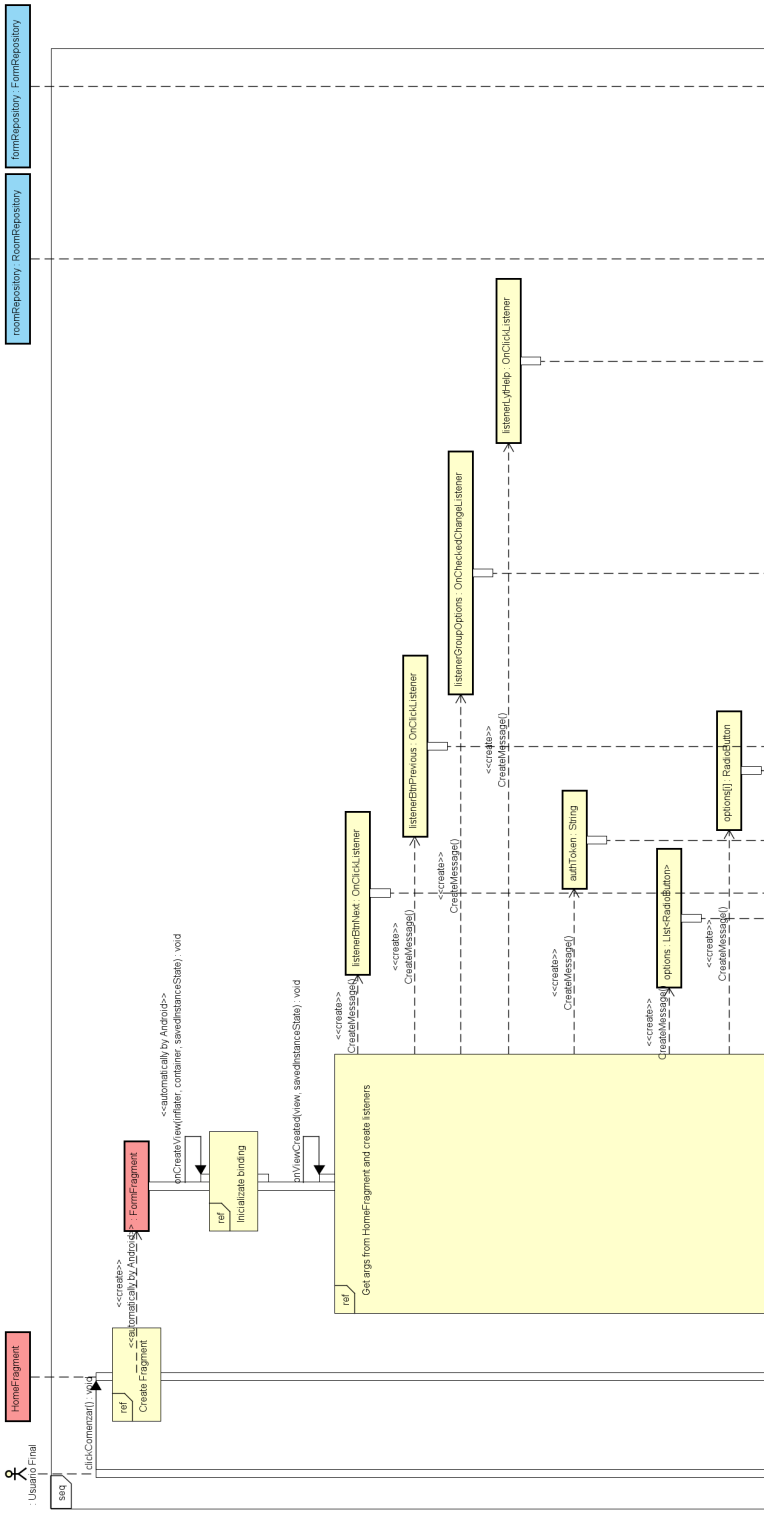

Figura 5.26: Diagrama de secuencia **Principal** de la realización de la HU 3.1 (1/2)

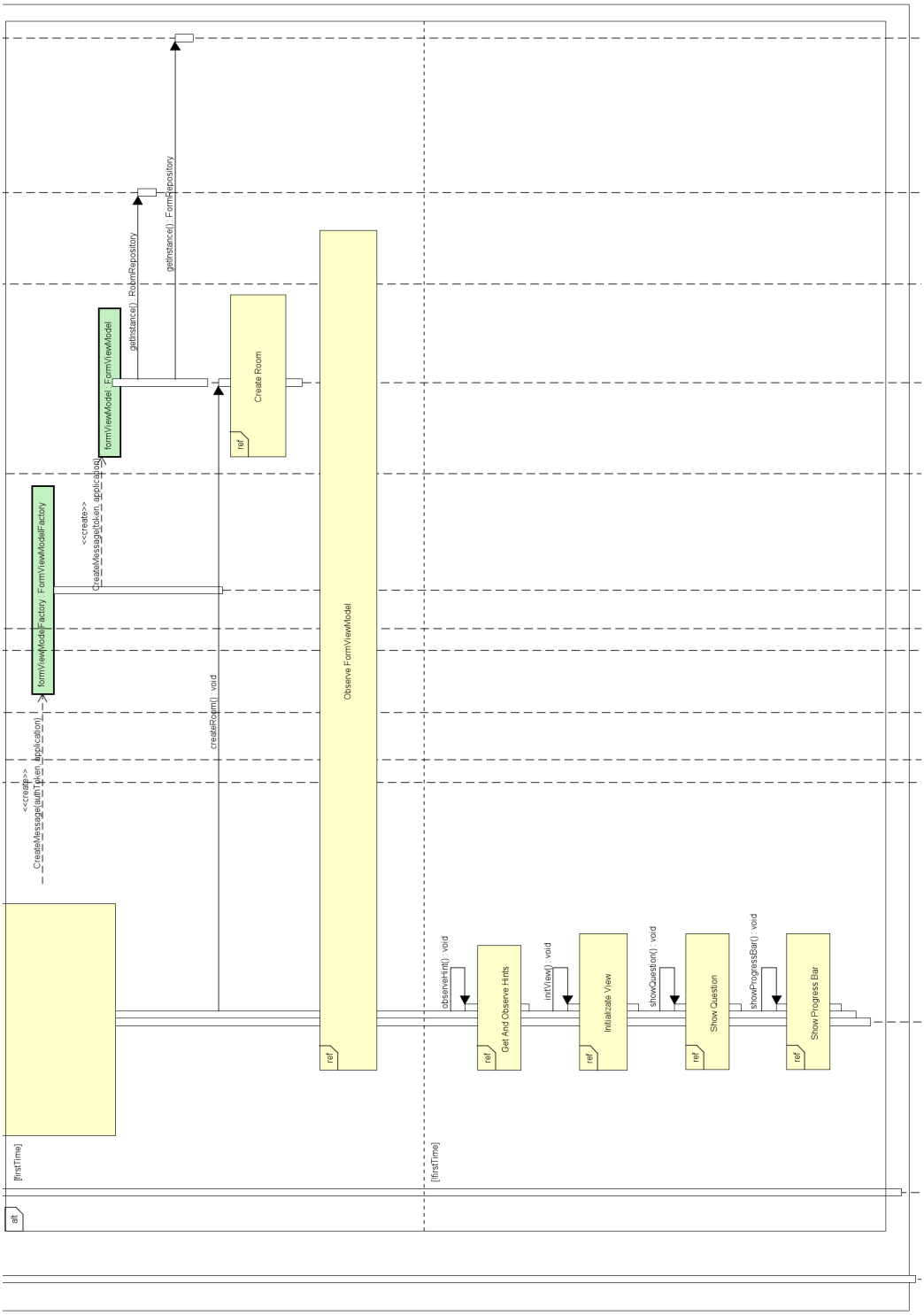

Figura 5.27: Diagrama de secuencia **Principal** de la realización de la HU 3.1 (2/2)

la vista de las pistas. La razón principal de esto es evitar que se cree una nueva instancia del *ViewModel*, y por tanto una sala nueva. Como el *ViewModel* ya esta creado, simplemente se observan las pistas, que es lo único que puede cambiar una vez recibidas las preguntas y creada la sala, y se muestra la pregunta actual con la respuesta marcada anteriormente por el jugador si la hubiera. El diagrama de las operaciones necesarias para mostrar una pregunta se explicará en el siguiente párrafo pero no se mostrará debido a su tamaño. En el anexo B se incluye un enlace al repositorio del proyecto donde está el archivo *Astah* con todos los diagramas.

Para **mostrar cada pregunta**, inicialmente se obtiene del *ViewModel* el número de preguntas totales y la pregunta actual. Posteriormente se actualiza la barra de progreso y se muestra la alerta de pregunta hipótesis si procede. Acto seguido se resetean los botones de las opciones, y se muestran tantos como opciones haya para la pregunta actual. Por último se marca la opción previamente seleccionada si es que el usuario ya había respondido a esta pregunta.

En la Figura 5.28 se realiza la consulta para **crear la sala**, y en la Figura 5.29 se trata la **respuesta de la API** obtenida. Esta respuesta llega de manera asíncrona por parte del servidor y si es correcta, se guarda en la base de datos local y se llama a la operación necesaria para iniciar la partida. Una vez comienza la partida, se realiza una consulta a la *API* para obtener las preguntas. Si cualquier operación relacionada con la base de datos falla, se guarda el error en *Room* y se observa desde la vista oportuna para mostrarlo al usuario.

La Figura 5.30 corresponde al diagrama de secuencia de un *listener*. El código que se encuentra dentro de un *listener* se ejecuta cuando sucede un evento. Este muestra lo que ocurre al pulsar el botón de **avanzar a la siguiente pregunta**. Mediante el *tag* del botón se sabe si la acción a realizar es la que tiene lugar cuando se encuentra en la última pregunta o no. Si no es la última, se actualiza la pregunta actual en el *ViewModel*, se solicita una pista para esa pregunta, y se muestra mediante las acciones realizadas en el anterior diagrama. Por último se comprueba si es la última pregunta para cambiar el *tag* y el icono del botón, y en caso de avanzar de la primera a la segunda pregunta, se habilita en botón de retroceder. Si es la última pregunta se comprueba que todas las cuestiones están respondidas. Si no lo están se avisa al usuario de que debe responder a todas las preguntas. Si ya ha contestado a todas, se envían las respuestas a la *API*, se inhabilita el botón de enviar para evitar que el usuario pudiera pulsar varias veces, y se avanza a la vista donde se muestran los resultados de cada ronda.

Por último, se muestra el diagrama de *Decomposition&Uses Style* de la **HU3.1**. Para simplificar el diagrama, está dividido en dos subdiagramas, uno con las clases relacionadas con la sala (Figura 5.31), y otro con las clases de las preguntas (Figura 5.32).

5.7. Diseño de despliegue

En la Figura 5.33 se muestra el diagrama de despliegue de la aplicación. En él se puede observar que el servidor está desplegado a través de *Tomcat*, que esta conectado a las dos bases de datos. En la de MySQL se almacena la información de la partida: sala, jugador,

