Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Mención en Ingeniería del Software

LecturApp

Aplicación multiplataforma para fomentar la lectura en niños

Alumno:
Elías Abril Lorenzo

Tutores:
Alejandra Martínez Monés
Oskar Barrio Ferreiro - HP SCDS
Jesús Lucas Natal - HP SCDS

Dedicado a mi familia

Agradecimientos

Quiero agradecer a mi tutora Alejandra por todos los consejos brindados, tanto generales como de usabilidad, y por resolverme cualquier duda sobre los trámites del Trabajo Fin de Grado.

Me gustaría dar las gracias también a mis tutores de HP SCDS, Oskar Barrio Ferreiro y Jesús Lucas Natal, por el seguimiento durante todo el transcurso de este proyecto.

Por último, quiero agradecer a mi familia, amigos y compañeros el apoyo depositado en mí y la ayuda mostrada siempre que lo he necesitado.

Gracias a todos.

Resumen

El objetivo de este proyecto es desarrollar una aplicación móvil orientada a niños de entre 6 y 12 años, para hacer la lectura más interactiva mediante realidad aumentada. Además, la aplicación permite realizar operaciones de búsqueda y filtrado de libros mediante distintos parámetros, e introduce elementos de gamificación para intentar animar a la lectura de dichos libros.

El proyecto se ha desarrollado como una app utilizando el framework Ionic, el lenguaje de programación C# con tecnología .NET6 y desarrollo Android. Su desarrollo ha seguido las pautas recomendadas por la metodología ágil Scrum.

Abstract

The goal of this project is to develop a mobile application aimed at children between the ages of 6 and 12, to make reading more interactive through augmented reality. In addition, the application allows search and filtering operations of books using different parameters, and introduces gamification elements to try to encourage the reading of these books.

The project has been developed as an app using the Ionic framework, the C# programming language with .NET6 technology and Android development. Its development has followed the guidelines recommended by the agile Scrum methodology.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

La empresa HP SCDS[1] ofrece a los estudiantes de diferentes universidades la oportunidad de realizar sus Trabajos de Fin de Grado dentro de un entorno empresarial mediante el Observatorio Tecnológico HP[2]. Por lo tanto, el estudiante cuenta con la asistencia de un tutor cualificado para guiarle durante el desarrollo del proyecto.

El proceso de selección y asignación de los distintos proyectos a cada estudiante consta de varios pasos entre los cuales se destaca que los candidatos son seleccionados por orden de nota media de la carrera. Más adelante, los tutores profesionales de HP SCDS se ponen en contacto con el estudiante para una reunión inicial donde se establecen los objetivos y la forma de trabajo, además de plantear la metodología con la que se desarrollaría el proyecto y la duración requerida para su consecución. En el caso de este TFG, los tutores por parte de la empresa son Oskar Barrio Ferreiro y Jesús Lucas Natal. Estos profesionales cuentan con experiencia previa en la tutorización de TFGs, además de ser expertos en las tecnologías requeridas para este proyecto y en metodología Scrum.

LecturApp [1] nace de la idea original de los tutores de la empresa para relacionar la lectura con la realidad aumentada. Una vez planteada esta idea, el estudiante es el responsable en decidir qué camino va a tomar el proyecto para lograr el objetivo, contando con la ayuda de los tutores profesionales de HP SCDS citados anteriormente y la profesora universitaria Alejandra Martínez Monés como tutora académica.

Este proyecto está planteado para tener como nicho de mercado a niños en la etapa de educación primaria. Se eligió esta edad porque es aquella en la que se puede suponer que los niños pueden leer de forma más o menos autónoma, y por ser una edad con gran potencial de impacto positivo en el aprendizaje.

[1] HP Solutions Creations & Development Services. Más información: `https://hpscds.com/`
[2] Más información: `https://hpscds.com/observatorio-hp/observatorio-hp-21-22/`

Los tutores de la empresa plantearon una serie de aspectos que debía incluir el proyecto, como una forma de explorar una serie de tecnologías. En concreto, las ideas planteadas incluían lo siguiente:

- Desarrollo de aplicaciones multiplataforma, con uso de Web API y acceso a datos.
- Implementación de sistemas de realidad aumentada en dispositivos móviles.
- Escaneo y reconocimiento de textos y objetos mediante dispositivos móviles.
- Aumento de funcionalidades de la aplicación mediante microservicios.
- Gamificación de aplicaciones con trasfondo educativo o formativo.

1.2. Motivación

En esta última década y con el consecuente avance de la tecnología y del entretenimiento audiovisual con plataformas como Netflix, podría parecer que existe una pérdida en el hábito de lectura, en especial, en edades jóvenes. Pero esto en realidad no es así, y estadísticas realizadas por FGEE (Federación de Gremios de Editores de España) [2] lo demuestran. Además, se hace hincapié en la época del confinamiento provocada por Covid-19, donde las personas han aumentado ese hábito.

La lectura es muy importante, en especial en edades tempranas, porque aumenta la comprensión, atención, observación, concentración, reflexión, pensamiento crítico y memoria de las personas. Además, la lectura es también una de las actividades más importantes en el desarrollo de los niños ya que es la edad perfecta para que les empiece a apasionar la lectura ya que un libro no es sólo un complemento para sus estudios, también es una puerta de entrada a fantásticas aventuras y un compañero inseparable [3]. Sin embargo, estamos inundados de entretenimiento multimedia mucho más fácil de consumir, por lo que podemos aprovechar este interés para fomentar el interés por la lectura en los niños.

Una de estas tecnologías que puede ayudar a fomentar el interés por la lectura es la realidad aumentada, que consiste en una experiencia interactiva de un entorno real en el que los objetos que residen en el mundo real se ven mejorados por información perceptiva generada por ordenador, a veces a través de múltiples modalidades sensoriales, como la visual, la auditiva, la háptica, la somatosensorial y la olfativa [4]. Es un pilar fundamental para que exista una conexión especial entre el estudiante y la aplicación, proporcionándole un punto de partida en este maravilloso mundo que es la lectura.

Otra forma de fomentar hábitos es incluir el juego, a través de gamificación [5], es decir, motivar el aprendizaje a través del juego. Al tratarse de niños de Educación Primaria, sería posible el contacto con colegios para que se incentive la lectura en clase mediante *LecturApp*, realizando metodologías innovadoras mediante esta gamificación.

Por todos estos puntos y con la motivación personal de poder relacionar la lectura con la tecnología plasmándolo en una aplicación móvil, vi en este proyecto la oportunidad perfecta para desarrollar una aplicación mientras marco mi propio aprendizaje en esta materia.

1.3. Objetivos del proyecto

Se plantea el siguiente objetivo: Desarrollar una aplicación multiplataforma que incluya diversas formas de aproximación interactiva a la lectura.

A continuación se listan los aspectos que debe tener la aplicación:

- Se deberá implementar un sistema de búsqueda y filtrado de libros.
- La aplicación deberá conectarse a algún servicio cloud para obtener información del libro. También se podrá "escanear" el libro físico para mostrar un pequeño clip audiovisual con el resumen de éste a modo de "tráiler", utilizando para ello realidad aumentada.
- La aplicación "gamificará" la lectura. Al finalizar cada libro se podrá acceder a un "desafío" tipo *quiz*.

Los tutores de empresa plantearon este proyecto como una exploración de las tecnologías que permitieran cumplir con estos aspectos. Conseguir dichos aspectos supondría finalizar el Producto Mínimo Viable, es decir, obtener el producto o software básico con características suficientes para satisfacer a los clientes (en este caso, los tutores de empresa).

1.4. Nicho de mercado y usuarios objetivo

El principal segmento de mercado al que se dirige la aplicación móvil LecturApp es a niños de 6 a 12 años, los cuales se encuentran en estudios primarios en el colegio. En esta etapa los niños empiezan a disponer de un móvil propio bajo supervisión de las personas adultas, o éstas prestan sus teléfonos móviles a los niños.

Los usuarios objetivo pueden ser los propios niños, junto al familiar correspondiente que pueda monitorizar el uso de la aplicación. Un posible usuario objetivo colectivo podría ser también el colegio para lograr una clase divertida mediante metodología innovadora. La constante evolución de diferentes formas de aprendizaje para lograr el objetivo principal en esta etapa permite a LecturApp mostrar su potencial y entrar en el mercado de un área que no está explotado completamente.

Por cuestiones de alcance del proyecto, en este Trabajo Fin de Grado nos hemos centrado en el primero de los usuarios objetivo. El escenario de uso típico sería, por tanto, un niño que usa la aplicación en un dispositivo móvil, propio o prestado por una persona adulta.

Es necesario consecuentemente que la aplicación sea fácil de utilizar por parte de los niños para provocar una evolución real en su aprendizaje, vital en esta etapa.

1.5. Análisis de competencia

Antes de realizar el proyecto se ha realizado una revisión de qué otras aplicaciones existen en el mercado parecidas a las que se plantea este Trabajo Fin de Grado. Este análisis se ha realizado siguiendo los principios de un análisis de competencia. El análisis de la competencia es el proceso que se pone en práctica para saber cómo actuar en el ambiente competitivo, el cual empieza reconociendo a los competidores de un determinado producto para determinar cuáles son sus principales objetivos, estrategias, puntos débiles y fuertes [6].

Para realizar un correcto análisis de competencia, es necesario establecer qué competidores hay en el mercado y obtener conocimiento y evaluarla para lograr una distinción frente a ellos. Para identificar a los competidores, es necesario investigar qué aplicaciones existen en el mercado. Para ello, se ha realizado una búsqueda en la tienda oficial de Android, la Google Play Store, filtrando por palabras clave tales como "lectura", "niño" o "realidad aumentada".

Con la palabra "lectura" se obtienen resultados como Wattpad [7] o Kindle [8], aplicaciones que no entrarían como competidores directos debido a la categoría de la aplicación y el rango de edad en el que se basa el uso de la aplicación.

Además, si se filtra con la palabra "niño" se empiezan a ver aplicaciones basadas en el aprendizaje básico de lectura para momentos muy tempranos del niño como puede ser edad preescolar. Por lo tanto, estas aplicaciones no reflejan un alto grado de competencia debido al rango de edad.

Otra palabra clave de nuestra aplicación es "realidad aumentada", un módulo claramente diferenciador de nuestra aplicación frente a otras. Los primeros resultados en la tienda son DevAR [9], ARLoopa [10] y Flapp [11], aplicaciones que no corresponden con el mismo tema de lectura como nuestra aplicación.