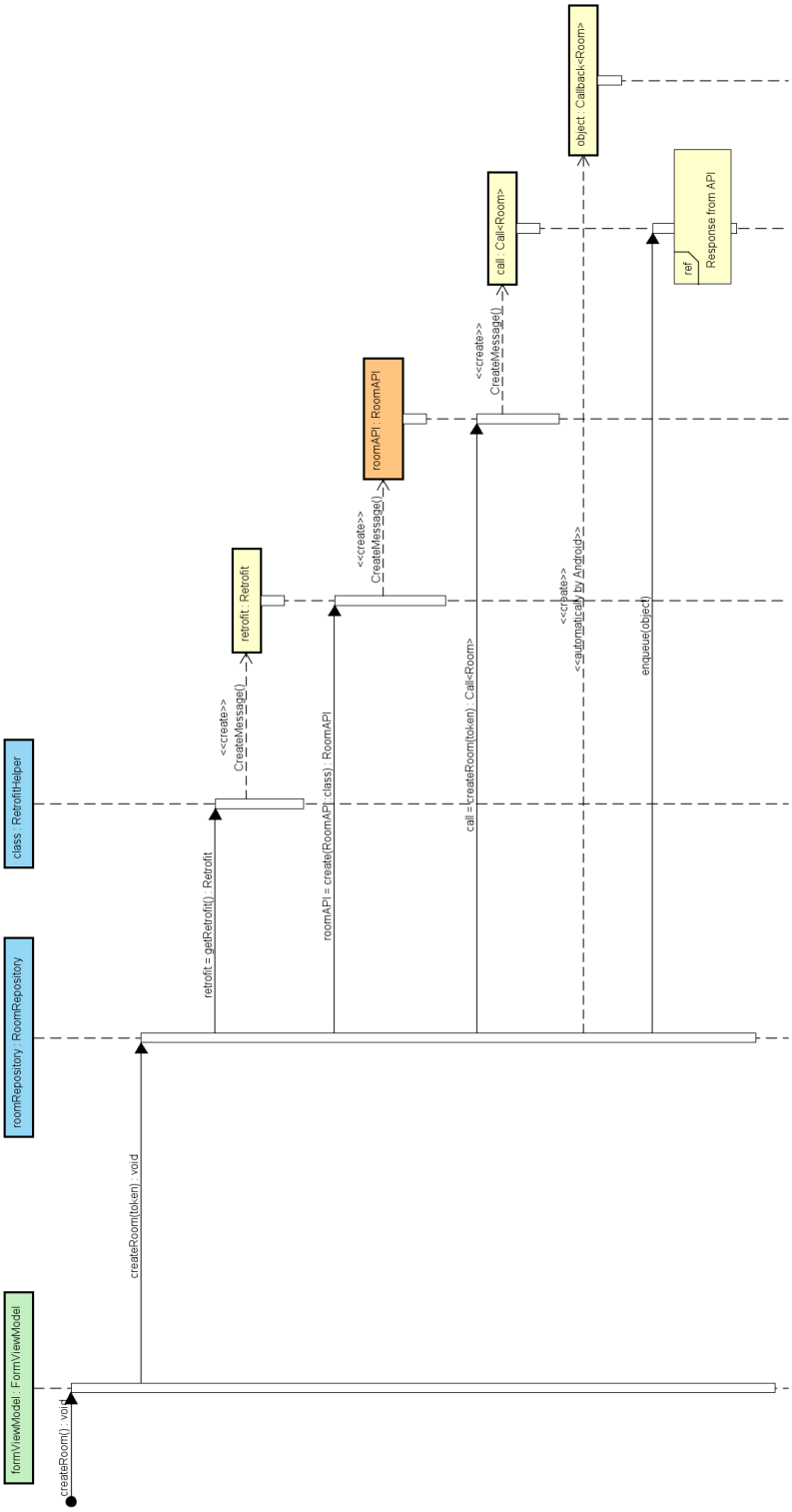

Figura 5.28: Diagrama de secuencia **Crear Sala**

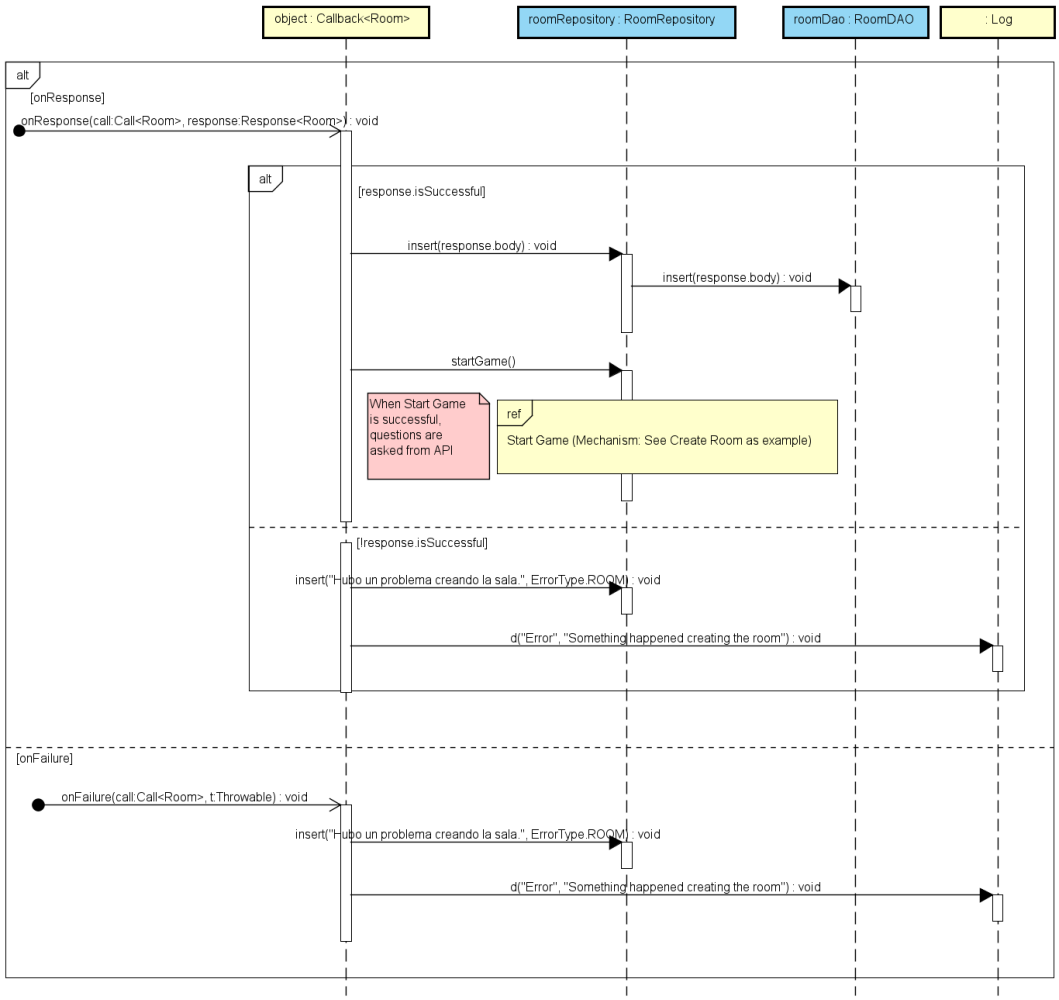

Figura 5.29: Diagrama de secuencia **Respuesta de la API**

moderador, respuestas etc., mientras que MongoDB contiene los datos de las simulaciones necesarias para representar las gráficas. También está enlazado al servicio *Flask* para obtener las recomendaciones. En cuanto al dispositivo *Android*, se observa la aplicación *Crossroad2* y la base de datos local *SQLite*.

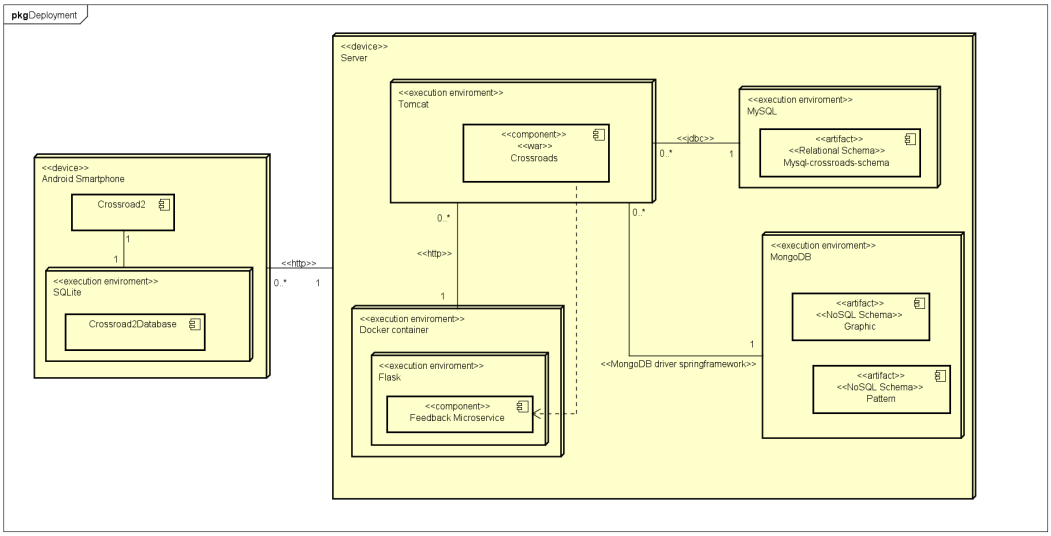

Figura 5.33: Diagrama de despliegue

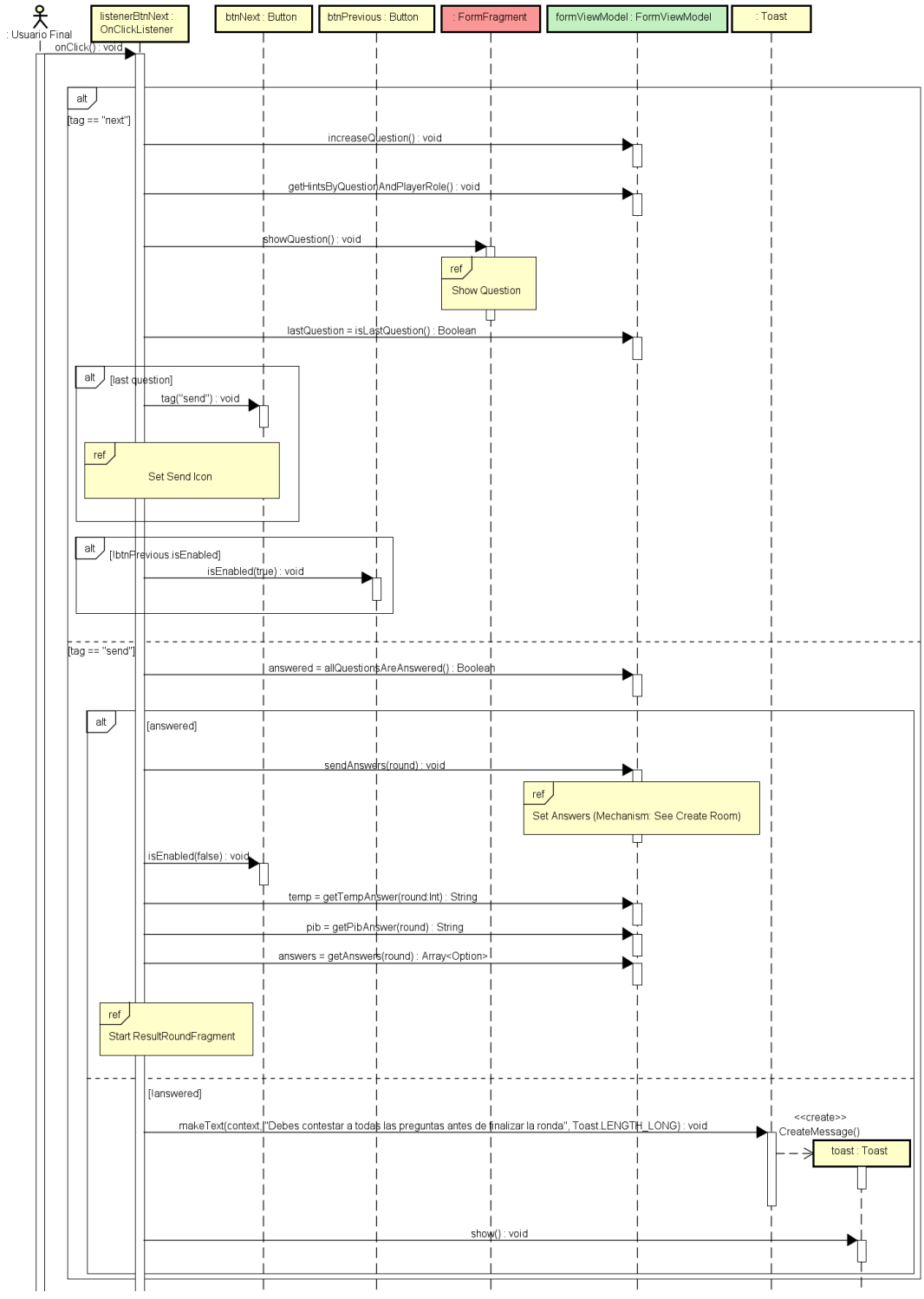

Figura 5.30: Diagrama de secuencia **Listener Botón Avanzar**

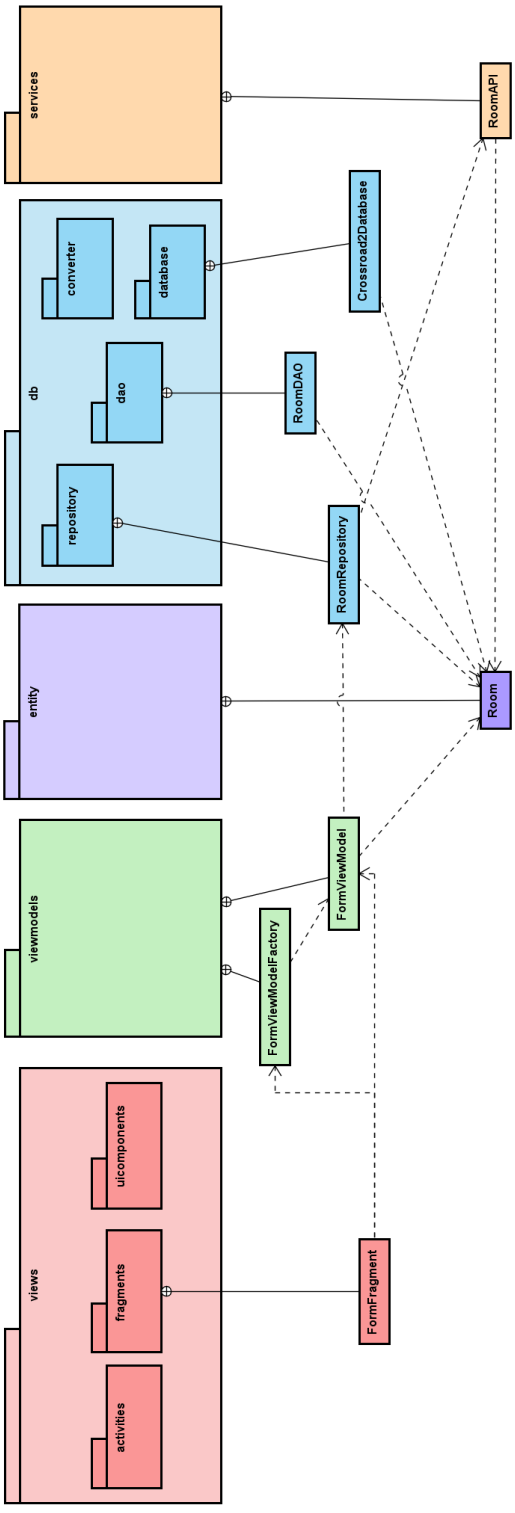

Figura 5.31: Descomposition & Uses Style HU3.1

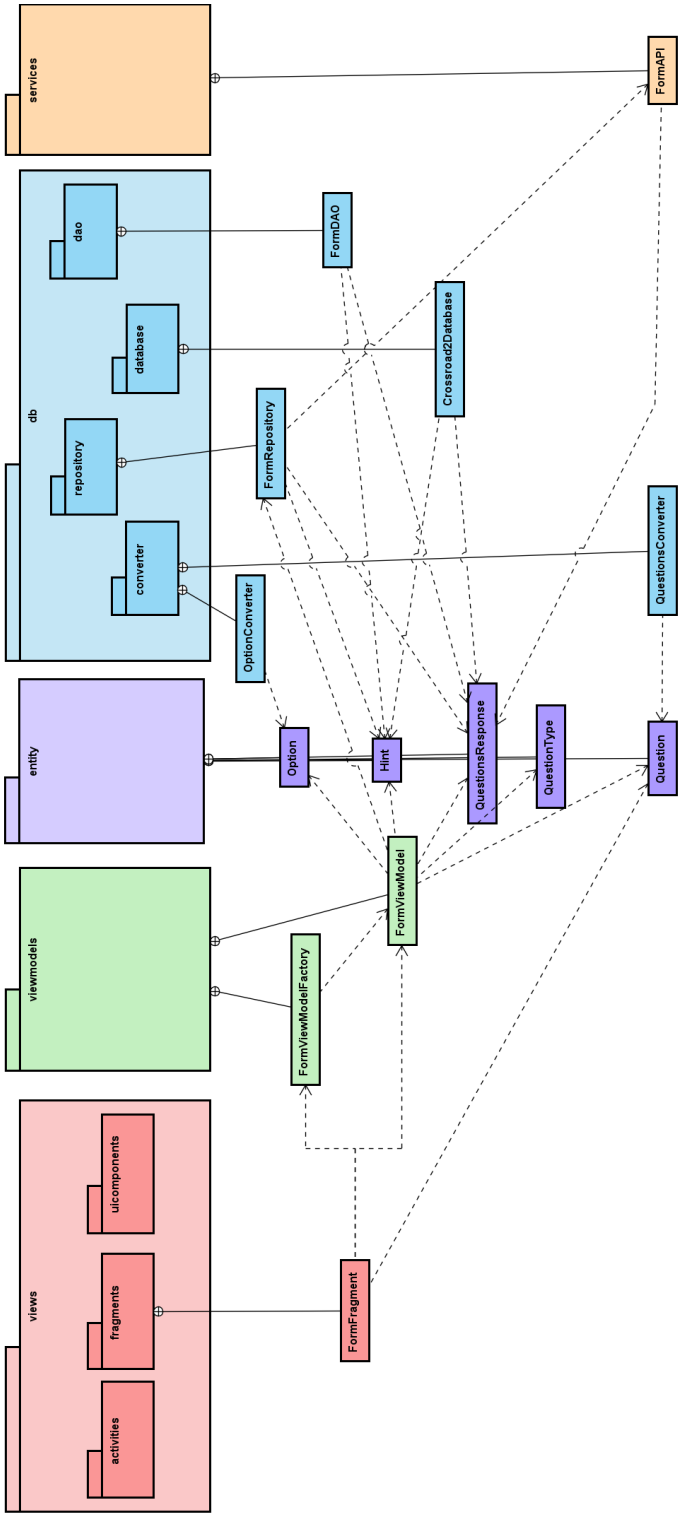

Figura 5.32: Descomposition & Uses Style HU3.1

Capítulo 6

Implementación y pruebas

La implementación de este proyecto se ha realizado utilizando el entorno de desarrollo integrado (IDE) **Android Studio**, el lenguaje de programación **Kotlin** y el gestor de dependencias **Gradle**. Para el diseño de la interfaz de usuario se ha utilizado el lenguaje de etiquetas **XML**.