También existen plataformas de lectura como Ta-tum [12] para gamificar la lectura que sí podría corresponder a un competidor de nuestra aplicación. Las estadísticas principales de Ta-tum es que 7000 docentes ya la han elegido para dinamizar la lectura en sus aulas y 150000 estudiantes de Primaria y Secundaria disfrutan de las lecturas mediante esta tecnología. La principal diferencia respecto con la nuestra es que Ta-tum no posee un módulo de realidad aumentada que lo pueda establecer como competidor directo.

Tras este análisis de competencia podemos concluir este proyecto se presenta como un trabajo con posibilidades en cuanto a su valor de mercado, aunque para poder afirmarlo habría que completar este análisis con una indagación sobre las necesidades de los usuarios objetivo, que no se ha realizado.

1.6. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: Este capítulo describe el contexto y la motivación para el desarrollo de este proyecto, junto a los objetivos principales que se pretenden obtener del proyecto y el nicho de mercado y el análisis de competencia estudiado.

Capítulo 2 Metodología y Planificación: Este capítulo describe la metodología utilizada para el desarrollo del proyecto junto a la planificación necesaria para llevarla a cabo: planificación inicial, plan de riesgos, plan de presupuesto y seguimiento.

Capítulo 3 Tecnologías empleadas: Análisis comparativo y selección: Este capítulo describe las tecnologías utilizadas para el desarrollo de este proyecto. Se enumeran todas las tecnologías y herramientas utilizadas haciendo una breve descripción de cada una de ellas. Se menciona también un estudio de las tecnologías y sus alternativas.

Capítulo 4 Análisis: Este capítulo describe la etapa de análisis dentro del ciclo de vida del software. Esta etapa fija los requisitos para desarrollar el software, es decir, las características que el sistema debe tener.

Capítulo 5 Diseño: Este capítulo describe el diseño del software y los patrones de diseño, es decir, la arquitectura y estructura general y los diversos diagramas para el entendimiento del sistema.

Capítulo 6 Implementación: Este capítulo describe el proceso de implementación que se ha llevado a cabo en el proyecto, tanto a nivel de microservicios, como el módulo de realidad aumentada y la implementación *frontend*.

Capítulo 7 Pruebas: Este capítulo describe las pruebas realizadas para probar el correcto funcionamiento del sistema. Se han realizado pruebas unitarias, pruebas de integración, pruebas de aceptación y pruebas de caja negra.

Capítulo 8 Conclusiones: Este capítulo extrae las conclusiones obtenidas del desarrollo del proyecto, además de analizar el trabajo futuro posible para continuar con el proyecto.

Bibliografía: Este capítulo extrae las referencias bibliográficas utilizadas durante todo el transcurso del proyecto.

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

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

Capítulo 2

Metodología y Planificación

2.1. Metodología Scrum

Para el desarrollo de este proyecto se ha utilizado una metodología ágil[1] basada en Scrum [13]. Esta metodología tiene ciertas características que la convierten en una buena opción para desarrollar un proyecto software.

En el modelo Scrum [14], los proyectos se dividen en pequeñas partes de trabajo que pueden ser desarrolladas y entregadas a lo largo de incrementales en el tiempo que se denominan sprints. Por lo tanto, el producto se desarrolla en una serie de de porciones manejables. Cada sprint suele tener una duración fija. Al final de cada sprint, los interesados y los miembros del equipo se reúnen para evaluar el progreso y los interesados sugieren al equipo de desarrollo los cambios y mejoras que consideren necesarios. Se suele utilizar con un tablero Kanban para definir y gestionar las tareas, además de las historias de usuario.

Scrum es un marco de referencia ágil que se basa en el empirismo, con un enfoque iterativo e incremental. Además, presenta una gran adaptación al cambio. Algunos pilares de Scrum son la transparencia, la inspección y la adaptación.

Los roles que se presentan en una estructura Scrum son los siguientes:

- Scrum Master: gerencia el proceso Scrum y remueve impedimentos, asegurándose que se toman las decisiones para aumentar la productividad.
- Product Owner: gerencia la lista de producto y optimiza el valor del producto. Gestiona el backlog y se encarga que sea transparente, visible y ordenado.
- Equipo de Desarrollo: es autogerenciado y produce incrementos de producto. Es responsable del producto y de la gestión del sprint, por lo que se encarga de convertir el backlog de producto en incrementos de funcionalidad.

[1] Más información: `https://agilemanifesto.org/iso/es/manifesto.html`

Los artefactos utilizados en una estructura Scrum son los siguientes:

- Product Backlog: reúne todas las historias de usuario deseadas para obtener el producto necesario, es decir, el Product Backlog es la lista ordenada de lo que se necesita para mejorar el producto. Las historias del Product Backlog que pueden ser realizados por el equipo de desarrollo dentro de un Sprint se consideran listos para ser seleccionados en el evento de planificación de sprint.
- Sprint Backlog: historias de usuario y tareas que se van a realizar en ese sprint determinado. El Sprint Backlog es un plan para los desarrolladores ya que permite visualizar en tiempo real el trabajo que el equipo planea realizar durante el Sprint para lograr el Objetivo del Sprint.
- Incrementales: funcionalidades liberadas al finalizar el sprint. Un incremento es un paso real hacia el objetivo del producto, además, cada incremento se suma a todos los anteriores y se verifica asegurando que todo funciona. El incremento debe ser utilizable para aportar valor.

Los eventos que surgen durante el trabajo con Scrum son los siguientes:

- Planificación del sprint: reunión inicial que fija contexto, alcance y planificación para el sprint.
- Daily Scrum: reunión corta que se realiza "diariamente"para poner en contexto al equipo sobre los avances del proyecto.
- Revisión: reunión que se realiza al final del sprint. Sirve para evaluar el producto desarrollado obteniendo retroalimentación de los stakeholders.
- Restrospectiva: reunión que se realiza al final del sprint. Suele ser el último evento del sprint y su principal objetivo es mejorar varios factores: la productividad, la metodología de trabajo, la calidad del producto, entre otras.

2.1.1. Scrum individual

Adaptando este modelo a un Scrum individual, los conceptos explicados anteriormente, es decir, los roles y eventos, se fusionan para lograr un Scrum más ajustado a las necesidades del proyecto.

Como el proyecto se realiza de manera individual, los roles de Scrum Master, Product Owner y Desarrollador se fusionan en un solo individuo, el cual es el estudiante. Por lo tanto, el estudiante tiene que realizar las funciones de los distintos roles.

A nivel de eventos, se han fusionado las reuniones de revisión, retrospectiva y planificación del siguiente sprint en una sola reunión, usualmente de una hora. La reunión daily no aplica en nuestro scrum individual por lo que queda descartada.

La Tabla 2.1 muestra los tiempos establecidos para cada una de las partes en las que se divide la reunión.

Reunión Sprint N	**Duración**
Sprint Review N	25 - 30 min
Sprint Retrospective N	10 - 15 min
Planificación Sprint N+1	15 - 20 min

Tabla 2.1: Duración de la reunión N

2.2. Planificación inicial

El Trabajo Fin de Grado es una asignatura de Ingeniería Informática de la Universidad de Valladolid que consta de 12 ECTS según la guía docente de la asignatura [15]. Según el Plan Bolonia existente en la actualidad, 12 ECTS equivalen a 300 horas de trabajo.

La Tabla 2.2 establece la duración de los sprints y las fechas que componen cada sprint. Como se puede ver, existe una duración fija de un mes a excepción del primer sprint (también llamado Sprint 0 en otras ocasiones) debido a que este sprint consiste en la presentación del TFG y fijar ciertos aspectos formales y de alcance de proyecto. Aunque la duración es mayor, la dedicación en cuanto a horas se reduce en gran medida debido a las responsabilidades como estudiante del cuatrimestre anterior, y por lo tanto, el inicio real del proyecto podría considerarse el Sprint 2. El Sprint 7 no tiene fecha de fin porque sería un Sprint especial para finalizar ciertos aspectos de presentación y entrega de proyecto.

Sprints Scrum	**Fecha**
Sprint 1	16/11/2021 - 20/01/2022
Sprint 2	21/01/2022 - 17/02/2022
Sprint 3	18/02/2022 - 17/03/2022
Sprint 4	18/03/2022 - 21/04/2022
Sprint 5	22/04/2022 - 19/05/2022
Sprint 6	20/05/2022 - 16/06/2022
Sprint Extra	17/06/2022 - 24/06/2022

Tabla 2.2: Duración de los sprints

2.2.1. Objetivos y tareas de los Sprints

A continuación se listan distintas tablas para cada Sprint, las cuales contienen la información relevante de cada uno de ellos como el objetivo principal del sprint y las tareas a realizar. Las completitud de las diversas tareas que hay que realizar en cada Sprint daría como resultado haber logrado el objetivo del Sprint. Si el objetivo de un determinado Sprint no se cumple, se detectaría ese Sprint como un fracaso y habría que revalorar las tareas y la planificación.

La Tabla 2.3 muestra la duración del Sprint 1, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 1	**16/11/2021 - 20/01/2022**
Objetivo	Comienzo (*inception*)
Tareas	▪ Decidir las tecnologías y herramientas. ▪ Decidir la manera de gestionar las tareas/implementaciones. ▪ Definir la interfaz de usuario base de la aplicación móvil. ▪ Definir la arquitectura básica para la aplicación móvil ▪ Definir la arquitectura básica para la aplicación backend.

Tabla 2.3: Objetivo y tareas del Sprint 1

La Tabla 2.4 muestra la duración del Sprint 2, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 2	**21/01/2022 - 17/02/2022**
Objetivo	Arquitectura básica de la aplicación y brainstorming de la documentación
Tareas	▪ Implementar la arquitectura básica para la aplicación móvil. ▪ Estudiar y desarrollar el modelo de inicio (inception template). ▪ Gestionar las historias de usuario y definir el modelo de paquetes. ▪ Gestionar las historias de usuario para todos los sprints.

Tabla 2.4: Objetivo y tareas del Sprint 2

La Tabla 2.5 muestra la duración del Sprint 3, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 3	**18/02/2022 - 17/03/2022**
Objetivo	Integrar el framework de RA con la interfaz de usuario de Angular
Tareas	▪ Mostrar la cámara en la pestaña correspondiente. ▪ Generar el objetivo de la imagen (Image Target) de algún libro de ejemplo. ▪ Mostrar un objeto 3D cuando un determinado libro es escaneado.

Tabla 2.5: Objetivo y tareas del Sprint 3

La Tabla 2.6 muestra la duración del Sprint 4, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 4	**18/03/2022 - 21/04/2022**
Objetivo	Implementar la gestión de libros usando la API de Google Books
Tareas	▪ Gestionar las operaciones CRUD para recibir datos (operaciones GET) ▪ Buscar libros con un determinado parámetro (título o autor). ▪ Implementar las actividades (Menú Inicio, Detalles de Libro, Libros Leídos, Pantalla de Quiz) y la navegación con botones. ▪ Desarrollar el microservicio de usuarios para realizar la conexión entre un usuario determinado y los libros leídos.