La estructura del proyecto, como se explicó en la Sección 5.4, está formada por los paquetes mostrados en la Figura 6.1.

El proyecto contiene dos archivos de dependencias. El primero es a nivel de proyecto, donde se añaden opciones de configuración comunes para todos los módulos (en el caso de que hubiera varios) y el segundo es a nivel del módulo de la aplicación, donde se declaran principalmente los *plugins* y las dependencias que utiliza la aplicación, y el código de la versión, que es necesario incrementar cada vez que se quiere actualizar la aplicación en el *Play Store*.

6.1. Cambios realizados durante la implementación

Según avanzaba el desarrollo del proyecto, han surgido algunos cambios debido a falta de experiencia con la arquitectura MVVM y por modificaciones del *backend*. A continuación se detallan los principales cambios:

- **Cambio de una actividad por vista a la arquitectura single-activity**: Como se comenta en la sección 5.1 se ha decidido utilizar una sola actividad que alberga todos los fragmentos. Este cambio fue realizado en el segundo *Sprint*.
- **Transferencia de la lógica de las vistas a los *ViewModels***: Una vez acabada la mayor parte de la implementación, se realizó una revisión para trasladar la lógica de negocio que estaba en las vistas a los *ViewModels*.

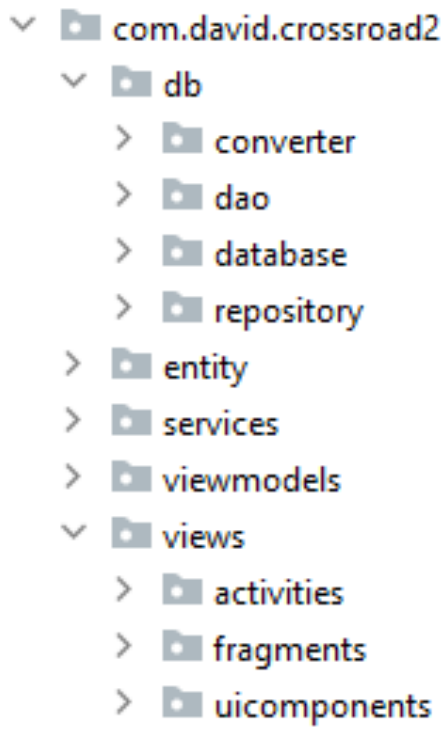

Figura 6.1: Estructura de paquetes del proyecto en el IDE

- **Transferencia de peticiones a APIs a los repositorios**: Se corrigió un error conocido moviendo las peticiones de los *ViewModels* a los repositorios.

- **Cambio en el orden de las preguntas**: Se realizó un cambio en el *backend* para modificar el orden de las preguntas, añadiendo junto a la lista de preguntas, una lista con el orden en el que deben aparecer estas. Debido a esto, hubo que modificar la aplicación para recibir de la base de datos dos listas, la de preguntas y la de orden, y mostrar las preguntas según la disposición especificada. Esta solución se tomó ante la previsión de nuevos reordenamientos de las preguntas para facilitar la evolución.

6.2. Plan de pruebas y evaluación

En esta sección se detallan todos los casos de prueba identificados y realizados durante el desarrollo de la aplicación. Inicialmente se optó por la creación de *tests* automáticos, pero en el segundo *Sprint* se sustituyeron por pruebas manuales, quedando de esta manera mucho más detallados en esta memoria.

Hay diferentes tipos de pruebas según su categoría.

- **Pruebas de sistema** 6.2.1: Se analiza el comportamiento observable. En cada caso de prueba se explican los pasos a seguir para realizar el *test* y el resultado esperado, además de una breve descripción de la prueba.

- **Pruebas de integración** 6.2.1: En este tipo de pruebas se verifica que distintos componentes se relacionan de forma adecuada entre sí para obtener el resultado correcto [47].

- **Pruebas de usabilidad** 6.2.2: El usuario final utiliza la aplicación una vez esté finalizada, y se toma nota de su experiencia, críticas y valoraciones.

- **Pruebas unitarias**: Consisten en aislar partes del código (normalmente por métodos) y comprobar que funciona correctamente.

6.2.1. Casos de prueba

Se ha optado por realizar las pruebas de manera manual, quedando documentadas en esta memoria de la Tabla 6.1 a la 6.47. Por ello, no se realizarán pruebas unitarias. En la subsección 6.2.2 se exponen los resultados obtenidos.

Pruebas de integración

CPI-01	**Comienzo de la partida con la creación de un jugador nuevo en almacenamiento local**
Descripción	Comienzo de la partida. No hay ningún jugador guardado en local y se crea uno aleatorio nuevo.
Esquema del escenario	▪ Se ejecuta por primera vez el método “getPlayer()”.
Resultado esperado	Creación de un jugador y se guarda en el almacenamiento local.

Tabla 6.1: Comienzo de la partida con creación de un jugador

CPI-02	**Comienzo de la partida con la recuperación de un jugador existente**
Descripción	Comienzo de la partida. Ya hay un jugador guardado en el almacenamiento local y se recupera.
Esquema del escenario	▪ Se ejecuta el método "getPlayer()".
Resultado esperado	Recuperación del jugador guardado en el almacenamiento local.

Tabla 6.2: Comienzo de la partida con recuperación de jugador existente

CPI-03	**Comienzo de la partida con la creación de un moderador en almacenamiento local**
Descripción	Comienzo de la partida. No está guardado el moderador en almacenamiento local y se crea.
Esquema del escenario	▪ Se ejecuta por primera vez el método "getModerator()".
Resultado esperado	Creación del moderador y se guarda en el almacenamiento local.

Tabla 6.3: Comienzo de la partida con creación de un moderador

CPI-04	**Comienzo de la partida con la recuperación desde el almacenamiento local del moderador**
Descripción	Comienzo de la partida. Ya hay un moderador guardado en el almacenamiento local y se recupera.
Esquema del escenario	▪ Se ejecuta el método "getModerator()".
Resultado esperado	Recuperación del moderador guardado en el almacenamiento local.

Tabla 6.4: Comienzo de la partida con la recuperación del almacenamiento local del moderador

CPI-05	**Inicio de sesión del moderador**
Descripción	Comienzo de la partida. El moderador inicia sesión en la base de datos.
Esquema del escenario	▪ Se ejecuta el método "loginModeratorInMySQL()" pasándole como argumento el moderador guardado en almacenamiento local. ▪ La API necesita un JSON con los datos de inicio de sesión del moderador en el Body.
Resultado esperado	Inicio de sesión del moderador en la base de datos remota, y obtención del objeto *AuthenticationResponse*, que se guarda en la base de datos local.

Tabla 6.5: Inicio de sesión del moderador

CPI-06	**Creación de sala**
Descripción	Creación de sala para albergar la partida
Esquema del escenario	▪ Se ejecuta el método "createRoom()". ▪ La API necesita el token del moderador en la cabecera.
Resultado esperado	Creación de la sala en la base de datos remota, y obtención del objeto *Room*.

Tabla 6.6: Creación de sala

CPI-07	**Inicio de la partida**
Descripción	Inicio de la partida
Esquema del escenario	▪ Se ejecuta el método "startGame()", pasándole el ID de la sala como argumento. ▪ La API necesita el token del moderador en la cabecera, y el ID de la sala en el Path.
Resultado esperado	Inicio de la partida en la base de datos remota, y obtención del código de respuesta 200.

Tabla 6.7: Inicio de la partida

CPI-08	**Obtención de las preguntas**
Descripción	Obtención de las preguntas con sus respectivas opciones de la base de datos remota.
Esquema del escenario	▪ Se ejecuta el método "getQuestionsfromMySQL()". ▪ La API necesita que se indique en lenguaje en la cabecera.
Resultado esperado	Obtención de las preguntas y almacenamiento de las mismas en la base de datos local.

Tabla 6.8: Inicio de la partida

CPI-09	**Envío de las respuestas de la ronda actual**
Descripción	Envío de las respuestas de la ronda actual a la base de datos.
Esquema del escenario	▪ Se ejecuta el método "setAnswers()" pasándole como argumento la ronda actual. ▪ La API necesita como argumento en el Body un JSON con las respuestas de la ronda actual.
Resultado esperado	Envío de las respuestas a la base de datos remota, y obtención del código de respuesta 200.

Tabla 6.9: Envío de las respuestas de la ronda actual

CPI-10	**Obtención de la puntuación**
Descripción	Obtención de la puntuación del jugador según las respuestas dadas en la ronda actual.
Esquema del escenario	▪ Se ejecuta el método "getScore()". ▪ La API necesita como argumento en el Body un JSON con el número del grupo, código de la sala y ronda.
Resultado esperado	Obtención de las puntuaciones del jugador desde la base de datos remota, que se guardan en la base de datos local.

Tabla 6.10: Obtención de la puntuación

CPI-11	**Obtención de las gráficas**
Descripción	Obtención de los valores para construir las gráficas según las respuestas dadas en la ronda actual, exceptuando las respuestas de tipo *Objetivo*.
Esquema del escenario	▪ Se ejecuta el método "getGraphicByCode()". ▪ La API necesita como argumento en el Body un array con las respuestas de la ronda, exceptuando las de tipo objetivo. Este array se puede conseguir como resultado de ejecutar la operación "getScore()".
Resultado esperado	Obtención de los valores para realizar las gráficas de temperatura y PIB desde la base de datos remota, que se guardan en la base de datos local.

Tabla 6.11: Obtención de las gráficas

CPI-12	**Obtención de la ayuda de cada pregunta**
Descripción	Obtención de las ayuda de cada pregunta del formulario según el rol.
Esquema del escenario	▪ Se ejecuta el método "getRecommendations()". ▪ La API necesita como argumento en el Path el id de la pregunta y el rol del jugador, que irá cambiando de forma aleatoria.
Resultado esperado	Obtención de la ayuda desde la base de datos remota, que se guarda en la base de datos local.

Tabla 6.12: Obtención de la ayuda de cada pregunta

CPI-13	**Obtención de las recomendaciones**
Descripción	Obtención de las recomendaciones entregadas por el recomendador.
Esquema del escenario	▪ Se ejecuta el método "getRecommendations()". ▪ La API necesita como argumento en el Body un JSON con las respuestas de objetivo del PIB, de temperatura, y un array con las demás respuestas.
Resultado esperado	Obtención de las recomendaciones desde la base de datos remota, que se guardan en la base de datos local.

Tabla 6.13: Obtención de la ayuda

CPI-14	**Finalización de la ronda**
Descripción	Finalización de la ronda actual.
Esquema del escenario	▪ Se ejecuta el método "endRound()". ▪ La API necesita como argumento en el Path el id de la sala, y un Header indicando el token de autenticación del moderador.
Resultado esperado	Finalización de la ronda actual y obtención del código de respuesta 200.

Tabla 6.14: Finalización de la ronda

CPI-15	**Avanzar a la ronda siguiente**
Descripción	Avanzar a la ronda siguiente siempre que no sea la última ronda de la partida.
Esquema del escenario	▪ Se ejecuta el método "nextRound()". ▪ La API necesita como argumentos en el Path el id de la sala y el número de ronda a la que se quiere avanzar, y un Header indicando el token de autenticación del moderador.
Resultado esperado	Avanzar a la ronda siguiente y obtención del código de respuesta 200.

Tabla 6.15: Avanzar a la ronda siguiente

CPI-16	**Finalización de la partida**
Descripción	Finalización de la partida después de haber jugado las tres rondas.
Esquema del escenario	▪ Se ejecuta el método "endGame()". ▪ La API necesita como argumento en el Path el id de la sala y un Header indicando el token de autenticación del moderador.
Resultado esperado	Finalización de la partida y obtención del código de respuesta 200.

Tabla 6.16: Finalización de la partida

CPI-17	**Recuperación de las respuestas**
Descripción	Recuperación de las respuestas dadas en cada ronda.
Esquema del escenario	▪ Se ejecuta el método "getPlayerChoicesInRound()". ▪ La API necesita como argumento en el Body el id del jugador y la ronda de la que se solicitan las respuestas.
Resultado esperado	Obtención de las respuestas del jugador para la ronda solicitada.

Tabla 6.17: Recuperación de las respuestas

Pruebas de sistema

CPS-01	**Reproducir vídeo de YouTube**
Descripción	Visualizar correctamente el vídeo explicativo de Crossroads.
Esquema del escenario	▪ Se pulsa el botón con el icono de *YouTube*.
Resultado esperado	Desaparición del icono y reproducción del vídeo de YouTube.

Tabla 6.18: Reproducir vídeo de YouTube

CPS-02	**Visualización de la información del proyecto**
Descripción	Visualización del texto explicativo del juego y los iconos de la Universidad de Valladolid, Escuela de Ingeniería Informática y Fundación Española para la Ciencia y la Tecnología.
Esquema del escenario	▪ Se pulsa el botón con el texto "Sobre el proyecto".
Resultado esperado	Navegación hacia el Fragment de la información del proyecto y correcta visualización del texto y los iconos.

Tabla 6.19: Visualización de la información del proyecto

CPS-03	**Navegación hacia la pantalla inicial**
Descripción	Navegación desde la pantalla de información del proyecto hacia la inicial.
Esquema del escenario	▪ Se pulsa el texto "Atrás" de la parte superior, o el botón de retroceso del dispositivo.
Resultado esperado	Navegación hacia la pantalla inicial de la aplicación.

Tabla 6.20: Navegación hacia la pantalla inicial

CPS-04	**Reproducción del video al regresar de otro Fragment**
Descripción	Reproducción del vídeo explicativo de la pantalla inicial al regresar de otro Fragment.
Esquema del escenario	▪ Se navega a la pantalla de la información del proyecto. ▪ Se retrocede a la pantalla inicial. ▪ Se pulsa el botón con el icono de *YouTube*.
Resultado esperado	Correcta reproducción del vídeo explicativo.

Tabla 6.21: Reproducción del vídeo al regresar de otro Fragment

CPS-05	**Comenzar partida**
Descripción	Visualización de la primera pregunta del cuestionario al pulsar en el botón de comenzar la partida.
Esquema del escenario	▪ Pulsar botón de comenzar partida.
Resultado esperado	Visualización de la pantalla del cuestionario.

Tabla 6.22: Comenzar partida

CPS-06	**Seleccionar una respuesta de una pregunta que no sea hipótesis**
Descripción	Seleccionar una respuesta de una pregunta que no sea hipótesis.
Esquema del escenario	▪ Comenzar partida. ▪ Seleccionar pregunta no hipótesis. ▪ Seleccionar primera opción.
Resultado esperado	Pregunta con opción seleccionada y guardada en local.

Tabla 6.23: Respuesta de pregunta no hipótesis

CPS-07	**Seleccionar una respuesta de una pregunta que sea hipótesis**
Descripción	Seleccionar una respuesta de una pregunta que sea hipótesis.
Esquema del escenario	▪ Comenzar partida. ▪ Seleccionar pregunta hipótesis. ▪ Seleccionar primera opción.
Resultado esperado	Pregunta con opción seleccionada y guardada en local.

Tabla 6.24: Respuesta de pregunta hipótesis

CPS-08	**Avanzar a la siguiente pregunta**
Descripción	Avanzar a la siguiente pregunta del formulario, siempre que la pregunta actual no sea la última.
Esquema del escenario	▪ Comenzar partida. ▪ Avanzar a la siguiente pregunta.
Resultado esperado	Visualización de la siguiente pregunta.

Tabla 6.25: Avanzar a la siguiente pregunta

CPS-09	**Retroceder a la pregunta anterior**
Descripción	Retroceder a la pregunta anterior del formulario, siempre que la pregunta actual no sea la primera.
Esquema del escenario	▪ Comenzar partida. ▪ Avanzar a la siguiente pregunta. ▪ Retroceder a la pregunta anterior.
Resultado esperado	Visualización de la primera pregunta, con el botón de retroceder pregunta deshabilitado.

Tabla 6.26: Retroceder a la pregunta anterior

CPS-10	**Finalizar cuestionario correctamente**
Descripción	Avanzar hasta la última pregunta habiendo contestado todas las preguntas para observar que el botón de Avanzar ha cambiado de icono para finalizar la ronda.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Avanzar a la última pregunta. ▪ Pulsar botón avanzar/finalizar.
Resultado esperado	El icono del botón ha cambiado, y al pulsarlo se dirige a la pantalla de resultados de ronda.

Tabla 6.27: Finalizar cuestionario correctamente

CPS-11	**Finalizar cuestionario no correctamente**
Descripción	Avanzar hasta la última pregunta sin haber contestado todas las preguntas.
Esquema del escenario	▪ Comenzar partida. ▪ No contestar todas las preguntas. ▪ Avanzar a la última pregunta. ▪ Pulsar botón avanzar/finalizar.
Resultado esperado	El icono del botón ha cambiado, y al pulsarlo se muestra un mensaje indicando que se debe contestar a todas las preguntas para poder ver los resultados.

Tabla 6.28: Finalizar cuestionario no correctamente

CPS-12	**Botón anterior deshabilitado**
Descripción	Botón de retroceder a la pregunta anterior deshabilitado al iniciar el cuestionario, debido a estar situado en la primera pregunta y no haber ninguna pregunta anterior e esta.
Esquema del escenario	▪ Comenzar partida.
Resultado esperado	Visualización de la primera pregunta, mostrando el botón con texto de “Anterior” deshabilitado.

Tabla 6.29: Botón Anterior deshabilitado

CPS-13	**Modificar respuesta de pregunta no hipótesis**
Descripción	Modificar la respuesta marcada en una ronda previa de cualquier pregunta que no sea una hipótesis.
Esquema del escenario	▪ Comenzar partida. ▪ Seleccionar pregunta no hipótesis. ▪ Marcar primera opción. ▪ Avanzar a la siguiente ronda. ▪ Seleccionar pregunta no hipótesis. ▪ Seleccionar segunda opción.
Resultado esperado	Visualización de la pregunta no hipótesis con la nueva opción seleccionada y guardada en almacenamiento local.

Tabla 6.30: Modificar respuesta de pregunta no hipótesis

CPS-14	**Imposibilidad de modificar respuesta de pregunta hipótesis**
Descripción	Imposibilidad de modificar la respuesta marcada en una ronda anterior de cualquier pregunta que sea una hipótesis.
Esquema del escenario	▪ Comenzar partida. ▪ Seleccionar pregunta hipótesis. ▪ Marcar primera opción. ▪ Avanzar a la siguiente ronda. ▪ Seleccionar pregunta hipótesis.
Resultado esperado	Visualización de la pregunta hipótesis con la primera opción seleccionada y todas ellas deshabilitadas ya que no se puede volver a seleccionar una nueva.

Tabla 6.31: Imposibilidad de modificar respuesta de pregunta hipótesis

CPS-15	**Seleccionar la primera opción de las dos primeras preguntas**
Descripción	Seleccionar la primera opción de las dos primeras preguntas del formulario.
Esquema del escenario	▪ Comenzar partida. ▪ Marcar primera opción de la primera pregunta. ▪ Avanzar a la siguiente pregunta. ▪ Marcar primera opción de la segunda pregunta.
Resultado esperado	Visualización de la segunda pregunta con la primera opción seleccionada.

Tabla 6.32: Seleccionar la primera opción de las dos primeras preguntas

CPS-16	**Visualización de la pantalla de resultados**
Descripción	Visualización de la pantalla de resultados.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Avanzar a la última pregunta. ▪ Pulsar botón avanzar/finalizar.
Resultado esperado	Visualización de las gráficas de predicciones de temperatura y PIB y las puntuaciones del jugador.

Tabla 6.33: Visualización de la pantalla de resultados

CPS-17	**Visualización de la pantalla de recomendaciones**
Descripción	Visualización de la pantalla de recomendaciones.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Avanzar a la última pregunta. ▪ Pulsar botón avanzar/finalizar. ▪ Pulsar el botón recomendaciones.
Resultado esperado	Visualización del texto de las recomendaciones.

Tabla 6.34: Visualización de la pantalla de recomendaciones

CPS-18	**Visualización de la pantalla de información sobre la puntuación**
Descripción	Visualización de la pantalla de información sobre como se calcula la puntuación obtenida.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Avanzar a la última pregunta. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el icono o texto de puntuaciones.
Resultado esperado	Visualización del texto de la información sobre la puntuación.

Tabla 6.35: Visualización de la pantalla de información sobre la puntuación

CPS-19	**Visualización de la pantalla de ayuda de cada pregunta**
Descripción	Visualización de la pantalla de ayuda de cada pregunta del formulario.
Esquema del escenario	▪ Comenzar partida. ▪ Pulsar el icono o texto de Ayuda.
Resultado esperado	Visualización de la pantalla de ayuda, con un vídeo, una pista, y un botón para más ayuda en la parte inferior.

Tabla 6.36: Visualización de la pantalla de ayuda de cada pregunta

CPS-20	**Visualización del resumen de las respuestas**
Descripción	Visualización de la pantalla con el resumen de las respuestas.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo.
Resultado esperado	Visualización de la pantalla con el resumen de las respuestas.

Tabla 6.37: Visualización del resumen de las respuestas

CPS-21	**Modificación de las respuestas (No hipótesis)**
Descripción	Posibilidad de modificar las respuestas elegidas que no sean hipótesis para no tener que responder todas de nuevo en las siguientes rondas.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Seleccionar una pregunta que se quiera modificar. ▪ Elegir otra respuesta. ▪ Pulsar el botón de aceptar.
Resultado esperado	Visualización de la pantalla con el resumen de las respuestas con la nueva opción guardada.

Tabla 6.38: Modificación de las respuestas

CPS-22	**No modificación de las respuestas (Hipótesis)**
Descripción	Para las preguntas que sean de tipo hipótesis se pueden visualizar para ver la respuesta seleccionada pero no se pueden modificar.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Seleccionar una pregunta que se quiera modificar. ▪ Elegir otra respuesta.
Resultado esperado	Visualización de la pantalla con la pregunta hipótesis con la respuesta original seleccionada, sin posibilidad de modificarla.

Tabla 6.39: No modificación de las respuestas (Hipótesis)

CPS-23	**Visualización de los resultados de las rondas posteriores a la primera**
Descripción	Visualización de los resultados de las rondas posteriores a la primera.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Modificar las respuestas deseadas. ▪ Pulsar botón comprobar respuestas.
Resultado esperado	Visualización de la pantalla de los resultados de ronda con las nuevas puntuaciones.

Tabla 6.40: Visualización de los resultados de las rondas posteriores a la primera

CPS-24	**Visualización de un *pop-up* de advertencia al no modificar ninguna respuesta**
Descripción	Visualización de un *pop-up* de advertencia al pulsar el botón de comprobar respuestas al no modificar ninguna.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Pulsar botón comprobar respuestas.
Resultado esperado	Visualización de un *pop-up* advirtiendo de que se no se ha modificado ninguna respuesta y se consumirá un intento.

Tabla 6.41: Visualización de un *pop-up* de advertencia al no modificar ninguna respuesta

CPS-25	**Confirmación del envío de las respuestas en el *pop-up* de advertencia**
Descripción	Confirmar el envío de las respuestas en el *pop-up* de advertencia mostrado al no modificar ninguna respuesta.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Pulsar botón comprobar respuestas. ▪ Pulsar botón confirmar.
Resultado esperado	Visualización de la pantalla de los resultados de la ronda.

Tabla 6.42: Confirmación del envío de las respuestas en el *pop-up* de advertencia

CPS-26	**Cancelar el envío de las respuestas en el *pop-up* de advertencia**
Descripción	Cancelar el envío de las respuestas en el *pop-up* de advertencia mostrado al no modificar ninguna respuesta.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas. ▪ Pulsar el botón de finalizar cuestionario. ▪ Pulsar el botón de intentar de nuevo. ▪ Pulsar botón comprobar respuestas. ▪ Pulsar botón cancelar.
Resultado esperado	Visualización de la pantalla de las respuestas con la posibilidad de cambiar las respuestas antes de enviarlas.

Tabla 6.43: Cancelar el envío de las respuestas en el *pop-up* de advertencia

CPS-27	**Terminar partida**
Descripción	Terminar partida y visualización de los resultados finales.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas de las tres rondas. ▪ Pulsar el botón de terminar partida.
Resultado esperado	Visualización de la pantalla de los resultados finales.

Tabla 6.44: Terminar partida

CPS-28	**Visualización de la mejor ronda**
Descripción	Visualización de la pantalla de resultados finales con la mejor ronda resaltada.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas de las tres rondas. ▪ Pulsar el botón de terminar partida.
Resultado esperado	Visualización de la pantalla de resultados finales con la mejor ronda resaltada.

Tabla 6.45: Visualización de la mejor ronda

CPS-29	**Volver a la pantalla inicial**
Descripción	Volver a la pantalla inicial.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas de las tres rondas. ▪ Pulsar el botón de terminar partida. ▪ Pulsar el botón de volver a la pantalla inicial.
Resultado esperado	Visualización de la pantalla inicial.

Tabla 6.46: Volver a la pantalla inicial

CPS-30	**Visualización de las respuestas dadas**
Descripción	Visualización de las respuestas dadas en la ronda seleccionada.
Esquema del escenario	▪ Comenzar partida. ▪ Contestar todas las preguntas de las tres rondas. ▪ Pulsar el botón de terminar partida. ▪ Pulsar en las puntuaciones de la primera ronda.
Resultado esperado	Visualización de las respuestas dadas en la primera ronda.

Tabla 6.47: Visualización de las respuestas dadas

6.2.2. Resultados de las pruebas

En esta subsección se analizan los resultados obtenidos de todos los casos de prueba explicados en el apartado anterior.

Caso de Prueba	Resultado obtenido
CPI-01	▪ Prueba superada con éxito
CPI-02	▪ Prueba superada con éxito
CPI-03	▪ Prueba superada con éxito
CPI-04	▪ Prueba superada con éxito
CPI-05	▪ Prueba superada con éxito
CPI-06	▪ Inicialmente no se borraba el anterior *Authentication-Response* recibido al iniciar sesión con el moderador, por lo que cuando el token cambiaba, la operación *createRoom()* fallaba al no recibir un token válido. ▪ Posteriormente se eliminaron los tokens previos antes de recibir el actual, y el test funcionó correctamente.
CPI-07	▪ Prueba superada con éxito
CPI-08	▪ Prueba superada con éxito

CPI-09	▪ Prueba superada con éxito
CPI-10	▪ Prueba superada con éxito
CPI-11	▪ Prueba superada con éxito
CPI-12	▪ Esta prueba al principio no era exitosa debido a que *Retrofit* no devolvía el objeto correctamente al intentar transformar de *Json* a objeto mediante *Gson*. ▪ Finalmente se utilizó la factoría *Scalars*, que permite obtener texto plano y convertirlo a *String*, y el test se realizó con éxito.
CPI-13	▪ Prueba superada con éxito
CPI-14	▪ Prueba superada con éxito
CPI-15	▪ Prueba superada con éxito
CPI-16	▪ Esta prueba al principio no era exitosa debido a que se intentaba finalizar la partida sin haber concluido la última ronda. Esto permitió identificar y solucionar dicho error para terminar todas las rondas, ya que no se llamaba al método necesario para terminar la tercera ronda.
CPI-17	▪ Prueba superada con éxito

CPS-01	▪ Prueba superada con éxito
CPS-02	▪ Prueba superada con éxito
CPS-03	▪ Prueba superada con éxito
CPS-04	▪ Inicialmente la prueba no se superaba con éxito debido a que se intentaba volver a instanciar un elemento con el mismo ID que otro existente (el reproductor de *YouTube*) ▪ Se optó por eliminar el reproductor al salir de este *Fragment* para que al regresar no existiera ese objeto y se creara otro sin problemas. De esta forma la prueba fue superada.
CPS-05	▪ Algunas veces se producía un error por tener los identificadores de las preguntas duplicados en la base de datos local *Room*. Esto se debía a que en ocasiones se llamaba dos veces al *fragment* que muestra el cuestionario, y dos veces al *ViewModel* y a todas sus operaciones. ▪ Esto se solucionó creando el *ViewModel* de *HomeFragment* desde el inicio de este, y en ese momento se realizan todas las llamadas necesarias previas a iniciar partida, en vez de hacerlo cuando se le da al botón de comenzar.
CPS-06	▪ Prueba superada con éxito
CPS-07	▪ Prueba superada con éxito

CPS-08	▪ Prueba superada con éxito
CPS-09	▪ Prueba superada con éxito
CPS-10	▪ Prueba superada con éxito
CPS-11	▪ Prueba superada con éxito
CPS-12	▪ Prueba superada con éxito
CPS-13	▪ Prueba superada con éxito
CPS-14	▪ Prueba superada con éxito
CPS-15	▪ Al principio no era posible seleccionar dos respuestas que estuvieran en la misma posición en dos preguntas consecutivas, debido a que se consideraba que ya estaba marcada aunque no se indicara de esa forma visualmente. ▪ La solución fue utilizar una opción por defecto que no se visualiza en la pantalla del formulario y marcar esa al avanzar a otra pregunta.
CPS-16	▪ Prueba superada con éxito
CPS-17	▪ Prueba superada con éxito

CPS-18	▪ Prueba superada con éxito
CPS-19	▪ Prueba superada con éxito
CPS-20	▪ Prueba superada con éxito
CPS-21	▪ Prueba superada con éxito
CPS-22	▪ Prueba superada con éxito
CPS-23	▪ Inicialmente se visualizaban las gráficas correctamente, pero las puntuaciones de la primera ronda en vez de las actuales. ▪ Esto se debía a que de la base de datos local, se obtenía una única puntuación en vez de una lista de puntuaciones, y se devolvía la de la primera ronda. ▪ La solución fue traer una lista de *Score* en vez de una sola puntuación.
CPS-24	▪ Prueba superada con éxito
CPS-25	▪ Prueba superada con éxito
CPS-26	▪ Prueba superada con éxito
CPS-27	▪ Prueba superada con éxito

CPS-28	▪ Al principio se producía un error al querer comenzar una nueva partida debido a que ya había registros guardados con el mismo identificador (en el caso de las preguntas). ▪ Esto se solucionó eliminando todos los datos de todos los repositorios al abrir el fragmento de inicio.
CPS-29	▪ Prueba superada con éxito
CPS-30	▪ Prueba superada con éxito

Tabla 6.48: Resultados de los casos de prueba

Pruebas de usabilidad

Una vez finalizada la aplicación, se han realizado las pruebas de usabilidad para que pueda ser examinada por usuarios finales y comprobar si todo funciona correctamente y si hay algún aspecto que mejorar. Para ello se realizó un formulario con el fin de obtener información sobre la opinión de los usuarios de distintos aspectos de la aplicación.