Tabla 2.6: Objetivo y tareas del Sprint 4

La Tabla 2.7 muestra la duración del Sprint 5, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 5	**22/04/2022 - 19/05/2022**
Objetivo	Completar el Producto Mínimo Viable
Tareas	▪ Enganche y relación entre backend y frontend ▪ Implementar el login de usuarios y el registro

Tabla 2.7: Objetivo y tareas del Sprint 5

La Tabla 2.8 muestra la duración del Sprint 6, con sus respectivas fechas, su objetivo y las tareas a realizar.

Sprint 6	**20/05/2022 - 16/06/2022**
Objetivo	Asegurar el Producto Mínimo Viable
Tareas	▪ Finalizar tareas de debugging ▪ Realizar cambios de interfaz de usuario ▪ Realizar el módulo de gamificación

Tabla 2.8: Objetivo y tareas del Sprint 6

La Tabla 2.9 muestra el inicio del Sprint Extra, con su objetivo y las tareas a realizar.

Sprint 7	**17/06/2022 - 24/06/2022**
Objetivo	Entrega de TFG
Tareas	▪ Finalizar tareas de documentación ▪ Visto bueno del TFG y preparación de la defensa ante tribunal

Tabla 2.9: Objetivo y tareas del Sprint Extra

2.3. Plan de riesgos

Para la realización de un proyecto es necesario elaborar un plan de riesgos [16]. Este proceso hay que realizarlo debido a que en un proyecto siempre existe la incertidumbre y hay que valorar situaciones que pueden hacer desviar al proyecto de la línea idónea.

Para elaborar correctamente un plan de riesgos asociado al proyecto es necesario la identificación y el análisis de riesgos teniendo en cuenta los siguientes factores:

- Probabilidad: posibilidad de que el riesgo se pueda materializar. Se asigna los valores "Baja", "Media", "Alta".
- Impacto: efecto si el riesgo se materializa.
- Plan de mitigación: serie de acciones para reducir la probabilidad de que el riesgo ocurra.
- Plan de contingencia: serie de acciones para minimizar el impacto del riesgo una vez que se ha presentado.

Para la elaboración de riesgos se ha estimado su probabilidad e impacto según un análisis previo y se han acordado los planes de mitigación y de contingencia para registrar la serie de acciones a realizar.

La Tabla 2.10 muestra el Riesgo 01 el cual consiste en la mala estimación de tiempo.

R01	**Mala estimación de tiempo**
Descripción	La implementación de un determinado desarrollo o tarea no se realiza en el tiempo estimado debido a dificultades técnicas a la hora de desarrollar o mala planificación del tiempo.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Obtener el producto mínimo viable en primer lugar y dejar los detalles o las implementaciones menos importantes del proyecto para el final. ▪ Obtener una estimación más real del tiempo, necesaria mayor experiencia en este proceso.
Plan de contingencia	▪ Definir el alcance del proyecto acorde al tiempo disponible.

Tabla 2.10: Riesgo 01

La Tabla 2.11 muestra el Riesgo 02 el cual consiste en baja por enfermedad.

R02	**Baja del estudiante por enfermedad**
Descripción	El estudiante presenta síntomas de enfermedad y no puede lograr un rendimiento acorde a lo previsto.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Evitar actividades peligrosas o que puedan provocar algún tipo de incidente.
Plan de contingencia	▪ Si la duración de la enfermedad es corta, redistribuir las horas perdidas en los próximos días. ▪ Si la duración de la enfermedad es más duradera, traspasar la tarea al siguiente sprint.

Tabla 2.11: Riesgo 02

La Tabla 2.12 muestra el Riesgo 03 el cual consiste en fallos técnicos en dispositivos.

R03	**Fallos técnicos en el equipo de trabajo del estudiante**
Descripción	Los dispositivos del estudiante sufren algún fallo técnico lo que aumenta el tiempo de duración del proyecto.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Realizar backup del proyecto con frecuencia para evitar pérdidas de información. ▪ Utilizar un repositorio online donde alojar el proyecto y la documentación.
Plan de contingencia	▪ Trabajar, si es posible, desde otro dispositivo mientras se arregla el dispositivo principal.

Tabla 2.12: Riesgo 03

La Tabla 2.13 muestra el Riesgo 04 el cual consiste en desconocimiento de las tecnologías.

R04	**Desconocimiento de las tecnologías que hay que utilizar**
Descripción	Las tecnologías utilizadas para el desarrollo de este proyecto son novedosas para el estudiante.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Obtener un nivel adecuado de conocimiento de la tecnología que se va a utilizar antes de empezar a realizar el proyecto.
Plan de contingencia	▪ Aprender mediante la documentación oficial o externa de la herramienta o tecnología a la hora de implementarlo en el proyecto.

Tabla 2.13: Riesgo 04

La Tabla 2.14 muestra el Riesgo 05 el cual consiste en cambio en los requisitos.

R05	**Cambio en los requisitos del proyecto**
Descripción	Los requisitos del proyecto cambian cuando el desarrollo o sprint ha sido ya iniciado lo que dificulta seguir la planificación inicial y provoca alteraciones en el fin del proyecto.
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Tener desde un primer momento claras las funcionalidades importantes y críticas y evitar un continuo cambio de requisitos.
Plan de contingencia	▪ Si es un requisito crítico, desarrollarlo haciendo uso del tiempo planificado para “desvíos”. ▪ Si no es un requisito crítico, omitirlo y proponerlo como investigación futura.

Tabla 2.14: Riesgo 05

La Tabla 2.15 muestra el Riesgo 06 el cual consiste en aplicar metodología Agile con Scrum y la falta de experiencia.

R06	**Aplicar metodología Agile con Scrum**
Descripción	La metodología es novedosa para el estudiante y no se realiza con suficiente soltura para gestionar tiempo y tareas correctamente.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Aprender sobre la metodología Agile con Scrum mediante cursos o charlas antes de empezar a realizar el proyecto.
Plan de contingencia	▪ Aprender sobre la marcha la metodología a base de prueba y error.

Tabla 2.15: Riesgo 06

La Tabla 2.16 muestra el Riesgo 06 el cual consiste dedicar pocas horas al proyecto en algún sprint.

R07	**Poca dedicación al proyecto**
Descripción	Los Sprints establecidos pueden no cumplirse debido a imprevistos en uno en concreto.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Establecer una planificación real atendiendo a posibles imprevistos.
Plan de contingencia	▪ Dedicar tiempo en una franja horaria inamovible para no desviarse del objetivo.

Tabla 2.16: Riesgo 07

2.4. Plan de presupuesto

Como última parte de planificación, es necesario realizar un plan de presupuestos del proyecto. Este plan cuenta con el análisis propio de los costes que puede acarrear el proyecto a distintos niveles. Estos niveles son coste del hardware, coste del software, coste de recursos humanos y coste del software. Existen otros costes que podrían tenerse en cuenta como la electricidad, pero que no se han valorado debido al ínfimo valor monetario que supone respecto de la escala del presupuesto general.

Para valorar el coste amortizado de los distintos elementos analizados, hay que extrapolar al tiempo total del proyecto. Este periodo abarca 5 meses, que abarca desde los inicios reales a finales de enero de 2022 hasta finales de junio de 2022.

2.4.1. Coste del hardware

El coste de hardware calcula el precio de los distintos dispositivos informáticos utilizados para el proyecto. Estos dispositivos son los siguientes:

- Teléfono móvil personal Android Redmi 9T (4GB de RAM y 64GB de almacenamiento): tiene un coste de 179,99€ según la tienda oficial de Xiaomi [17]. Teniendo en cuenta que un teléfono móvil tiene una vida media de 3 años (36 meses) y el proyecto abarca únicamente 5 meses, pues el coste del teléfono es de 24,99€.
- Ordenador portátil Dell - Vostro 3590 (8GB de RAM y procesador i5 de décima generación): tiene un coste de 648,51€ según la tienda PCExpansion [18]. Teniendo en cuenta que un ordenador portátil tiene una vida media de 4 años (48 meses) y el proyecto abarca únicamente 5 meses, el coste del portátil es de 67,55€.
- Monitor ASUS: tiene un coste de 125€ y una vida media de 15 años (180 meses). Por lo tanto, el coste del monitor es de 3,47€.

El coste total del hardware es de 96,01€.

2.4.2. Coste del software

Los programas software utilizados para este proyecto han sido gratuitos en su plenitud debido al hecho de ser estudiante. Este rango ha permitido obtener licencias para algunos programas de manera "gratuita"por parte de la universidad.

Si se extrapola este proyecto a un ambiente empresarial donde los programas tienen su coste y existen varios tipos de licencias o ediciones para cada programa, el coste de software se incrementaría según la opción escogida.

2.4.3. Coste de recursos humanos

El coste de recursos humanos no se aplica en el contexto ya que se trata un proyecto estudiantil.

Por otro lado, sí existiría un coste para este proyecto en un entorno empresarial. El tiempo establecido para realizar este proyecto es de 300 horas según la guía docente de la asignatura. Al tratarse de un desarrollador con perfil junior, el salario medio por hora es de 9,06€ bruto en España según Glassdoor [19], por lo tanto, se obtiene un coste de 2718€.

2.4.4. Coste de infraestructura

La mayoría del trabajo realizado para este proyecto se ha realizado en el hogar personal del estudiante, y para poder realizar el trabajo es necesario contar con conexión a Internet. El plan contratado es de 107,92€ al mes y en esta unidad familiar conviven 4 personas, por lo que el coste de Internet para el estudiante es de 26.98€/mes. Al desarrollar el proyecto en un plazo de 5 meses, el coste de Internet se propaga a 134,90€.

No se ha valorado el precio de una habitación o de una vivienda por lo que el precio total de infraestructura es de 134,90€.

2.4.5. Coste total

Para obtener el coste total del proyecto se han tomado los valores de los costes anteriores y se ha realizado la suma, estableciendo posteriormente un 25 % extra a modo de colchón.

La Tabla 2.17 muestra el coste de los distintos conceptos detallados anteriormente y la suma final del proyecto.

Concepto	**Coste**
Coste de hardware	96,01€
Coste de software	0€
Coste de recursos humanos	2718€
Coste de infraestructura	134,90€
Coste total	**2948,91€**
Coste total + extra	**3686,14€**

Tabla 2.17: Coste total del proyecto

2.5. Seguimiento del proyecto

Durante el transcurso del proyecto se ha realizado un seguimiento del mismo a nivel de tareas y de gestión de tiempo. La Figura 2.1 muestra el tiempo empleado para las tareas importantes o críticas realizadas durante el proyecto. Como se ha comentado anteriormente, el *sprint 1* y el *sprint 7* no son considerados *sprints* reales, de ahí la duración del mismo no se asemeja a la del resto de *sprints*, que duraba aproximadamente un mes.