Las pruebas han consistido en que los usuarios seleccionados utilicen la aplicación libremente, con el fin de que sin ningún tipo de ayuda ni conocimiento previo sobre el proyecto consigan entender el objetivo de la aplicación y realizar una partida completa para obtener los mejores resultados posibles.

El cuestionario de usabilidad SUS (*System Usability Scale)* estándar de [72] proporciona una medición rápida sobre la usabilidad del sistema.

Las preguntas realizadas en él son las siguientes:

1. Me gustaría utilizar este sistema con frecuencia.
2. Encuentro que el sistema es innecesariamente complejo.
3. Pienso que el sistema es fácil de usar.
4. Pienso que necesito el apoyo de personal técnico o recurrir a manuales para poder utilizar este sistema.
5. Las funciones de la versión actual se encuentran bien integradas.

6. Existen inconsistencias en la versión actual de la aplicación.
7. Creo que la mayoría de usuarios aprenderían a utilizar rápidamente el sistema en su versión actual.
8. Encuentro el sistema incómodo de utilizar.
9. Me he sentido seguro usando el sistema.
10. Se necesita aprender muchas cosas antes de poder utilizar el sistema .

Todas las respuestas a estas preguntas siguen la Escala de Likert [61], con valores del 1 al 5, siendo 1 nada de acuerdo y 5 totalmente de acuerdo.

Además de las preguntas del test de usabilidad SUS, se realizan otras para saber el perfil del usuario (actividad principal, edad, género y experiencia previa con el juego) y una de respuesta abierta para cualquier comentario o sugerencia sobre la aplicación.

Resultados de las pruebas de usabilidad

En la Tabla 6.49 se muestran los resultados del formulario contestado por 7 personas. Los identificadores P1 a P10 hacen referencia al identificador de cada pregunta. En la primera columna se muestra el código identificativo de cada usuario que ha respondido el cuestionario, en la segunda la actividad del encuestado y en la tercera el nivel de experiencia previa con el juego, siendo 1 ningún tipo de experiencia y 5 mucha. En las demás celdas se muestran las respuestas de cada persona a las preguntas del formulario.

ID	Actividad	Exp.	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
U01	Profesor	5	3	4	2	4	3	2	3	3	2	5
U02	Profesor	5	5	1	2	2	3	3	1	1	1	3
U03	Estudiante	3	4	1	4	2	5	1	4	1	5	1
U04	Estudiante	2	4	4	5	4	4	2	2	1	2	4
U05	Estudiante	1	3	1	4	5	4	5	4	2	4	3
U06	Estudiante	3	4	2	4	1	5	1	5	1	4	2
U07	Estudiante	3	2	1	5	1	5	1	5	1	5	2

Tabla 6.49: Respuestas del formulario SUS

Sobre las respuestas del cuestionario SUS, se ha calculado la mediana y la moda para cada una de ellas y posteriormente la puntuación para evaluar la usabilidad del sistema [56].

Las preguntas impares son las que sería positivo que el usuario esté completamente de acuerdo, y las pares lo contrario. La manera de calcular la puntuación es restar uno a la valoración dada en las preguntas impares, y para las preguntas pares restar cinco menos la nota dada. Por último se suman los nuevos valores y se multiplica el resultado por 2.5 [56].

El promedio de la puntuación de todos los usuarios es de 67.5, es decir, el sistema tiene una usabilidad aceptable pero se podría mejorar. Algunas de las sugerencias dadas en la pregunta de respuesta abierta son que el proceso del juego debería ser más dinámico, las pistas no son demasiado útiles y que algunos indicadores son demasiado específicos.

Un estudio en el que se analizaron los datos de 5000 usuarios en 500 evaluaciones estableció el promedio de las puntuaciones para los tests de usabilidad en 68 puntos [41], por lo que la puntuación obtenida es aproximadamente la media para este tipo de tests.

La cuestión con peor valoración es la número 10, en la que los usuarios piensan que es necesario adquirir información antes de jugar, ya que la mediana de las respuestas es 3. Para las demás preguntas las valoraciones son las esperadas, con una nota mediana de 4 para todas las preguntas impares, y 1 o 2 para las demás preguntas pares.

6.3. Licencia

Este proyecto utiliza la **Licencia MIT** o licencia X11, que es una licencia software creada por el Instituto Tecnológico de Massachusetts. Esta permite la modificación, publicación o distribución del proyecto, ya que la intención es que este proyecto continúe después de finalizar este Trabajo de Fin de Grado. Esta licencia se encuentra en el repositorio de *GitLab*, cuyo enlace está en el apéndice B.

Capítulo 7

Seguimiento del proyecto

7.1. Introducción

En este capítulo se realizará el seguimiento del proyecto, presentando lo realizado en cada *Sprint*. Para todos ellos, se hará un resumen en forma de tabla con la siguiente información:

- **Historia de usuario**: El identificador de la historia de usuario a realizar.
- **Tareas**: Las actividades a realizar en esa historia de usuario.
- **Puntos de historia estimados**: El tiempo considerado que se empleará.
- **Tiempo empleado**: El tiempo que realmente se ha invertido.
- **Detalles**: Indica la situación de la historia de usuario, por ejemplo "No comenzado" o "Finalizado".

Si la labor no corresponde con una historia de usuario, se podrá un título representativo como por ejemplo "Documentación" o si no, un "-", y en la columna de "Tareas" se describirá la actividad a realizar. En estos casos, no se realizará una estimación del tiempo.

7.2. Seguimiento de los Sprints

7.2.1. Sprint 0 (02/02/2022 - 10/02/2022)

Este primer *Sprint* se ha dedicado a realizar la preparación inicial del proyecto, con todo lo relativo a la planificación del mismo (Capítulo 2). A continuación, se explicarán las tareas desarrolladas:

- **Scrum**: Explicación del marco de trabajo elegido y sus principales características.
- **Adaptación a Scrum**: Cambios realizados al marco de trabajo para ajustarla a las características del proyecto.
- **Product Backlog inicial**: Identificación de los *stakeholders*, versión inicial del *Product Backlog* con épicas y su descomposición en historias de usuario.
- **Planificación inicial**: Establecimiento del número de *Sprints* planeados y de las horas que se dedicarán a cada uno. También se fijaron los puntos de historia como medida del esfuerzo y se realizó un calendario con los *Sprints*.
- **Riesgos**: Definición de riesgo y las diferentes características que hay que tener en cuenta para que no afecten negativamente al proyecto. También se expuso el análisis de los riesgos encontrados.
- **Presupuestos**: Se realizó el análisis del presupuesto simulado y el real.
- **Tablero GitLab**: Se preparó el tablero que se utilizará para realizar el seguimiento de las tareas desarrolladas en GitLab.
- **Model View ViewModel**: Se buscó información y ejemplos acerca de la arquitectura que se utiliza en *Android* para facilitar el inicio del desarrollo en el *Sprint* 1.
- **Binding**: Por último, se examinó la forma más moderna y utilizada de enlazar elementos de la vista con el código en *Android*.

El tiempo empleado para realizar estas tareas fue de **33 horas** como se mencionó en la planificación inicial 2.5.

7.2.2. Sprint 1 (10/02/2022 - 24/02/2022)

En este *Sprint* se han corregido aspectos observados en la reunión semanal con la tutora de la planificación inicial realizada en la anterior fase, y añadido algunos otros que faltaban como los *Stakeholders* o el modelo de usuario esperado que use la aplicación. También se ha empezado a plantear la arquitectura inicial, con el desarrollo de los diagramas de despliegue, de arquitectura general, modelo de dominio y modelo de proceso de negocio.

Por otro lado, gran parte del tiempo empleado en este *Sprint* ha sido para el despliegue del *Backend* existente, debido al gran numero de dependencias y complejidad del mismo. Surgieron numerosos problemas que fueron resueltos con éxito gracias a otros compañeros del equipo de *Crossroads*.

Adicionalmente ha sido necesario aprender a realizar pruebas de instrumentación en *Android* con *Espresso*, que es un *framework* desarrollado por *Google* para desarrollar tests de la interfaz de usuario.

Por último, antes de empezar con los requisitos a desarrollar en este *Sprint*, se ha implementado la pantalla inicial que se muestra en la aplicación mientras se carga, denominada *Splash Screen*.

Todo lo mencionado ha sido desarrollado en 14 horas y ha servido como base antes de poder implementar las tareas que se encontraban en el *Sprint Backlog*.

En la Tabla 7.1 se mostrarán los avances realizados para cada una de las historias de usuario planificadas para este *Sprint* y las tareas explicadas anteriormente.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU2.1	▪ Diseño ▪ Implementación ▪ Pruebas	1 Punto	7 Horas	Finalizado
HU2.2	▪ Diseño ▪ Implementación ▪ Pruebas	1 Punto	4 Horas	Finalizado
HU2.3	▪ Diseño ▪ Implementación ▪ Pruebas	1 Punto	1 Hora	Finalizado
HU2.4	▪ Diseño ▪ Implementación ▪ Pruebas	1 Punto	1 Hora	Finalizado

HU3.1	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	5 Puntos	7 Horas	Análisis inicial realizado.
HU3.2	▪ Implementación ▪ Pruebas	1 Punto	2 Horas	Finalizado
-	▪ Despliegue Backend	-	10 Horas	Finalizado
-	▪ Formación Tests con Espresso	-	3 Horas	Finalizado
-	▪ Splash Screen	-	1 Horas	Finalizado
Resumen			**36 Horas**	**8/9 Finalizadas**

Tabla 7.1: Tareas realizadas en el Sprint 1

En cuanto a la historia de usuario 2.1, cabe destacar que fue necesario cambiar la forma de poder reproducir un vídeo de *YouTube* por la que se había optado en un principio. La implementación de la arquitectura *Model View ViewModel* obliga a que la actividad que contenga el reproductor implemente *AppCompatActivity* o *Fragment* para poder crear el *ViewModel*, ya que requiere un argumento que sea un *LifecycleOwner* (mediante la palabra reservada *this*). Al utilizar *YoutubeBaseActivity* [17], era necesario implementar esta clase en lugar de una de las dos anteriores, y como consecuencia *this* ya no hacia referencia a un *LifecycleOwner*, si no que simplemente representaba la actividad. Finalmente se pudo conseguir mediante el uso de *YouTubePlayerFragment*, que hereda de *Fragment*.

En total se han empleado para la realización de este *Sprint* **36 Horas**, de las cuales 22

horas corresponden a la implementación de historias de usuario, y 14 horas a otras tareas como el despliegue del *backend* o documentación para poder realizar la aplicación.

7.2.3. Sprint 2 (24/02/2022 - 10/03/2022)

Como en las dos semanas anteriores no fue posible finalizar la historia de usuario 3.1, fue la prioridad de este *Sprint* intentar finalizarla, y se estimó en 2 puntos de historia las tareas pendientes, que consistían principalmente en modificar algunos detalles del modelo de dominio y la máquina de estados realizados en el primer *Sprint*, y la realización de toda la implementación de la historia de usuario, test y diseño, dejando el diagrama de secuencia para más adelante, debido a que uno de los objetivos de este *Sprint* es cambiar la navegación entre las distintas vistas de la aplicación. Actualmente cada pantalla es una *Activity* y se desea transformar estas en *Fragments*. Estos últimos, tienen ventajas como una mejor reutilización de las vistas, diseños más dinámicos y flexibles y son más ligeros que las *Activities*.

El último objetivo de este *Sprint* será implementar la funcionalidad de guardar un usuario moderador en el almacenamiento interno y poder iniciar sesión con él para posteriormente unirse a una sala.

Todas estas tareas se muestran en la Tabla 7.2.

También se completó la máquina de estados con las transiciones y eventos que faltaban.

La mayor complicación de este *Sprint* residía en la asincronía de las consultas a la base de datos. Debido a que las operaciones con el *backend* son asíncronas, no podía devolver el resultado de estas acciones como resultado de la función. Después de múltiples intentos, la solución llevada a cabo fue guardar los productos de estas consultas en la base de datos local *Room* [9]. De esta forma, al implementar la arquitectura *Model View ViewModel*, se puede observar esta base de datos local, para que se notifique cuando haya un cambio en la misma, es decir, cuando se obtenga el resultado de la llamada a la base de datos.

Antes de guardar las preguntas en la base de datos local, es necesario borrar el contenido anterior si lo hubiera, para no obtener un error por contener entradas con ID duplicado. Finalmente, se optó por eliminar las preguntas almacenadas en la misma al iniciarse el repositorio, de esta forma cuando se le solicitara insertar unas nuevas, la base de datos estará vacía y no habrá problema. Se desestimó la opción de borrarlas al final de cada partida, porque si se cierra la aplicación sin haber finalizado el juego, esas preguntas nunca se borrarán.

Sin embargo, surgió un inconveniente con la restricción de identificadores únicos de las preguntas después de haber guardado al moderador en la base de datos. Al comenzar la partida, se observó que se producía dos veces el borrado y la inserción de las preguntas. Después de inspeccionar durante horas el código, observé en el dispositivo móvil, que se abría dos veces la actividad del formulario, y por tanto se creaban dos actividades, dos *ViewModels* y dos repositorios de preguntas y se realizaban dos veces las operaciones de borrado y almacenamiento de las cuestiones. Esto se producía porque desde la actividad se observan tres valores del *ViewModel*, la creación del jugador, la del moderador y el inicio de

sesión del mismo en la base de datos remota, y se avanzaba a la actividad del cuestionario, cuando dentro de esos observadores, alguno de ellos tuviera los tres objetos inicializados (el último que recibiera el objeto que estaba observando). El problema es que en algunas ocasiones dos de ellos cumplían la condición si obtenían el objeto prácticamente al mismo tiempo, produciéndose así el fallo. Esto se solucionó creando el *ViewModel* cuando se inicia la actividad, y no cuando se desea avanzar a la pantalla del formulario, de esta forma ya se tendrán los tres objetos cuando se desee comenzar el cuestionario.

Otro de los problemas ha sido la reproducción del vídeo de *YouTube* en el *Fragment*. Al volver a esa vista desde otra, salta una excepción porque intenta volver a crear el reproductor, y como ya existe un elemento con ese mismo ID (él mismo la primera vez que se accede a esa pantalla), se produce un error y se cierra la aplicación. Quedará pendiente la resolución de dicho fallo para el *Sprint* 3.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU3.1	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 Puntos	26 Horas 40 Minutos	Falta diagrama de secuencia
HU3.3	▪ Implementación ▪ Pruebas	1 Punto	1 Hora	Finalizado
Épica 1	▪ Guardar e iniciar sesión con moderador	2 Puntos	7 Horas	Finalizado
-	▪ Refactorización de Activities a Fragments	3 Puntos	8 Horas	Error con el Fragment de YouTube

-	▪ Test manuales de HU del Sprint 1	-	1 Hora 40 Minutos	Finalizado
Resumen			**44 Horas y 20 Minutos**	**3/5 Finalizadas**

Tabla 7.2: Tareas realizadas en el Sprint 2

Por último, se ha decidido sustituir la automatización de los tests por su implementación de manera manual, como se puede ver en la sección 6.2. Se han realizado las pruebas de la funcionalidad desarrollada en este *Sprint*, y se han adaptado las del anterior.

Para calcular el tiempo empleado en la realización de este *Sprint*, hay que sumar 2 horas y 40 minutos empleadas en la realización de los nuevos tests y modificación de los ya realizados para adaptarlos a la realización manual de los mismos. Por tanto, el tiempo invertido en el desarrollo del proyecto durante estas dos semanas ha sido de **44 horas y 20 minutos**, un poco más de lo estimado por cada etapa inicialmente, debido a la necesidad de realizar la funcionalidad pendiente del primer *Sprint* y la planificada para estas dos semanas.

7.2.4. Sprint 3 (10/03/2022 - 24/03/2022)

Para este *Sprint* se ha planeado realizar una primera versión del diagrama de secuencia de la historia de usuario 3.1, arreglar los problemas ocasionados por el uso del reproductor de vídeo de *YouTube* en un *Fragment*, crear una sala virtual que albergue la partida del jugador, y la posibilidad de finalizar la ronda actual para poder ver los resultados de la misma. Todas estas tareas se recogen en la tabla 7.3.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU3.1	▪ Diseño	-	5 Horas 50 Minutos	Finalizada primera versión.

-	▪ Vídeo YouTube	-	2 Hora 40 Minutos	Finalizado
Épica 1	▪ Crear Sala	1 Punto (5 Horas)	5 Horas	Finalizado
Épica 1	▪ Iniciar partida	1 Punto (5 Horas)	6 Horas	Finalizado
HU3.4	▪ Enviar respuestas ▪ Obtener puntuación ▪ Crear gráficas	5 Puntos (25 Horas)	21 Horas	Finalizado
Resumen			**40 Horas 30 Minutos**	**5/5 Finalizadas**

Tabla 7.3: Tareas realizadas en el Sprint 3

La creación de la sala no tuvo mucha dificultad más allá de realizar una petición a la base de datos remota y guardar el resultado en la base de datos local. El único aspecto a destacar es la manera de añadir una cabecera a la petición de manera dinámica [30] debido a que se necesita el token obtenido de la autenticación del moderador.

Sin embargo empezar la partida fue más difícil de lo esperado a pesar de no tener aparentemente inconvenientes debido a su similitud con otras tareas relacionadas con peticiones a la base de datos. Al realizar la consulta, se obtenía un error de que el *JSON* estaba mal formado, que fue solucionado añadiendo una factoría con el objeto *Gson*. Posteriormente, se recibía como resultado de la petición que el *JSON* no era completamente consumido. Este error se obtenía por obtener una cadena de texto de la base de datos en vez de un objeto del paquete *Entity* [54]. Para no añadir complejidad y dependencias a las peticiones, se cambió el objeto a recibir de la consulta, siendo este de tipo *Unit*, un tipo genérico, ya que la *API* no devuelve nada relevante en esta consulta, solo se necesita el código de la respuesta.

Para resolver el problema originado por la múltiple creación de *YouTubeFragment* cada vez que se iniciaba el *Fragment* que lo contenía, se optó por destruirlo al ir a otra pantalla, ya que parece que no se hace automáticamente.

La historia de usuario 3.4 se ha desarrollado sin inconvenientes, aunque con la necesidad de buscar información sobre diferentes librerías para implementar las gráficas y para representar los datos y personalizar la apariencia de este. Se ha elegido la librería *MPAndroidChart* debido a su gran popularidad, lo que influye directamente en su mantenimiento y la documentación disponible, y las posibilidades de personalización de los gráficos.

Por último, en cuando al diseño de la historia de usuario 3.1, surgieron dudas principalmente sobre como desarrollar la secuencia relativa a los *listeners* y sobre la respuesta recibida al realizar una petición a la base de datos remota, debido a la asincronía de estas llamadas. La solución fue realizar diagramas independientes para los *listeners* que se originan con una llamada por parte del usuario cuando pulsa un botón o una de las opciones de la pregunta, y para la respuesta de la API, un diagrama cuyo comienzo se da al recibir la respuesta (o el fallo) por parte del servidor de forma asíncrona.

7.2.5. Sprint 4 (24/03/2022 - 07/04/2022)

Las tareas a realizar en este *Sprint* son las tres últimas historias de usuario de la épica 2, y la corrección de las dudas surgidas en el desarrollo de la primera versión de los diagramas de secuencia de la historia de usuario 3.1. Estas tareas están reflejadas en la tabla 7.4.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU3.1	▪ Diseño	-	3 Horas 40 Minutos	Finalizada segunda versión.
HU2.5	▪ Diseño ▪ Implementación ▪ Pruebas	3 Puntos (15 Horas)	8 Horas 30 Minutos	Finalizado

HU2.6	▪ Diseño ▪ Implementación ▪ Pruebas	1 Punto (5 Horas)	1 Hora 20 Minutos	Finalizado
HU2.7	▪ Diseño ▪ Implementación ▪ Pruebas	2 Punto (10 Horas)	9 Horas 40 Minutos	Finalizado
Documentación	▪ Tecnologías utilizadas	-	8 Horas 20 Minutos	Finalizada primera versión
Backend	▪ Despliegue Backend	-	3 Horas	Finalizado
Resumen			**34 Horas 30 Minutos**	**6/6 Finalizadas**

Tabla 7.4: Tareas realizadas en el Sprint 4

Al comienzo de estas dos semanas se arreglaron fallos de la primera versión de los diagramas de secuencia de la historia de usuario 3.1, entre los que se encuentran las llamadas asíncronas de las respuestas de las peticiones y el comienzo de la secuencia de los *Listeners*.

El desarrollo de la historia de usuario 2.7 fue más difícil de lo previsto debido a que la respuesta recibida de la base de datos no está en *UTF-8*, si no que su codificación es *ISO-8859-1*. Esto complicó mucho las cosas ya que la respuesta no se podía transformar directamente a un *String*, y ni si quiera utilizando el tipo genérico *Unit* se recibía una cadena de caracteres, siendo este el único tipo que no daba error.

Finalmente, se solucionó usando una factoría para convertir la respuesta a texto [2]. De esta manera se pudo devolver un objeto *String* sin ningún tipo de error.

Durante este *Sprint* surgió un gran imprevisto. Una fallida actualización del ordenador, hizo que se quedara congelado durante la misma, sin posibilidad de cancelarla ni completarla.

En el modo recuperación había una opción para instalar una nueva copia del sistema operativo sin perder los datos, pero una vez descargada, se volvió a quedar congelado antes de poder recuperar los archivos del ordenador. La única solución que quedó fue formatear el ordenador, perdiendo así todos los progresos no guardados en *Git* (lo relativo a la historia de usuario 2.7 desarrollado hasta este momento), y con la necesidad de volver a desplegar el *backend*, lo que hizo que se retrasara el desarrollo del *Sprint*. Dicho imprevisto se incluyó en el plan de riesgos, ya que se materializó y no se había contemplado (**Riesgo R12** Tabla 2.22).

Las historias de usuario 2.5 y 2.6 se desarrollaron sin inconvenientes, y se aprovechó el tiempo restante para avanzar con la memoria del Trabajo de Fin de Grado. En concreto, se desarrolló el capítulo de tecnologías utilizadas, y se mejoró la tabla de los resultados obtenidos en los casos de prueba 6.48.

7.2.6. Sprint 5 (07/04/2022 - 21/04/2022)

Para este *Sprint* está planeado terminar la funcionalidad básica de la aplicación. Estas tareas se presentan en la tabla 7.5.

Historia de Usuario	Tareas	Puntos de historia estimados	Tiempo empleado	Detalles
HU3.5	▪ Modificación	-	5 Horas	Finalizado
HU3.6	▪ Diseño ▪ Implementación ▪ Pruebas	1 Puntos (5 Horas)	4 Horas 20 Minutos	Finalizado
HU3.7	▪ Implementación ▪ Pruebas	1 Puntos (5 Horas)	10 Horas 30 Minutos	Finalizado
HU3.8	▪ Implementación ▪ Pruebas	1 Puntos (5 Horas)	1 Hora	Finalizado

HU3.9	▪ Implementación ▪ Pruebas	1 Puntos (5 Horas)	4 Horas 25 Minutos	Finalizado
-	▪ Toolbar	-	1 Hora	Finalizado
-	▪ Solución fallo al mostrar ayuda	-	1 Hora 45 Minutos	Finalizado
Documentación	▪ Análisis	-	-	No comenzado
Resumen			**28 Horas**	**7/8 Finalizadas**

Tabla 7.5: Tareas realizadas en el Sprint 5

Para la realización de la Historia de Usuario 3.9 surgieron inconvenientes al tener que recuperar en el *ViewModel* actual datos guardados en otros *ViewModels*. El repositorio encargado de recuperar los datos de la base de datos local *Room* devolvía un objeto nulo. Finalmente se vió que ese problema surgía al querer obtener y usar el objeto desde el *ViewModel*, pero al observarlo desde el *Fragment* no se reproducía ese problema, por lo que se optó por observar los objetos *Room* y *AuthResponse*, y llamar al método del *ViewModel* para finalizar la ronda desde la vista.

Se necesitó modificar el desarrollo relacionado con la HU3.5 para mostrar las respuestas de las siguientes rondas, debido a la necesidad de guardar las puntuaciones de todas las rondas, y por lo tanto obtener una lista de puntuaciones en lugar de un único objeto *Score*.

Debido a esta modificación, y a la implementación de la *Toolbar* en todos los *Fragments* y la solución de un fallo al mostrar las pistas adecuada según la pregunta, no hubo tiempo suficiente para realizar la parte de análisis de la documentación.

7.2.7. Sprint 6 (21/04/2022 - 05/05/2022)

En este *Sprint* la prioridad será avanzar con la documentación, en concreto con la sección de análisis y la revisión del capítulo de tecnologías utilizadas. También está previsto realizar la historia de usuario 4.2, con lo que se concluirá la funcionalidad básica prevista de la aplicación.

Historia de Usuario	Tareas	Puntos de historia estimados	Tiempo empleado	Detalles
HU4.2	▪ Diseño ▪ Implementación ▪ Pruebas	1 Puntos (5 Horas)	4 Horas	Finalizado
Épica 1	▪ Finalizar partida	-	1 Hora 10 Minutos	Finalizado
-	▪ Desactivar botón retroceder	-	40 Minutos	Finalizado
-	▪ Animaciones entre fragmentos	-	1 Hora 15 Minutos	Finalizado
-	▪ Mover lógica a ViewModels	-	2 Horas 50 Minutos	Finalizado
-	▪ Resaltar mejor ronda	-	1 Horas 50 Minutos	Finalizado

-	▪ Mover peticiones a APIs a repositorios	-	2 Horas 35 Minutos	Finalizado
-	▪ Corrutinas	-	1 Horas 55 Minutos	Finalizado
Documentación	▪ Revisión Tecnologías utilizadas	-	1 Hora 45 Minutos	Finalizado
Documentación	▪ Análisis	-	5 Horas 35 Minutos	Finalizado
Documentación	▪ Diseño	-	13 Horas 30 Minutos	No Finalizado
Resumen			**37 Horas 05 Minutos**	**10/11 Finalizadas**

Tabla 7.6: Tareas realizadas en el Sprint 6

El desarrollo del *Sprint* transcurrió mejor de lo previsto. Al ya tener realizado el modelo de dominio y la máquina de estados, el capítulo de Análisis se completó con bastante rapidez.

El tiempo restante se empleó en avanzar el capítulo de Diseño. También se aprovechó para refactorizar el código y revisar si se podía mover algo de funcionalidad de las vistas a los *ViewModels*. Esto conllevó que fuera necesaria una tercera versión del diagrama de secuencia.

También se corrigió un error de estructura conocido. Se movieron las peticiones a APIs del *ViewModel* a los repositorios. Durante esta tarea, se descubrió que para que un *ViewModel* pueda utilizar un objeto guardado en el repositorio sin tener que pasarlo mediante parámetro de una operación desde el fragmento, este objeto debe estar siendo observado. De lo contrario su valor no será actualizado.

Por último se han optimizado los tiempos de carga al requerir o enviar datos a la base

de datos, mediante el uso de **corrutinas**, que resumidamente, sirven para liberar al hilo principal de tareas pesadas, ya que este hilo es el encargado de mostrar la interfaz. De esta manera, se han pasado todas las peticiones a APIs a hilos secundarios.

7.2.8. Sprint 7 (12/05/2022 - 26/05/2022)

Este *Sprint* se retrasó una semana debido a que el alumno no estuvo disponible la semana anterior por razones personales. El objetivo principal de este *Sprint* es finalizar el capítulo de diseño, adaptar la aplicación para que sea fácil realizar un cambio del orden de las preguntas, preparar la aplicación para realizar pruebas de sistema y crear un formulario para poder hacer pruebas de usabilidad.

En la tabla 7.7 se muestra el seguimiento de las tareas desarrolladas en este *Sprint*. No se muestra la tarea de preparar la aplicación para realizar pruebas de sistema, ya que simplemente consistía en cambiar la URL de la API para utilizar la del servidor remoto en lugar del local.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU3.1	▪ Diseño	-	10 Horas 35 Minutos	Finalizada tercera versión.
Épica 7	▪ Orden preguntas	-	6 Horas 10 Minutos	Finalizado
Épica 5	▪ Ajuste a diferentes tamaños de pantalla	-	2 Horas 45 Minutos	Finalizado
-	▪ Gestión de errores	-	4 Horas 10 Minutos	Finalizado

Pruebas	▪ Cuestionario pruebas de usabilidad	-	2 Horas 10 Minutos	Finalizado
Documentación	▪ Diseño	-	15 Horas 15 Minutos	Finalizado
Documentación	▪ Introducción	-	5 Horas 30 Minutos	Finalizado
Documentación	▪ Conclusión	-	2 Horas	Finalizado
Resumen			**48 Horas 35 Minutos**	**8/8 Finalizadas**

Tabla 7.7: Tareas realizadas en el Sprint 7

Al adaptar la aplicación para admitir cambios de orden el las preguntas mediante un *array* que se recibe de la base de datos junto a las cuestiones, fue necesario modificar el diagrama de secuencia de la Historia de Usuario 3.1. Ahora en lugar de recibir un *array* de preguntas, se recibe un objeto con un vector de preguntas y otro con el orden en el que se deben mostrar.

También se ha finalizado en este *Sprint* toda la documentación relativa al capítulo de diseño. Por último, se ha preparado el cuestionario para realizar las pruebas de usabilidad.