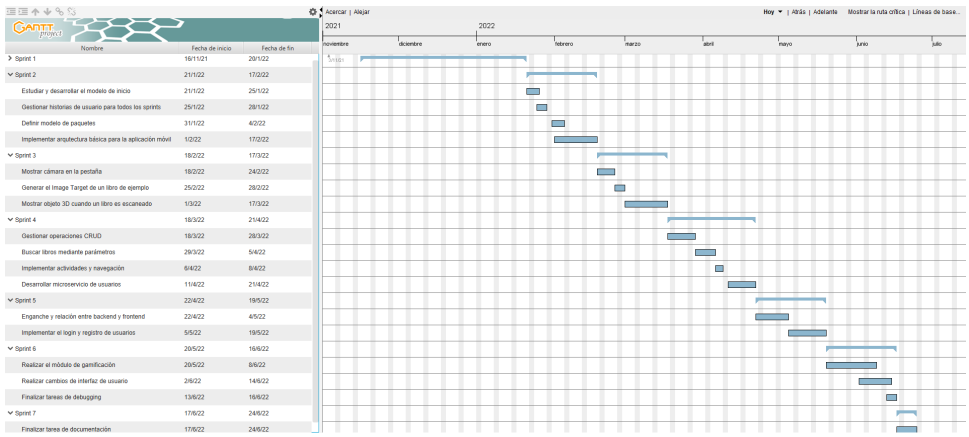

Figura 2.1: Diagrama de Gantt

La Figura 2.2 muestra el tablero de GitLab Issues y cómo se han gestionado las tareas o historias de usuario. Se ha incluido una etiqueta correspondiente al *sprint* a cada una de ellas.

- *Open*: historias de usuario que se encuentran en el *backlog* y aún no se han iniciado.
- *In Progress*: historias de usuario que se han iniciado y se encuentran en progreso.
- *Testing*: historias de usuario que han terminado de implementarse y falta revisión o pulir ciertos aspectos.
- *Closed*: historias de usuario finalizadas.

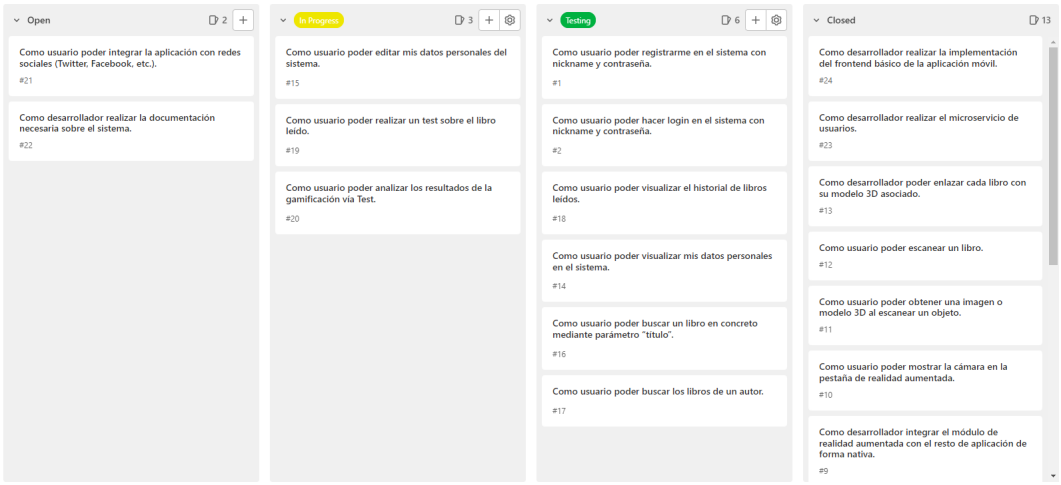

Figura 2.2: Tablero de GitLab Issues con tareas en las columnas

La Tabla 2.18 muestra el trabajo estimado y el trabajo real para cada uno de los *sprints*. La estimación de tareas *software* no es fácil y una mala estimación de tiempo de las tareas puede provocar retrasos o mayor coste en el proyecto. Como se puede apreciar en la Tabla el trabajo real se ha desviado respecto del trabajo estimado debido a riesgos incluidos en la sección 2.3 como mala estimación de tiempo ya que se infravalora el tiempo de ciertas tareas que en el momento de implementar se dificulta.

Sprint	**Trabajo estimado**	**Trabajo real**
Sprint 1	15h	15h
Sprint 2	40h	50h
Sprint 3	45h	55h
Sprint 4	45h	60h
Sprint 5	60h	70h
Sprint 6	70h	80h
Sprint 7	25h	30h
Total	**300h**	**360h**

Tabla 2.18: Calendario final

Capítulo 3

Tecnologías empleadas: Análisis comparativo y selección

Como ya se ha señalado, un aspecto al que se dio mucha importancia en este trabajo, a petición de los tutores de empresa, fue a la selección de tecnologías necesarias para utilizar por parte del estudiante. En este capítulo se presenta un estudio de las tecnologías analizadas y sus alternativas, además de un resumen de las tecnologías utilizadas tanto para la gestión del proyecto, como para el desarrollo de la aplicación.

3.1. Estudio de tecnologías y sus alternativas

Durante una primera fase se realizó el estudio de alternativas en cuanto a tecnologías a utilizar para desarrollar el proyecto. Era importante realizar este estudio para entender y fijar las necesidades del proyecto, además de evitar riesgos en el mismo que puedan implicar retrasos o mayor coste.

Se ha realizado una valoración, basada en el estudio de las características de dichas tecnologías por parte del estudiante. Se han analizado las posibles ventajas y desventajas que supone utilizar cada una de las tecnologías teniendo en cuenta varios factores, como funcionalidades del software, facilidad de aprendizaje de la tecnología, y nivel de experiencia en la misma por parte del estudiante. La valoración se ha realizado en una escala 0 a 10, donde 0 significa que la tecnología no cumple con el criterio y 10 que cumple con él perfectamente.

3.1.1. Planificación de tareas

La planificación de tareas es un aspecto muy importante para el seguimiento de un proyecto software ya que establece las tareas necesarias para lograr un objetivo o un hito.

Para la planificación de tareas existían varias herramientas que se podían utilizar para cumplir con esta funcionalidad. Las herramientas analizadas han sido GitLab Issues[1] , Trello[2], Azure DevOps[3] o Jira[4].

La Figura 3.1 muestra la comparación de las distintas herramientas para la planificación de tareas teniendo en cuenta los criterios y escala ya mencionados.

Herramienta	**Funcionalidad**	**Facilidad de Aprendizaje**	**Experiencia**
GitLab Issues	9	9	8
Trello	9.25	8.25	6
Azure DevOps	8.75	8	4
Jira	9.5	8.5	9

Tabla 3.1: Comparación de herramientas de planificación de tareas. Se ha valorado cada aspecto según una puntuación cuantitativa subjetiva de 0 a 10 (siendo 0 - no cumple el criterio; 10-cumple perfectamente el criterio)

Finalmente, se decidió por utilizar GitLab Issues debido a la integración con el repositorio de GitLab donde se aloja el proyecto.

[1]Más información sobre GitLab Issues: `https://docs.gitlab.com/ee/user/project/issues/`
[2]Más información sobre Trello: `https://trello.com/es`
[3]Más información sobre DevOps: `https://azure.microsoft.com/es-es/services/devops/`
[4]Más información sobre Jira: `https://www.atlassian.com/es/software/jira`

3.1.2. UML

Para el modelado de software mediante diagramas UML existen varias opciones como Astah[5], Visual Paradigm[6], GenMyModel[7] o Draw.io[8].

La Figura 3.2 muestra la comparación de las distintas herramientas para el modelado de software teniendo en cuenta los criterios y escala ya mencionados.

Herramienta	**Funcionalidad**	**Fácil Aprendizaje**	**Experiencia**
Astah	9.5	9.5	9
VP	9.25	8.5	8
GenMyModel	8	8	3
Draw.io	7.5	9	7

Tabla 3.2: Comparación de herramientas de modelado de software. Se ha valorado cada aspecto según una puntuación cuantitativa subjetiva de 0 a 10 (siendo 0 - no cumple el criterio; 10 - cumple perfectamente el criterio)

Finalmente, se decidió por utilizar Astah debido a la experiencia previa, además de contar con la suficiente funcionalidad requerida para realizar el modelado necesario y con una curva de aprendizaje asequible.

[5]Más información sobre Astah: `https://astah.net/`
[6]Más información sobre Visual Paradigm: `https://www.visual-paradigm.com/`
[7]Más información sobre GenMyModel: `https://www.genmymodel.com/`
[8]Más información sobre Draw.io: `https://app.diagrams.net/`

3.1.3. *Frontend*

Una decisión critica en este proyecto se refería a la tecnología para implementar el *frontend* de la aplicación. La elección de esta tecnología supondría una gran diferencia respecto a la metodología de programación y los patrones de diseño a utilizar según el framework elegido y sus características propias.

Por lo tanto, para la parte de *frontend* de la aplicación multiplataforma se dejó a decisión del estudiante elegir qué tecnología utilizar teniendo en cuenta los consejos dados por los tutores del trabajo. Existían varias tecnologías posibles que podían implementar la funcionalidad requerida. Entre ellas se encontraba Angular[9] o MAUI .NET6[10]. Existían otras tecnologías como Flutter[11] o Native Script[12] las cuales también se valoraron en su medida.

La Figura 3.3 muestra la comparación de las distintas tecnologías para el frontend teniendo en cuenta los criterios y escala ya mencionados.

Tecnología	**Funcionalidad**	**Facilidad de Aprendizaje**	**Experiencia**
Angular	9.5	8	8
MAUI .NET6	9.25	8.5	4
Flutter	9	7	1
Native Script	8.75	7.5	2

Tabla 3.3: Comparación de tecnologías frontend. Se ha valorado cada aspecto según una puntuación cuantitativa subjetiva de 0 a 10 (siendo 0 - no cumple el criterio; 10 - cumple perfectamente el criterio)

Finalmente, se decidió por tecnología Angular debido al poco recorrido existente de MAUI en el momento de realizar este estudio, ya que al ser una fase Beta podría tener errores y bugs y se retrasó la fecha de lanzamiento respecto del lanzamiento de .NET6, además de carecer de mucha documentación.