Al haber finalizado las prácticas de empresa, he podido aprovechar el tiempo adicional para realizar los capítulos de introducción y conclusión, la épica 5 para adaptar todas las vistas a cualquier tamaño de dispositivo y la gestión de errores para las consultas a la API.

7.2.9. Sprint 8 (26/05/2022 - 09/06/2022)

Este *Sprint* se utilizó para realizar los retoques necesarios para finalizar la documentación y la aplicación.

Principalmente se completó el capítulo de implementación y pruebas con las pruebas de usabilidad, se hizo el resumen, se realizó la épica 6, es decir, la documentación de la aplicación mediante los manuales de despliegue e instalación, mantenimiento y usuario, y se corrigió algún detalle de la interfaz de la aplicación.

Por último se incluyó una nueva historia de usuario para visualizar las respuestas dadas en cada ronda desde la vista de resultados finales.

En la tabla 7.8 se pueden ver todas las tareas realizadas.

Historia de Usuario	**Tareas**	**Puntos de historia estimados**	**Tiempo empleado**	**Detalles**
HU4.3	▪ Implementación ▪ Pruebas	1 Punto (5 Horas)	7 Horas 5 Minutos	Finalizado.
Épica 6	▪ Manuales	-	5 Horas 20 Minutos	Finalizado.
Documentación	▪ Resumen ▪ Abstract	-	2 Horas 30 Minutos	Finalizado.
Documentación	▪ Implementación y Pruebas	-	6 Horas 10 Minutos	Finalizado.
-	▪ Mejoras interfaz	-	3 Horas	Finalizado.
Documentación	▪ Corrección memoria	-	13 Horas 15 Minutos	Finalizado.

Resumen	**37 Horas 20 Minutos**	**6/6 Finalizadas**

Tabla 7.8: Tareas realizadas en el Sprint 8

Una vez desarrollada la historia de usuario pendiente y todos los capítulos de la documentación, se realizaron las correcciones oportunas para finalizar el Trabajo de Fin de Grado.

7.3. Resumen del proyecto

7.3.1. Calendarización final

Finalmente no hizo falta hacer uso de los dos *Sprints* extras, y se finalizó una semana más tarde de lo previsto, debido a que entre el 5 y el 12 de mayo el alumno no estuvo disponible.

En la tabla 7.9 se muestra la calendarización final:

Calendario de Sprints		
Sprint	**Fecha inicio**	**Fecha fin**
Sprint 0	02/02/2022	10/02/2022
Sprint 1	10/02/2022	24/02/2022
Sprint 2	24/02/2022	10/03/2022
Sprint 3	10/03/2022	24/03/2022
Sprint 4	24/03/2022	07/04/2022
Sprint 5	07/04/2022	21/04/2022
Sprint 6	21/04/2022	05/05/2022
Sprint 7	12/05/2022	26/05/2022
Sprint 8	26/05/2022	09/06/2022

Tabla 7.9: Planificación final de *Sprints*

7.3.2. Tiempo empleado final

El tiempo estimado para la realización de este proyecto fue **313 horas** incluyendo el *Sprint* 0, tal y como se explica en la Sección 2.5. Finalmente, sumando el tiempo empleado en cada *Sprint*, se han empleado **339 horas y 20 Minutos**, lo que supera las 300 horas para las que estaba prevista la asignatura del Trabajo de Fin de Grado.

7.3.3. Costes finales

La duración del proyecto ha sido de 4 meses exactos, descontando la semana entre el *Sprint* 6 y 7 que no se trabajó y como se comentó en la anterior subsección, se han consumido 339 horas y 20 Minutos.

Con estos nuevos valores, se calcularán los costes simulado y real para compararlos con los presupuestos estimados en la Sección 2.7.

Coste simulado

En la tabla 7.10 se muestran los mismos elementos que en la tabla del presupuesto simulado, pero con los datos de meses y horas empleados finales. Al haber empleado los mismos meses, el presupuesto simulado no difiere mucho del coste simulado.

Coste simulado del proyecto			
Elemento	**Coste**	**Cantidad**	**Coste total**
Desarrollador Android	17,47€/hora	339 horas	5922,33€
Scrum Master	21,69€/hora	64 horas	1.388,17€
Product Owner	24,03€/hora	64 horas	1.538,13€
Ordenador	13,54€/mes	4 meses	54,16€
Smartphone	8,29€/mes	4 meses	33,16€
Ratón	0,75€/mes	4 meses	3€
Licencia Astah	13,5€/mes	4 meses	54€
GitLab	17,70€/mes	4 meses	70,80€
Overleaf	5€/mes	4 meses	20€
Teams	5,10€/mes	4 meses	20,40€
Subtotal			9104,15€
Costes indirectos	15 % del proyecto		1365,62€
Total			**10469,77€**

Tabla 7.10: Coste simulado del proyecto

Coste real

El coste real se mantiene igual que el presupuestado, como se puede ver en la Tabla 7.11.

Coste real del proyecto			
Elemento	**Coste/mes**	**Cantidad**	**Coste total**
Ordenador	13,54€/mes	4 meses	54,16€
Smartphone	8,29€/mes	4 meses	33,16€
Ratón	0,75€/mes	4 meses	3€
Licencia Google	-	-	24,39€
Total			**114,71€**

Tabla 7.11: Coste real del proyecto

Debido al recibo de la beca los dos últimos meses del desarrollo de este proyecto, finalmente ha habido un beneficio de **108,31€** en vez de costes.

Capítulo 8

Conclusiones

La memoria describe el desarrollo de un proyecto software que cumple con los objetivos propuestos (Sección 1.4), pasando por todas las fases (planificación, análisis, diseño, implementación y pruebas) y se han desarrollado todas las historias de usuario mostradas en el *Product Backlog* final (Sección 2.4).

Tal y como se planteó al inicio, se ha desarrollado una aplicación para concienciar sobre el cambio climático y el desarrollo sostenible. Se ha seguido una metodología ágil como es *Scrum*, la cual ha resultado ser de gran ayuda debido a sus reuniones periódicas y a establecer objetivos con la intención de cumplirlos cada dos semanas.

En cuanto a los riesgos detallados en la Sección 2.6, se manifestaron los siguientes:

- Riesgo 1 2.11: Algunas tareas no estuvieron finalizadas en el *Sprint* en el que fueron planeadas, por lo que hubo que reprogramarlas para *Sprints* posteriores.
- Riesgo 6 2.16: Como se comentó en la Sección 6.1, el orden de las preguntas fue modificado, y con ello la manera de mostrar las preguntas y obtenerlas de la base de datos. Finalmente no fue necesario sacrificar ninguna funcionalidad ni utilizar los *Sprints* extras.
- Riesgo 12 2.22: Como se ha comentado en el seguimiento del *Sprint 4* 7.2.5 fue necesario formatear el ordenador e instalar todo el software necesario.

Finalmente no ha habido ningún inconveniente por la posible falta de conocimiento del lenguaje de programación *Kotlin*, aunque sí ha sido necesario aplicar el plan de mitigación establecido en el riesgo 4 2.14 para el uso de tecnologías como los *Fragments* siguiendo la arquitectura *Single-Activity*, *Retrofit* y *Room*.

Se han realizado pruebas de integración, sistema y usabilidad, y el software final ha superado con éxito todas, como se documenta en el capítulo 6. Algunas de ellas se han superado con éxito desde el inicio, y otras han servidor para depurar fallos en la aplicación.

Como valoración personal, el balance del trabajo es altamente satisfactorio. Primero porque se cumple uno de los objetivos principales que fue propuesto antes de empezar el proyecto como era desarrollar una aplicación para *Android* y que esta fuera útil y sirviera para algo más que para aprobar esta asignatura. Segundo porque esta aplicación será usada en institutos debido a que tiene un proyecto de varios años detrás (LOCOMOTION) y seguirá siendo mantenida y desarrollada en el futuro. Y tercero, porque la realización de este proyecto ha supuesto la aplicación de los conocimientos adquiridos a lo largo de la carrera en un problema práctico de interés práctico evidente.

8.1. Líneas de trabajo futuras

A continuación, se mencionan algunas características que podría incluir la aplicación en el futuro:

- Permitir al usuario elegir el número de rondas que desea que dure la partida.
- Dar la posibilidad al usuario de acabar la partida al final de cada ronda.
- Incluir unos botones en la parte inferior del formulario con los números de las preguntas para avanzar directamente a cada una de ellas y marcar de alguna manera los botones para saber que cuestiones faltan por contestar.
- Opción para guardar los resultados conseguidos en cada partida y crear un ranking (Requeriría trabajo adicional en la base de datos).

Algunas de estas nuevas características han sido solicitadas por los directores del proyecto a pocos días de finalizar el Trabajo de Fin de Grado, por lo que no era posible incluirlas en el mismo. Debido a la beca otorgada por el Consejo Social, el alumno continuará con su desarrollo de la aplicación durante el mes de julio.

Bibliografía

[1] A. Nieto. Scrum: Las reuniones. https://blog.bi-geek.com/scrum-las-reuniones/. Accessed: 2022-02-06.

[2] Adrian K. Read plain text response from server using retrofit. https://stackoverflow.com/questions/62901224/read-plain-text-response-from-server-using-retrofit. Accessed: 2022-03-28.

[3] Advanced REST Client. Meet advanced rest client. https://install.advancedrestclient.com/install. Accessed: 2022-04-04.

[4] alonsojpd. Restablecer resetear contraseña usuario root de mysql 8 en linux centos 7. https://proyectoa.com/restablecer-resetear-contrasena-usuario-root-de-mysql-8-en-linux-centos-7/. Accessed: 2022-05-27.

[5] Android Developers. 3. los fragmentos y su ciclo de vida. https://developer.android.com/codelabs/basic-android-kotlin-training-fragments-navigation-component#2. Accessed: 2022-04-02.

[6] Android Developers. Android basics in kotlin. https://developer.android.com/courses/android-basics-kotlin/course. Accessed: 2022-05-03.

[7] Android Developers. android studio. https://developer.android.com/studio. Accessed: 2022-03-26.

[8] Android Developers. Corrutinas de kotlin en android. https://developer.android.com/kotlin/coroutines. Accessed: 2022-05-03.

[9] Android Developers. Cómo guardar contenido en una base de datos local con room. https://developer.android.com/training/data-storage/room?hl=es-419. Accessed: 2022-02-28.

[10] Android Developers. Cómo pasar datos entre destinos. https://developer.android.com/guide/navigation/navigation-pass-data#Safe-args. Accessed: 2022-04-02.

[11] Android Developers. Descripción general de livedata. https://developer.android.com/topic/libraries/architecture/livedata. Accessed: 2022-04-27.

[12] Android Developers. Develop android apps with kotlin. https://developer.android.com/kotlin. Accessed: 2022-03-26.

[13] Android Developers. Dispositivos compatibles con cada versión de android. Accessed: 2022-06-02.

[14] Android Developers. Fragmentos. `https://developer.android.com/guide/components/fragments`. Accessed: 2022-04-02.

[15] Android Developers. Navigation. `https://developer.android.com/guide/navigation`. Accessed: 2022-04-02.

[16] Android Developers. Que es android. `https://www.android.com/intl/es_es/what-is-android/`. Accessed: 2022-03-13.

[17] Android Developers. Youtubebaseactivity. `https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubeBaseActivity?hl=es`. Accessed: 2022-02-15.

[18] Antevenio. Scrum vs agile: ¿en qué se diferencian estas metodologías? `https://www.antevenio.com/blog/2021/02/scrum-vs-agile/`. Accessed: 2022-02-08.

[19] Apache. Apache tomcat. `https://tomcat.apache.org/`. Accessed: 2022-04-04.

[20] AristiDevs. Mvvm en android con kotlin, livedata y view binding – android architecture components. `https://cursokotlin.com/mvvm-en-android-con-kotlin-livedata-y-view-binding-android-architecture-components/`. Accessed: 2022-04-26.

[21] Astah. Full-featured modeling tool available. `https://astah.net/products/astah-professional/`. Accessed: 2022-04-04.

[22] ayushpandey3july. How to use singleton pattern for room database in android? `https://www.geeksforgeeks.org/how-to-use-singleton-pattern-for-room-database-in-android/`. Accessed: 2022-04-28.

[23] B. Hughes y M. Cotterell. Software project management, 5ª edición. PMBOK Fifth Edition. Accessed: 2022-02-03.

[24] B. Kitpitak. Reasons to use android single-activity architecture with navigation component. `https://oozou.com/blog/reasons-to-use-android-single-activity-architecture-with-navigation-component-36`. Accessed: 2022-05-03.

[25] Boehm. Boehm's top ten risk items: Type of risks. `https://people.cs.pitt.edu/~chang/153/c03manage/risk4.htm`. Accessed: 2022-02-04.

[26] C. Álvarez. Usando el patron factory. `https://www.arquitecturajava.com/usando-el-patron-factory/`. Accessed: 2022-04-29.

[27] Crossroads. Crossroads game api. `http://157.88.62.117:443/swagger-ui.html`. Accessed: 2022-05-17.

[28] D. Alvarez Antelo, I. Capellán-Pérez, L. 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. Accessed: 2022-05-17.

[29] E. Largo. Patrones de diseño en java: Mvc, dao y dto. https://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/. Accessed: 2022-05-15.

[30] Flip Android. Retrofit .ªutorizaciÓn", "portador- token. https://www.flipandroid.com/retrofit-autorizacin-portador-token.html. Accessed: 2022-03-14.

[31] Git. git –distributed-is-the-new-centralized. https://git-scm.com/. Accessed: 2022-03-27.

[32] GitLab. The devops platform has arrived. https://about.gitlab.com/. Accessed: 2022-03-26.

[33] GitLab. Issue boards. https://docs.gitlab.com/ee/user/project/issue_board.html. Accessed: 2022-03-27.

[34] GitLab. What is devops? https://about.gitlab.com/topics/devops/. Accessed: 2022-03-26.

[35] Glassdoor. Salario medio para product owner en españa, 2022. https://es.talent.com/salary?job=product+owner. Accessed: 2022-02-21.

[36] Glassdoor. Sueldos para project manager en reino de españa. https://www.glassdoor.es/Sueldos/espa~na-project-manager-sueldo-SRCH_IL.0,6_IN219_KO7,22.htm?clickSource=searchBtn. Accessed: 2022-02-05.

[37] Glassdoor. ¿cuánto gana un desarrollador android? https://www.glassdoor.es/Sueldos/desarrollador-android-sueldo-SRCH_KO0,21.htm. Accessed: 2022-02-05.

[38] Glassdoor. ¿cuánto gana un product owner? https://www.glassdoor.es/Sueldos/espa~na-product-owner-sueldo-SRCH_IL.0,6_IN219_KO7,20.htm. Accessed: 2022-02-21.

[39] Google. Cómo utilizar play console. https://support.google.com/googleplay/android-developer/answer/6112435?hl=es#zippy=%2Cpaso-paga-la-cuota-de-registro. Accessed: 2022-02-21.

[40] I. de Blas, I. Capellán-Pérez, J. Nieto, C. de Castro, L. J. Miguel, O. Carpintero, M. Mediavilla, L. F. Lobejón, N. Ferreras-Alonso, P. Rodrigo, F. Frechoso, D. Alvarez Antelo. Medeas: a new modeling framework integrating global biophysical and socioeconomic constraints. Energy Environ. Sci, 13:986–1017, 2020. Accessed: 2022-05-17.