Otro punto a tener en cuenta es que como estudiante, el autor había tenido mayor contacto con Angular. Las tecnologías Flutter y Native Script no presentaban gran competencia por los diversos factores analizados, como la falta de experiencia por parte del estudiante.

[9]Más información sobre Angular: `https://angular.io/`
[10]Más información sobre MAUI: `https://docs.microsoft.com/es-es/dotnet/maui/what-is-maui`
[11]Más información sobre Flutter: `https://flutter.dev/`
[12]Más información sobre Native Script: `https://nativescript.org/`

3.1.4. Módulo de realidad aumentada

Otra decisión importante para este proyecto era la de qué tecnología utilizar para el módulo de realidad aumentada. La tecnología propuesta en un principio fue ARCore[13], tecnología desarrollada por Google la cual es muy potente y permite realizar una gran variedad de funcionalidades. Sin embargo, al basar el *frontend* en tecnología web, se apostó también por la investigación de distintos sistemas que permiten realizar el enganche con la web de forma fácil. Estas tecnologías son AR.js[14], MindAR[15] y EasyAR[16].

La Figura 3.4 muestra la comparación de las distintas tecnologías para el módulo de Realidad Aumentada teniendo en cuenta los criterios y escala ya mencionados.

Tecnología	**Funcionalidad**	**Facilidad de Aprendizaje**	**Experiencia**
ARCore	9.75	6	4.5
AR.js	7	9	4
MindAR	9	9.5	5
EasyAR	8	8.5	3

Tabla 3.4: Comparación de tecnologías de AR. Se ha valorado cada aspecto según una puntuación cuantitativa subjetiva de 0 a 10 (siendo 0 - no cumple el criterio; 10 - cumple perfectamente el criterio)

MindAR tiene una nota alta en cuanto a facilidad de aprendizaje ya que el enganche con Angular se realiza de manera inmediata y la documentación es clara, además de que la funcionalidad requerida para este proyecto era suficiente con una herramienta como MindAR, sin necesitar una funcionalidad máscompleta como podría otorgar ARCore.

Otro punto en contra de ARCore, también llamado Google Play Services for AR, es que tenía limitación en cuanto a dispositivos compatibles[17]

Por otro lado, AR.js también tenía una facilidad de aprendizaje alta al igual que MindAR, pero las pruebas realizadas con la misma mostraron que el módulo de *Image Tracking*[18] no funcionaba de la manera esperada ya que el rendimiento era peor.

Por lo tanto, MindAR era la mejor opción para desarrollar el módulo de realidad aumentada.

[13]Más información sobre ARCore: `https://developers.google.com/ar`
[14]Más información sobre AR.js: `https://ar-js-org.github.io/AR.js-Docs/`
[15]Más información sobre MindAR: `https://hiukim.github.io/mind-ar-js-doc/`
[16]Más información sobre EasyAR: `https://www.easyar.com/`
[17]Más información: `https://developers.google.com/ar/devices`
[18]*Image Tracking* es el proceso por el cual se detecta imágenes planas y se realiza un seguimiento continuo de las posiciones y orientaciones de las imágenes aplicando acciones como mostrar un objeto en 3D.

3.2. Herramientas y tecnologías utilizadas en el proyecto

Las tecnologías utilizadas para este proyecto han correspondido con las recomendadas por HP SCDS, las cuales tenían un requisito imprescindible en cuanto a licencias de uso, que debía ser Licencia MIT [20] o similar.

La licencia MIT da a los usuarios permiso expreso para reutilizar el código para cualquier propósito, a veces incluso si el código forma parte de un software propietario. Siempre que los usuarios incluyan la copia original de la licencia MIT en su distribución, pueden realizar cualquier cambio o modificación en el código para adaptarlo a sus propias necesidades [21]. Es uno de los acuerdos de licencia de código abierto más sencillos. La intención era que el texto fuera comprensible para los usuarios medios y evitar los extensos litigios que pueden surgir de otras licencias similares de software libre y de código abierto (FOSS).

A continuación se lista el resto de tecnologías y herramientas utilizadas en el proyecto, que cumplen con dicha licencia MIT.

3.2.1. Overleaf

Para redactar la memoria del TFG se ha utilizado Overleaf [22]. Es un editor LaTeX colaborativo basado en la nube que se utiliza para escribir, editar y publicar documentos científicos. Overleaf fue creado por John Hammersley y John Lees-Miller, que empezaron a desarrollarlo en 2011.

La Figura 3.1 muestra el logo de Overleaf.

Figura 3.1: Logo de Overleaf

Para la elaboración de la memoria se ha partido de una plantilla con el esqueleto principal de lo que constaría un Trabajo Fin de Grado. Para una mejor organización de los archivos se ha dividido en varios ficheros .tex según los capítulos de la memoria, además de utilizar varios paquetes LaTeX que permite una mejor escritura o adaptación a este software.

La Figura 3.2 muestra la división realizada en Overleaf para el documento LaTeX.

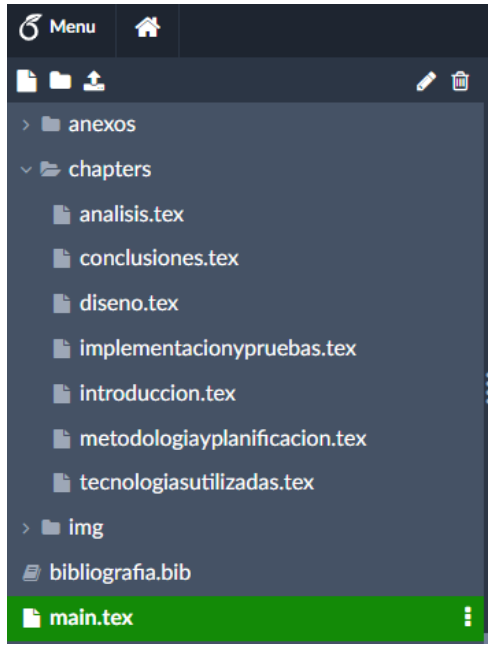

Figura 3.2: División del documento LaTeX

3.2.2. Astah

Para las fases de análisis y diseño de software se ha utilizado Astah [23], que permite modelar rápidamente diagramas para poderlos visualizar en las fases de análisis y diseño.

La Figura 3.3 muestra el logo de Astah.

Figura 3.3: Logo de Astah

Con Astah Professional se pueden construir diversos diagramas UML como modelos ER[19], diagramas de flujo de datos, diagramas de flujo o mapas mentales, entre otros. La interfaz del programa es sencilla de utilizar y con un alto grado de facilidad de aprendizaje, además de ser una tecnología ligera que permite crear los diagramas rápidamente.

La Figura 3.4 muestra una pantalla principal de Astah con dos diagramas, un diagrama de clases y un diagrama de casos de uso, además de mostrar la opción de crear diagrama.

[19]Entidad-Relación

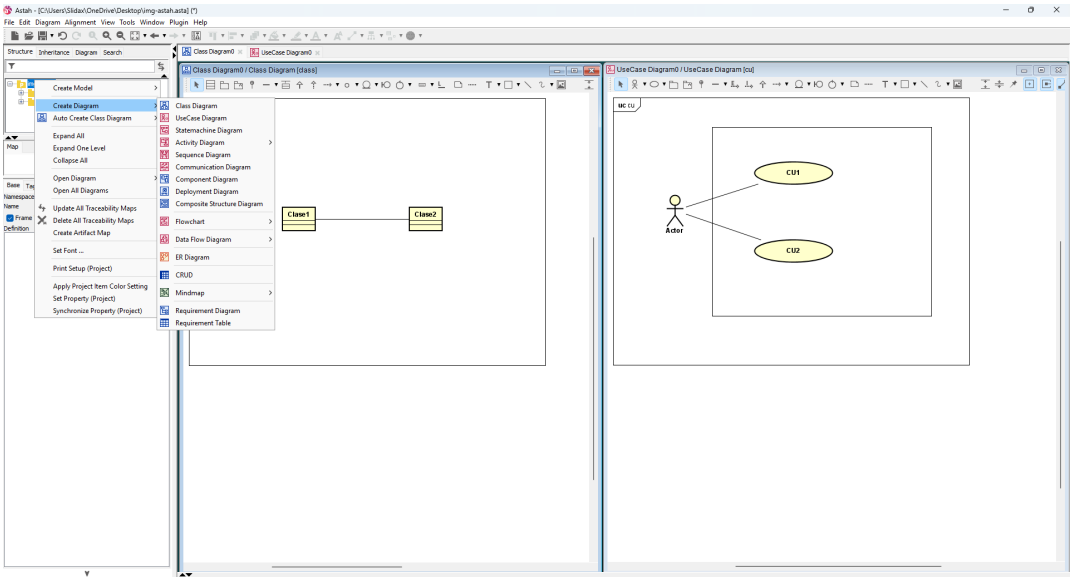

Figura 3.4: Pantalla básica de Astah

3.2.3. Balsamiq

Para realizar los bocetos iniciales del frontend de la aplicación móvil y poder lograr un contexto de los elementos visuales necesarios para construir la aplicación se ha utilizado Balsamiq Wireframes [24], la cual es una herramienta que sirve como guía visual que representa el esqueleto o estructura visual de una aplicación móvil y ha servido para comunicar las ideas iniciales de diseño con los clientes (tutores de HP SCDS) y refinar dicho diseño.

La Figura 3.5 muestra el logo de Balsamiq.

Figura 3.5: Logo de Balsamiq

Balsamiq Wireframes cuenta con una herramienta online denominada Balsamiq Cloud, que permite diseñar y revisar las interfaces de usuario de manera online. Se puede también realizar el diseño entre varios usuarios.

Existen varios planes con su precio concreto según el número de proyectos para un determinado espacio. Existe también una prueba gratuita de 30 días.

La Figura 3.6 muestra una pantalla principal de Balsamiq Cloud con un ejemplo de prueba.