[41] J. Sauro. Measuring usability with the system usability scale (sus). https://measuringu.com/sus/. Accessed: 2022-06-04.

[42] J. Sutherland y K. Schwaber. The 2020 scrum guide. https://scrumguides.org/scrum-guide.html. Accessed: 2022-02-07.

[43] K. Beck, M. Beedle, A. van Bennekum y 14 autores más. Manifesto for agile software development. https://agilemanifesto.org/. Accessed: 2022-02-07.

[44] L. M. González Calderón. Diseño de la interacción y desarrollo del back-end de crossroads 2.0, un juego educativo para concienciar sobre el cambio climático. https://uvadoc.uva.es/bitstream/handle/10324/50036/TFG-G5227.pdf?sequence=1&isAllowed=y. Accessed: 2022-05-17.

[45] LOCOMOTION H2020. Low-carbon society: an enhanced modelling tool for the transition to sustainability. https://www.locomotion-h2020.eu/about-project/overview/. Accessed: 2022-05-17.

[46] M. Alda Peñafiel. Desarrollo del front-end y mejoras en el back-end de un juego didáctico multijugador de competición y consenso sobre el cambio climático. https://uvadoc.uva.es/bitstream/handle/10324/50085/TFG-G5228.pdf?sequence=1&isAllowed=y. Accessed: 2022-05-17.

[47] M. Cillero. Pruebas de integración. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/integracion. Accessed: 2022-06-08.

[48] M. Perminova. What is miro? https://help.miro.com/hc/en-us/articles/360017730533-What-is-Miro-. Accessed: 2022-04-04.

[49] Maven. Welcome to apache maven. https://maven.apache.org/. Accessed: 2022-04-23.

[50] Microsoft. Reinventar la productividad con microsoft 365 y microsoft teams. https://www.microsoft.com/es-es/microsoft-365/business/compare-all-microsoft-365-business-products-b. Accessed: 2022-02-21.

[51] Microsoft. Windows 10 original, ventajas reales. https://www.microsoft.com/es-es/d/windows-10-home/d76qx4bznwk4/1nt3?rtc=1&activetab=pivot:informaci%C3%B3ngeneral%C2%A0tab. Accessed: 2022-02-21.

[52] Miro. Precios. https://miro.com/es/pricing/. Accessed: 2022-02-21.

[53] MongoDB. ¿qué es mongodb? https://www.mongodb.com/es/what-is-mongodb. Accessed: 2022-04-03.

[54] MortezaNedaei. json document was not fully consumed. https://github.com/square/retrofit/issues/3004. Accessed: 2022-03-15.

[55] MySQL. What is mysql? https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Accessed: 2022-04-03.

[56] N. Thomas. How to use the system usability scale (sus) to evaluate the usability of your website. https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/. Accessed: 2022-06-04.

[57] Oracle. Printable version ¿qué es la tecnología java y para qué la necesito? https://www.java.com/es/download/help/whatis_java.html. Accessed: 2022-04-04.

[58] Overleaf. About us. https://es.overleaf.com/about. Accessed: 2022-04-04.

[59] Overleaf. Get instant access to overleaf. https://es.overleaf.com/user/subscription/plans. Accessed: 2022-02-21.

[60] P. Ramírez. ¿cuáles son los sistemas operativos más usados o utilizados en 2021? https://itsoftware.com.co/content/sistemas-operativos-mas-usados/. Accessed: 2022-03-13.

[61] Qualtrics. Escala likert: ¿qué es? pros y contras de la escala de evaluación. https://www.qualtrics.com/es/gestion-de-la-experciencia/investigacion/escala-de-likert/. Accessed: 2022-06-04.

[62] Redacción BBC Mundo. Por qué 2030 es la fecha límite de la humanidad para evitar una catástrofe global. https://www.bbc.com/mundo/noticias-45785972. Accessed: 2022-05-17.

[63] refactoring guru. Observer. https://refactoring.guru/es/design-patterns/observer. Accessed: 2022-04-29.

[64] refactoring guru. Singleton. https://refactoring.guru/es/design-patterns/singleton. Accessed: 2022-04-28.

[65] Scrum Manager. Epic. https://www.scrummanager.net/bok/index.php?title=Epic. Accessed: 2022-02-09.

[66] Square. Retrofit. https://square.github.io/retrofit/. Accessed: 2022-04-03.

[67] Talent. Salario medio para desarrollador android en españa, 2022. https://es.talent.com/salary?job=desarrollador+android. Accessed: 2022-02-05.

[68] Talent. Salario medio para project manager en españa, 2022. https://es.talent.com/salary?job=project+manager. Accessed: 2022-02-05.

[69] Team Asana. 7 riesgos comunes de un proyecto y cómo prevenirlos. https://asana.com/es/resources/project-risks. Accessed: 2022-02-04.

[70] Telegram. Telegram. https://telegram.org/. Accessed: 2022-04-23.

[71] 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. Accessed: 2022-02-07.

[72] U.S. General Services Administration. System usability scale (sus). https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Accessed: 2022-05-30.

[73] Ventana System. Vensim official webpage. https://vensim.com/. Accessed: 2022-05-18.

[74] Visual Paradigm. The #1 development tool suite. https://www.visual-paradigm.com/. Accessed: 2022-04-23.

[75] Vysor. Vysor. https://www.vysor.io/. Accessed: 2022-04-04.

[76] Webex. Webex. https://www.webex.com/es/index.html. Accessed: 2022-04-23.

[77] Wikipedia. Adaptador (patrón de diseño). https://en.wikipedia.org/wiki/Adapter_pattern. Accessed: 2022-04-28.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

Para instalar la aplicación se requiere un dispositivo móvil o *tablet*, con el sistema operativo **Android** y como versión mínima **Android 9.0 Pie**.

Para descargar la aplicación simplemente hay que acceder a la tienda de aplicaciones de *Google*, **Play Store** y buscar "crossroad2" (Enlace disponible en B).

A.2. Manual de mantenimiento

Para continuar con el desarrollo de este proyecto, es necesario descargar **Android Studio** como entorno de programación, y clonar el repositorio mostrado en el Apéndice B. También se deberá crear en el archivo *local.properties* una variable con la clave de la API de *Google* para poder reproducir los vídeos de *YouTube*.

Para desplegar el *backend*, es necesario seguir una serie de pasos explicados con más detalle en el Apéndice A.1.1. del TFG de Manuel Alda [46]. A continuación se muestra una lista muy sintetizada con los pasos necesarios en forma de instrucciones y comandos para realizar el despliegue:

- Instalar HomeBrew
- Instalar Java 11
- Instalar MySQL: brew install mysql
- Instalar MongoDB: brew tap mongodb/brew

- brew install mongodb-community
- Instalar MySQL Workbench: brew install –cask mysqlworkbench
- Instalar Tomcat
- Instalar Maven
- Clonar backend: git clone https://gitlab.inf.uva.es/manalda/crossroads-backend
- Si durante la instalación de MySQL no se ha solicitado una contraseña, consultar este enlace [4]
- Crear BD en MySQL y MongoDB "crossroads"
- Ejecutar mvn spring-boot:run
- Instalar XCode: xcode-select −−install
- Instalar Python: brew install python@3.9
- Instalar requests: pip install requests
- Ir a carpeta *deploy*
- Descargar ficheros de *Google Drive* y guardar en *deploy* (Enlace disponible en el Apéndice B)
- Con *backend* y bases de datos lanzados: python upload_patterns_and_graphics.py Posteriormente seleccionar la opción 0 y 1. La primera tarda bastante, pero es normal. Da fallo de verificación del email pero no pasa nada
- Poblar la base de datos MySQL ejecutando reset_db.sh

A.3. Manual de usuario

Al iniciar la aplicación, se accede a la pantalla principal (Figura A.1), mediante la cual se puede visualizar el vídeo de presentación de Crossroad2, comenzar el juego mediante el botón "Comenzar" (Figura A.2) o ver más información sobre el proyecto pulsando en el botón "Sobre el proyecto" (Figura A.3).

Una vez iniciado el juego, el usuario debe ir contestando a todas las preguntas, seleccionando una de las opciones disponibles para cada cuestión. En todas ellas, el jugador puede abrir la vista de ayuda (Figura A.4), pulsando en el icono o texto de Ayuda situado en la parte inferior de la pantalla. En esta vista, se puede ver un vídeo, leer una pista o pulsar en en botón inferior para obtener más ayuda en Internet.

Una vez respondidas todas las preguntas, se muestran los resultados de la ronda (Figura A.5). En ellos aparecen dos gráficas, representando las simulaciones de temperatura y PIB per cápita anual. Debajo, se ve la puntuación obtenidas en las variables de Precisión, Ecológico

y Equilibrio. Si se pulsa en el icono o texto de "Puntuación" se navega a una vista donde se muestra la información sobre como se calculan las puntuaciones de cada variable (Figura A.6). Desde la vista de resultados podemos avanzar a la siguiente ronda mediante el botón "Intentar de nuevo" (Figura A.8) o ver las recomendaciones (Figura A.7) al pulsar en botón "Recomendaciones".

En la Figura A.8 se muestran las respuestas dadas en la anterior ronda, y se pueden modificar las preguntas deseadas para tratar de mejorar los resultados. Las dos primeras preguntas se pueden visualizar pero no se pueden modificar, ya que son hipótesis y solo se pueden responder en la primera ronda. Mediante el botón "comprobar" se avanza de nuevo a la vista de resultados de ronda.

Una vez finalizadas las tres rondas, aparece un nuevo botón: "Terminar partida". Al pulsarlo se muestran los resultados finales (Figura A.9), compuestos por las puntuaciones obtenidas en las tres rondas, y aparecen resaltados en verde los resultados de la mejor ronda. Si se pulsa los resultados de cualquiera de las tres rondas, se muestran las respuestas seleccionadas en dicha ronda. En la parte inferior hay un botón para regresar a la pantalla inicial del juego.

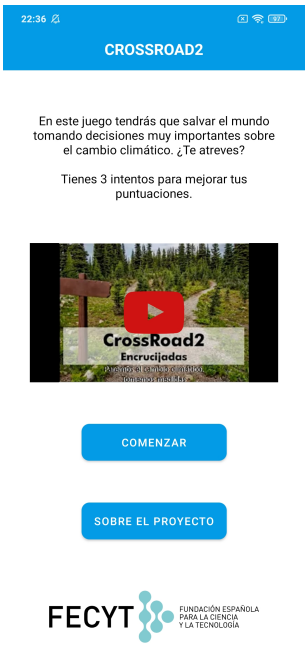

Figura A.1: Pantalla inicial de Crossroads2

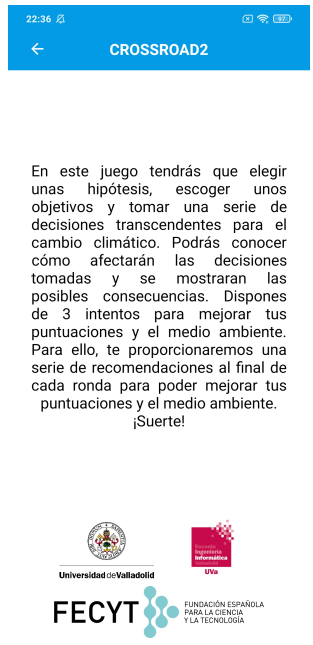

Figura A.2: Información sobre el proyecto

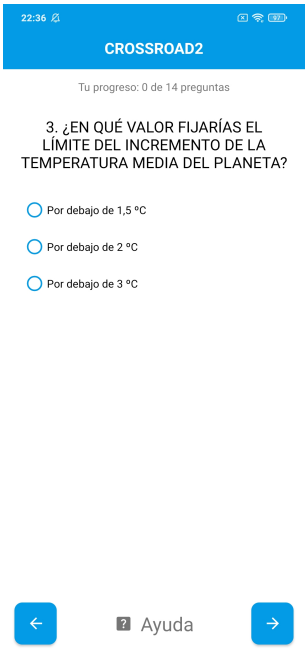

Figura A.3: Pantalla inicial del formulario

Figura A.4: Pantalla de ayuda en cada pregunta

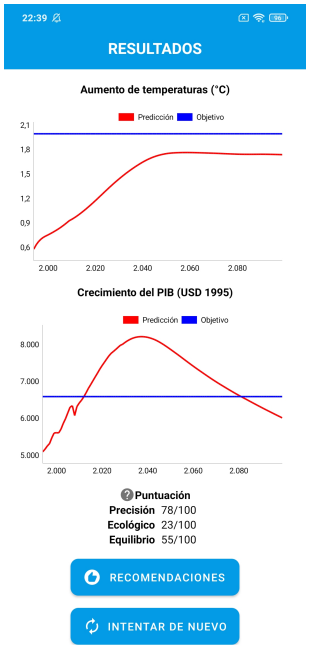

Figura A.5: Resultados de ronda

Figura A.6: Información sobre las puntuaciones

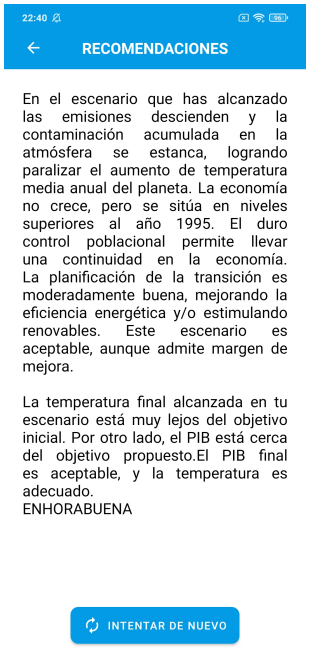

Figura A.7: Pantalla de recomendaciones

Figura A.8: Resumen de las preguntas

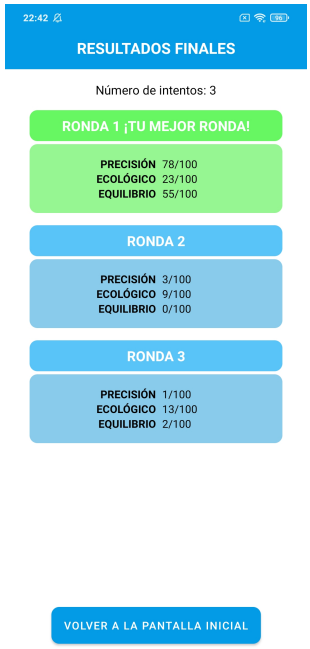

Figura A.9: Resultados finales

Apéndice B

Enlaces adicionales

El enlace útil de interés en este Trabajo Fin de Grado es:

- Repositorio donde se encuentra el código fuente de la aplicación y los archivos de *Visual Paradigm* y *Astah* con los diagramas de análisis y diseño respectivamente.
  `https://gitlab.inf.uva.es/davcres/tfg-davidcrespo`
- Enlace para descargar la aplicación del *Play Store*:
  `https://play.google.com/store/apps/details?id=com.david.crossroad2&gl=ES`
- Ficheros para poblar la base de datos MongoDB para desplegar el *backend* en local:
  `https://drive.google.com/file/d/1xCL3O_hujNYx-HUSZuhWmrC6P1AxyVQ9/view`