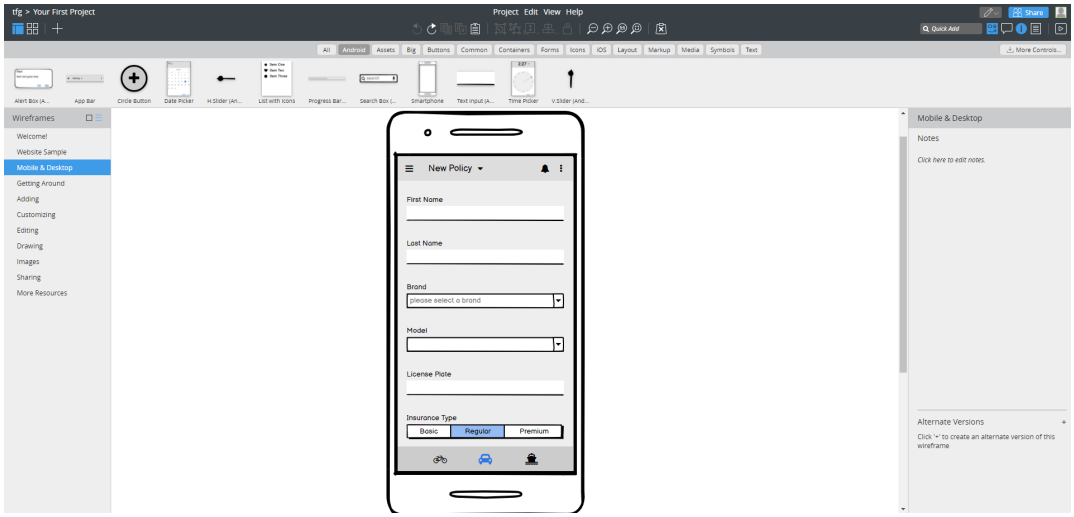

Figura 3.6: Pantalla básica de Balsamiq

3.2.4. .NET6

El backend se ha desarrollado siguiendo un sistema basado en microservicios con tecnología .NET6 [25] con lenguaje C#. Se eligió esta tecnología por tener una curva de aprendizaje asequible, y por recomendación por parte de los tutores de este proyecto. Era una tecnología novedosa para el estudiante por lo que hubo que realizar un estudio previo de la misma, la sintaxis utilizada y el alcance que se puede lograr con este tipo de tecnología.

Algunos datos importantes sobre .NET Core y .NET6 se listan a continuación:

- .NET Core es compatible con C# y F# y con Visual Basic .NET desde la versión 3.0.
- .NET6 fue lanzado el 8 de noviembre de 2022 junto a la nueva versión de Visual Studio 2022 versión 17.0. Esta versión de .NET6 tendrá soporte hasta el 8 de noviembre de 2024.
- Las principales novedades de .NET6 [26] son:
  - Desarrollo simplificado: se reduce la cantidad de código que se necesita escribir en C#, sobre todo en la creación de microservicios más rápidos y pequeños.
  - Mejor Rendimiento y mayor productividad: se reduce el coste de proceso si se ejecuta en la nube y se han implementado nuevas características como la recarga activa, mayor integración con Git y una edición de código inteligente.

La Figura 3.7 muestra el logo de .NET Core.

Figura 3.7: Logo de .NET Core

3.2.5. Base de Datos

Para gestionar la base de datos relacional se ha utilizado el sistema de gestión Microsoft SQL Server [27]. En la actualidad, está disponible para sistemas operativos Windows, Linux e incluso para Docker desde 2017.

Existen varias ediciones según las características necesitadas como Enterprise, Developer, Standard, Express y SQL Azure.

La Figura 3.8 muestra el logo de Microsoft SQL Server.

Figura 3.8: Logo de Microsoft SQL Server

Algunas de las principales características de Microsoft SQL Server son [28]:

- Fácil de instalar: los usuarios puede realizar la instalación de Microsoft SQL Server mediante un asistente de software, en vez de tener que realizar configuraciones extensas en líneas de comandos. Además, la interfaz gráfica de usuario es *user-friendly*, lo que facilita esta tarea.
- Encriptación de datos y seguridad: su sistema de protección y monitorización la ha convertido en una de las plataformas más seguras.
- Escalabilidad y rendimiento: se puede integrar sus sistemas de gestión de bases de datos con cualquier dispositivo y con los servicios de Azure para un mejor rendimiento.

Para administrar la infraestructura de SQL se ha utilizado el entorno integrado de Microsoft SQL Server Management Studio [29]. Con esta herramienta se puede configurar, administrar y desarrollar todos los componentes de Microsoft SQL Server.

La Figura 3.9 muestra el logo de Microsoft SQL Server Management Studio.

Figura 3.9: Logo de Microsoft SQL Server Management Sutdio

SQL Server Management Studio presenta varias características, herramientas y opciones que merece la pena conocer [30]:

- *Object Explorer*: ofrece una vista jerárquica de los objetos del sistema. Suele ser la herramienta más utilizada.
- *Query Editor*: ofrece una herramienta para escribir, ejecutar y *debuggear* consultas SQL. Está equipado con IntelliSense para el autocompletado de código
- *Database Designer*: es una herramienta visual para ayudar al diseño de bases de datos y sus componentes. Permite visualizar cada base de datos con tablas, columnas, restricciones y dependencias.

3.2.6. Angular

Angular [31] es uno de los frameworks más potentes en la actualidad para desarrollo web. Permite crear aplicaciones intuitivas en una sola página (*SPA - Single Page Application*) utilizando las tres tecnologías de desarrollo web principales: HTML, CSS y JavaScript (TypeScript).

Es un *framework* que utiliza el patrón MVC (Modelo-Vista-Controlador), lo que permite la separación de los datos de la representación visual, facilita la mantenibilidad y la resolución de errores, además de proporcionar mayor escalabilidad.

La Figura 3.10 muestra el logo de Angular.

Figura 3.10: Logo de Angular

Las características principales de Angular son las siguientes [32]:

- Aplicaciones multiplataforma:
  - Aplicaciones web progresivas: se pueden implementar soluciones con alto rendimiento y *offline*.
  - Aplicaciones móviles nativas: se puede implementar este tipo de aplicaciones con estrategias como Cordova, Ionic o NativeScript.
  - Aplicaciones de escritorio: se puede crear este tipo de aplicaciones de manera similar que las aplicaciones web, además de poder contar con APIs nativas del sistema operativo.
- Velocidad y rendimiento:
  - Generación de código: Angular permite convertir sus plantillas en código optimizado para las máquinas virtuales JavaScript, ofreciendo las ventajas del código escrito a mano y la productividad de un framework.
  - Universal: se pueden utilizar servidores como Node.js, .NET o PHP para una renderización prácticamente instantánea en sólo HTML y CSS.
  - División del código: las aplicaciones de Angular se cargan rápidamente con el enrutador de componentes, el cual ofrece una división automática del código. Esto consiste en que los usuarios pueden cargar únicamente el código necesario para representar la vista que solicitan.
  - Código reutilizable: el desarrollador puede reutilizar el código debido a que Angular adopta el estándar de componentes web.
- Productividad:
  - Plantillas: Angular permite crear rápidamente vistas de interfaz de usuario con una sintaxis de plantillas sencilla y potente.
  - Angular CLI: herramientas de línea de comandos donde se puede construir rápidamente, añadiendo componentes y pruebas, y desplegarlo al instante.
  - IDEs: obtenga una terminación de código inteligente, errores instantáneos y otros comentarios en editores e IDEs populares.

3.2.7. Ionic

El *frontend* se ha desarrollado haciendo uso de tecnología Ionic, que es un *framework* de código abierto que se integra con Angular para desarrollar aplicaciones híbridas basado en tecnología web como HTML, SCSS y TypeScript. De esta forma, este *framework* permite dar prioridad a la web y, a la vez, desarrollar aplicaciones para iOS, Android y Web desde un único código fuente con un alto rendimiento. Ionic fue fundada en 2012 por los desarrolladores Max Lynch y Ben Sperry [33].

Cuando Ionic se creó, el uso de las tecnologías web como medio para crear aplicaciones nativas estaba en sus inicios. Así, crearon una forma para que los desarrolladores web utilizaran sus habilidades existentes para crear aplicaciones.

En la actualidad, Ionic es la plataforma líder en el mundo para la creación de aplicaciones móviles multiplataforma. Por ejemplo, los productos Ionic han sido utilizados por más de 5 millones de desarrolladores en más de 200 países, impulsando más del 20 % de todas las aplicaciones en las tiendas de aplicaciones.

La Figura 3.11 muestra el logo de Ionic.

Figura 3.11: Logo de Ionic

Las características principales de Ionic son las siguientes [34]:

- Plataforma cruzada: la construcción de una aplicación mediante Ionic permite desplegarla en múltiples plataformas con un código base, por lo que este tipo de aplicaciones se escriben una sola vez.
- Basado en estándares web: Ionic está construido con tecnologías web estándar y modernas APIs web.
- Diseño agradable: Ionic permite construir una aplicación limpia, simple y funcional.
- Licencia MIT: Ionic framework es un proyecto libre y de código abierto que se libera bajo la licencia MIT. Esto significa que podemos utilizarlo en proyectos personales o comerciales de forma gratuita.
- Ionic CLI: herramienta que proporciona varios comandos útiles a los desarrolladores de Ionic. Es un comando que se utiliza para iniciar, construir, ejecutar y emular aplicaciones de Ionic.
- Compatibilidad con el marco de trabajo: Las versiones anteriores de Ionic estaban estrechamente acopladas a Angular. Pero la versión reciente de Ionic, es decir, la v4 fue rediseñada para trabajar como una biblioteca de componentes web independiente, con integración para los últimos *frameworks* de JavaScript. También podemos utilizarlo en la mayoría de los frameworks de front-end, como React.js y Vue.js.
- Angular: es el responsable de hacer grande a Ionic. Mientras que los componentes del núcleo funcionan como una biblioteca de componentes web independiente, el paquete angular hace que la integración con el ecosistema angular sea muy fácil.

3.2.8. MindAR

Para el módulo de realidad aumentada se ha decidido usar una librería web desarrollada para dicho propósito. Esta librería llamada MindAR permite implementar en nuestro código fuente la tecnología necesaria para desarrollar un *Image Tracker* haciendo uso de la cámara de nuestro teléfono móvil o la propia *webcam* desde el equipo con el que se ha desarrollado este proyecto.

La Figura 3.12 muestra el logo de MindAR.

Figura 3.12: Logo de Mind AR

MindAR [35] es una biblioteca de realidad aumentada de código abierto. Admite *Image Tracking* y *Face Tracking*. La simplicidad del código para integrar esta biblioteca a los proyectos lo convierte en una gran tecnología.

3.2.9. Visual Studio

El IDE utilizado para implementar y gestionar los microservicios ha sido Visual Studio [36], tecnología desarrollada por Microsoft. Se utiliza para desarrollar programas informáticos, así como sitios web, aplicaciones web, servicios web y aplicaciones móviles con tecnología Microsoft en su gran mayoría.

Visual Studio es compatible con una gran cantidad de lenguajes de programación diferentes los cuales incluyen C, C++, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML y CSS. Además, cuenta con soporte para otros lenguajes como Python, Ruby y Node.js mediante *plug-ins*.

La Figura 3.13 muestra el logo de Visual Studio.

Figura 3.13: Logo de Visual Studio

Este IDE cuenta con gran cantidad de funcionalidades y características propias de un editor de código como el resaltador de sintaxis, refactorización de código y completitud de código utilizando *IntelliSense*, además de poder utilizar un *debugger* y *designer*.

Visual Studio también presenta varias herramientas que facilitan el trabajo y la navegación por el IDE:

- Navegador de Pestañas Abiertas: para listar todas las pestañas abiertas.
- Editor de Propiedades: se utiliza para editar las propiedades en un panel de la interfaz gráfica de usuario. Enumera todas las propiedades disponibles para todos los objetos, incluidas las clases, los formularios y las páginas web.
- Explorador de Soluciones: una solución es un conjunto de archivos de código y otros recursos que se utilizan para crear una aplicación. Estos archivos están organizados jerárquicamente y el Explorador de Soluciones se utiliza para gestionar y examinar los archivos de una determinada solución.

Visual Studio cuenta, además, con varias ediciones según el modelo de trabajo que el desarrollador o desarrolladores utilizarán:

- Community: esta edición está disponible de manera gratuita y sin ningún cargo. El eslogan de esta edición es ÏDE gratuito y con todas las funciones para estudiantes, desarrolladores de código abierto y particulares".
- Professional: esta edición permite ser utilizada en entorno empresarial, además de tener funcionalidades extra respecto su edición anterior.
- Enterprise: presenta todas las funcionalidades disponibles con las que cuenta Visual Studio para convertirlo en un IDE muy completo.

La Figura 3.14 [37] muestra las diferencias existentes entre las ediciones Community, Professional y Enterprise.

Características admitidas	Visual Studio Comunidad Descarga gratuita	Visual Studio Professional Comprar	Visual Studio Enterprise Comprar
⊕ Escenarios de uso admitidos	●●●○	●●●●	●●●●
Compatibilidad con la plataforma de desarrollo [2]	●●●●	●●●●	●●●●
⊕ Entorno de desarrollo integrado	●●●○	●●●○	●●●●
⊕ Depuración y diagnóstico avanzados	●●○○	●●○○	●●●●
⊕ Herramientas de pruebas	●○○○	●○○○	●●●●
⊕ Desarrollo multiplataforma	●●○○	●●○○	●●●●
⊕ Herramientas y características de colaboración	●●●●	●●●●	●●●●

Figura 3.14: Comparación de las distintas ediciones de Visual Studio

3.2.10. Visual Studio Code

El editor de código utilizado para implementar el frontend con Ionic ha sido Visual Studio Code, permitiendo una rápida ejecución en modo desarrollo. Esta tecnología también ha sido desarrollada por Microsoft.

La Figura 3.15 muestra el logo de Visual Studio Code.

Figura 3.15: Logo de Visual Studio Code

Las características principales de Visual Studio Code son las siguientes [38]:

- Editor de código rápido y ligero: Visual Studio Code permite editar, construir y hacer debug con facilidad, con todas las herramientas necesarias para hacer el trabajo diario.
- Customizable: existe una infinidad de integraciones y extensiones fácilmente instalables para poder utilizarlo dentro del editor de código.
- Arquitectura robusta: Visual Studio Code combina lo mejor de las tecnologías web, nativas y de lenguajes específicos. Gracias a Electron, VS Code combina tecnologías web como JavaScript y Node.js con la velocidad y la flexibilidad de las aplicaciones nativas.
- Multiplataforma: Visual Studio Code se encuentra disponible en diversas plataformas como macOS, Linux y Windows.

3.2.11. GitLab

Como repositorio remoto de código, se ha decidido utilizar GitLab, el cual es un repositorio de código abierto y una plataforma de desarrollo de software colaborativo para grandes proyectos de DevOps.

Git es un sistema de control de versiones distribuido de código abierto diseñado. GitLab está construido él. Además, se puede realizar muchas operaciones de Git directamente desde GitLab.

Fue la herramienta sugerida por los tutores para alojar el código y poder tener todos los proyectos de HP SCDS unificados en un mismo punto.

GitLab ofrece alojamiento para el almacenamiento de código en línea y el repositorio permite a los usuarios inspeccionar el código de anteriores versiones y volver a él en caso de problemas imprevistos.

La Figura 3.16 muestra el logo de Gitlab.

Figura 3.16: Logo de GitLab

Para la gestión de tareas y de planificación se ha decidido utilizar Gitlab Issues, una herramienta integrada con Gitlab que permite realizar esta gestión de manera satisfactoria y fácil de usar.

Esta herramienta cuenta con el tablero Kanban para gestionar los estados por los que pasa la tarea y conserva el estado actual.

El tablero permite añadir hitos y etiquetas a las tareas para una mejor organización y planificación y las columnas que corresponden a los estados se pueden configurar para modificar los ya existentes o crear estados nuevos.

3.2.12. Postman

Para comprobar y validar las peticiones CRUD (GET, POST, PUT, DELETE) de los microservicios se ha utilizado la plataforma Postman, la cual es un cliente HTTP. Esta tecnología permite al desarrollador realizar las pruebas API de su aplicación.

La Figura 3.17 muestra el logo de Postman.

Figura 3.17: Logo de Postman

La API de Postman permite realizar todas las operaciones CRUD clásicas en las colecciones, entornos y mocks. De esta forma, es una manera simple de comprobar las peticiones y las respuestas de una determinada API.

La Figura 3.18 muestra la pantalla principal de Postman.

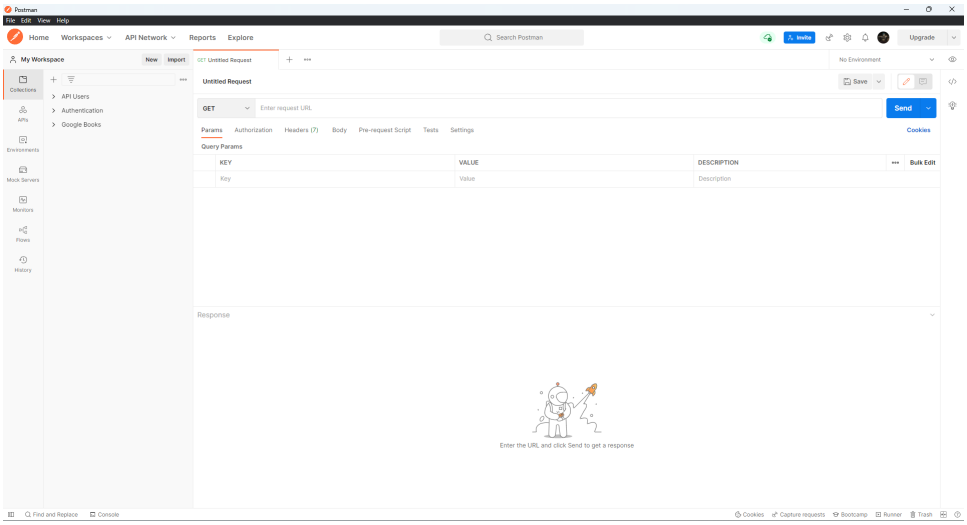

Figura 3.18: Pantalla principal de Postman

Capítulo 4

Análisis

4.1. Descripción del sistema

El proyecto consiste en crear una aplicación móvil multiplataforma que permita a los usuarios, niños con edades comprendidas entre 6 y 12 años, poder buscar información sobre libros apropiados para su edad haciendo uso de una API[1], obtener detalles propios de un libro en específico y realizar un test sobre los libros para gamificar la lectura. Además, se quiere integrar con tecnología de realidad aumentada para hacer más interactivo el proceso de lectura.

4.1.1. Rol

El rol principal y básico es el usuario por defecto que representa a un niño, el cual puede registrarse en la aplicación mediante un sistema básico de autenticación y, a partir de ahí, tener acceso al resto de la aplicación. Este usuario tiene las funcionalidades totales que están pensadas para realizar en la aplicación.

No se ha definido un rol para el adulto o tutor encargado de supervisar al niño ya que no se ha establecido un control parental en los requisitos del sistema, pero sí puede supervisar desde la propia cuenta del niño.

4.2. Elicitación de requisitos

La elicitación de requisitos es el proceso mediante el cual se analiza y se comprende los requisitos del sistema que se desea.

[1] *Application Programming Interfaces*. Más información: `https://www.xataka.com/basics/api-que-sirve`

De esta forma, se establece un procedimiento que es el ciclo de vida de la ingeniería de software donde existen varias etapas en las que hay que estructurar y documentar lo realizado con el producto para la satisfacción del cliente.

Los objetivos de la elicitación de los requisitos son:

- Conocer y comprender el dominio del problema.
- Identificar las necesidades y expectativas de usuarios.
- Establecer una primera aproximación real sobre el desarrollo del sistema.
- Evitar cambios posteriores que puedan dificultar el seguimiento de la planificación.

4.2.1. Requisitos funcionales

Los requisitos funcionales describen los servicios y el funcionamiento que debe ofrecer el sistema, es decir, cómo interacciona el sistema con el entorno y qué debe hacer el sistema.

- **RF01:** El usuario debe poder realizar login en el sistema con nombre de usuario y contraseña.
- **RF02:** El usuario debe poder registrarse en el sistema.
- **RF03:** El usuario debe poder buscar un libro determinado por su título haciendo uso de la API de Google Books.
- **RF04:** El usuario debe poder buscar un libro determinado por su autor haciendo uso de la API de Google Books.
- **RF05:** El usuario debe poder marcar en la aplicación un libro como leído.
- **RF06:** El usuario debe poder visualizar los libros marcados como leído.
- **RF07:** El usuario debe poder visualizar los detalles de un determinado libro.
- **RF08:** El usuario debe poder visualizar sus datos personales.
- **RF09:** El usuario debe poder editar sus datos personales.
- **RF10:** El usuario debe poder escanear la portada de un libro y obtener un objeto en 3D relacionado con dicho libro.
- **RF11:** El usuario debe poder realizar un quiz sobre un libro en concreto.
- **RF12:** El usuario debe poder visualizar el resultado del quiz del libro.
- **RF13:** El usuario debe poder visualizar cerrar sesión en el sistema.
- **RF14:** El usuario debe poder retroceder a la pantalla anterior.

4.2.2. Requisitos no funcionales

Los requisitos no funcionales son las restricciones y las propiedades del sistema en términos de rendimiento, seguridad y disponibilidad, es decir, explica cómo el sistema hace lo que debe hacer.

- **RNF01:** Las tecnologías utilizadas para desarrollar este proyecto deben tener licencia MIT [20].
- **RNF02:** La aplicación se debe poder migrar a distintas plataformas tales como Android e IOS.
- **RNF03:** La aplicación debe ser segura en cuanto a temas de privacidad para no violar el RGPD [39].
- **RNF04:** La aplicación debe ajustarse a la clasificación por edades propia de PEGI [40].
- **RNF05:** La aplicación debe tener unos tiempos de respuesta imperceptibles para que la interacción persona-sistema sea fluida.

4.3. Casos de uso

Para expresar los requisitos funcionales de modo semi-formal es necesario describir el funcionamiento del sistema mediante casos de uso. Un caso de uso es, por lo tanto, un conjunto de escenarios relacionados con el resultado que el actor espera obtener.

Un escenario es una secuencia de acciones realizadas por el sistema, donde un actor es conducido a un resultado valioso. Un actor es cualquier entidad que se comunica con el sistema. En este caso, el actor correspondería al usuario de la aplicación.

La Figura 4.1 muestra el diagrama de casos de uso.

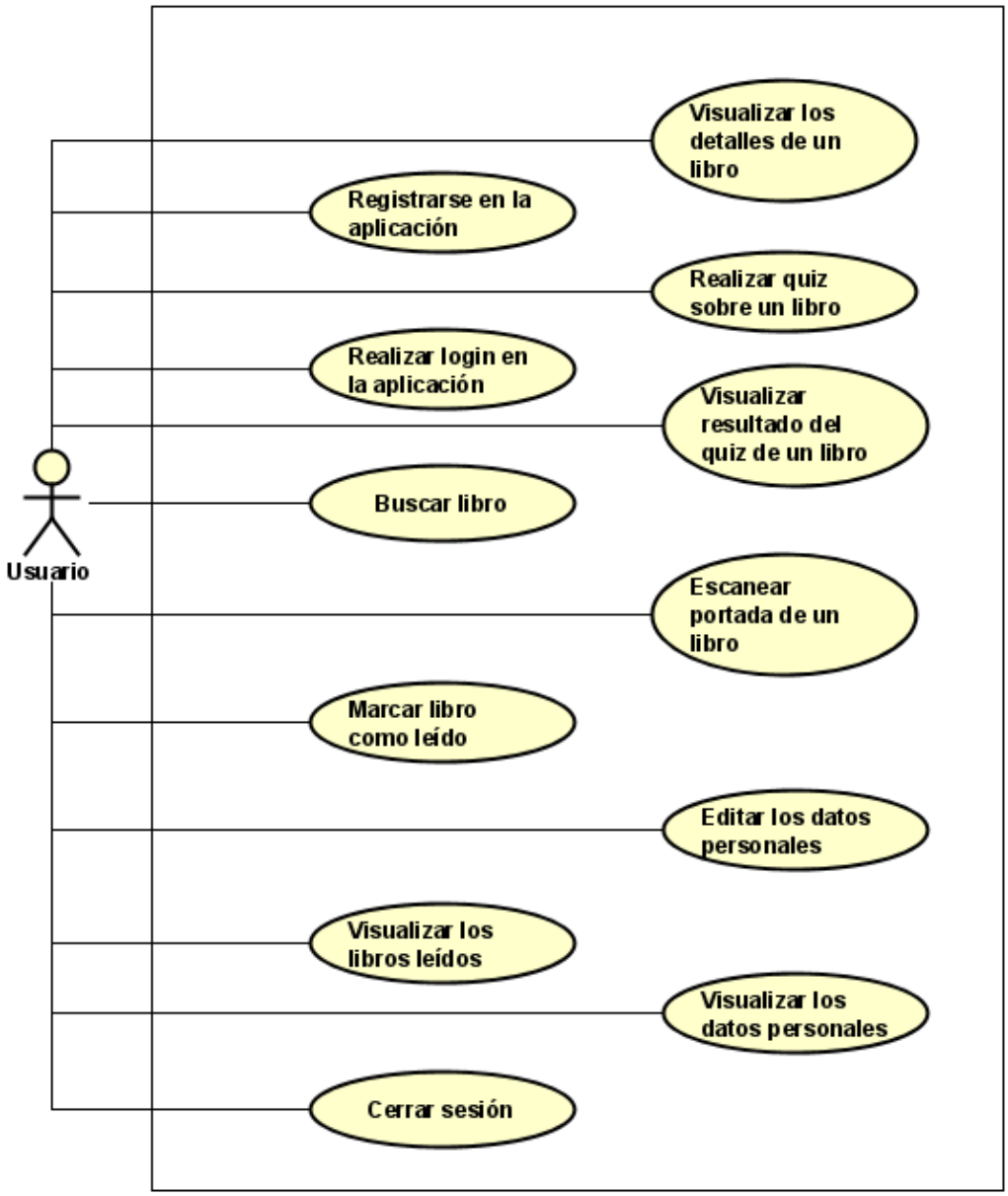

Figura 4.1: Diagrama de casos de uso

Capítulo 5

Diseño

5.1. Diseño de la interfaz del sistema

Se ha prestado atención al diseño de la interfaz para lo que se han tenido en cuenta principios generales (descritos en 5.1.1) y se realizaron prototipos que fueron validados con los clientes. Debido a las características y alcance del proyecto, no se validó con usuarios finales, aunque habría sido lo ideal.

5.1.1. Principios de diseño utilizados para una buena usabilidad

Al tratarse de una aplicación móvil orientada a niños, es necesario disponer de una interfaz clara y sencilla para dichos usuarios por lo que hay que tener en cuenta principios de diseño claves, que consisten en conceptos de un nivel de abstracción muy alto que guían el diseño de interfaces de usuario [41]:

- Sencillez: emplear acciones, iconos y palabras acordes a la edad de los usuarios. Por lo tanto, es necesario establecer controles naturales y evitar tareas complejas.
- Estructura: las características relacionadas deben aparecer juntas y viceversa. Para ello, es necesario separar claramente los 3 puntos siguientes:
  - Información personal del usuario.
  - Módulo de realidad aumentada.
  - Información del libro y sus detalles.
- Consistencia: establecer uniformidad en la apariencia. El usuario se hace un modelo mental de la interfaz y se espera misma respuesta de todas las aplicaciones. Por ello, es necesario basarse en aplicaciones anteriores desarrolladas para el mismo público.
- Tolerancia: consiste en prevenir posibles errores de usuario.

5.1.2. Bocetos del sistema

La Figura 5.1 muestra el boceto de la pantalla de login de la aplicación móvil. En esta pantalla se introduciría los datos del usuario para acceder al sistema.

Figura 5.1: Boceto de la pantalla de login

La Figura 5.2 muestra el boceto de la pantalla de registro de la aplicación móvil. En esta pantalla se introduciría los datos personales para registrar el usuario y acceder al sistema.

Figura 5.2: Boceto de la pantalla de registro

La Figura 5.3 muestra el boceto de la pantalla principal de búsqueda de libros. En esta pantalla se buscaría los libros mediante diferentes parámetros y listaría los libros.

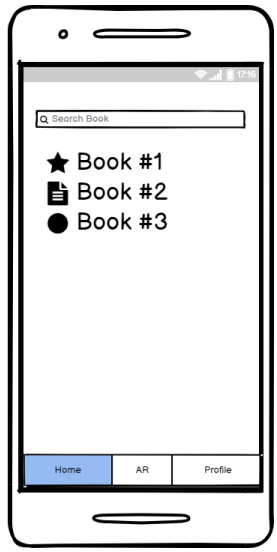

Figura 5.3: Boceto de la pantalla de búsqueda de libros

La Figura 5.4 muestra el boceto de los detalles del libro. En esta pantalla se podría visualizar campos importantes del libro como el nombre del libro, su autor, una imagen de la portada y una pequeña descripción del libro.

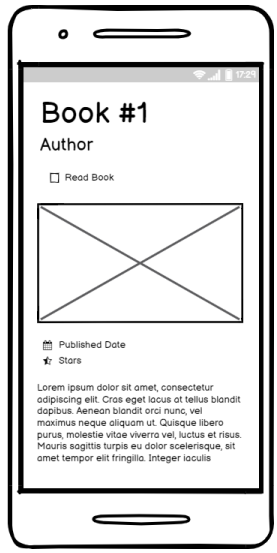

Figura 5.4: Boceto de la pantalla de detalles del libro

La Figura 5.5 muestra el boceto del módulo de realidad aumentada donde se accedería a la cámara para escanear la portada de un determinado libro.

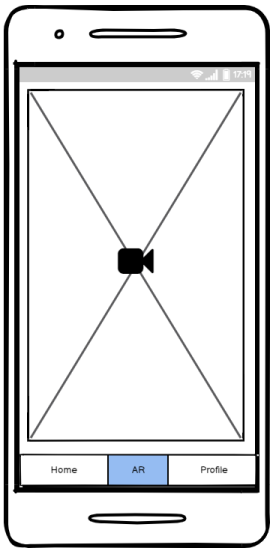

Figura 5.5: Boceto de la pantalla del módulo de realidad aumentada

La Figura 5.6 muestra el boceto de la pantalla de los datos personales del usuario, además de botones para cerrar sesión, para acceder a la pantalla de los libros leídos por el usuario y para acceder a editar la información personal.

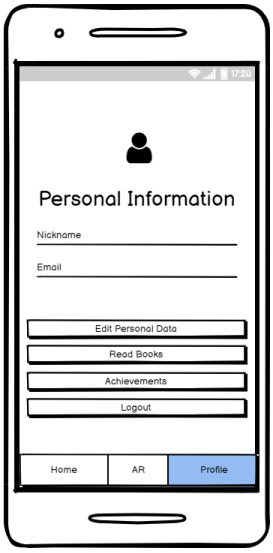

Figura 5.6: Boceto de la pantalla de visualización de datos personales

La Figura 5.7 muestra el boceto de la pantalla del quiz donde se realizan preguntas y respuestas sobre el libro.

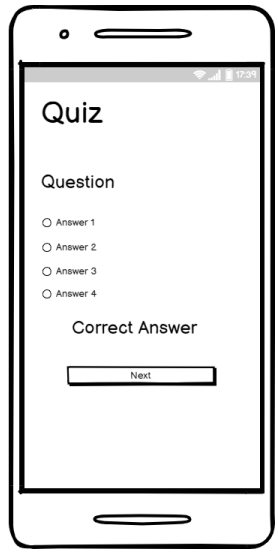

Figura 5.7: Boceto de la pantalla de Quiz

5.2. Arquitectura del sistema

5.2.1. Patrón arquitectónico MVC

El patrón Modelo-Vista-Controlador es un estilo de arquitectura software que separa los datos de la aplicación (Modelo), la interfaz de usuario (Vista) y la lógica de control (Controlador).

Al haber utilizado Ionic para el desarrollo de la aplicación y, por consiguiente, el framework Angular se implementa con facilidad este patrón ya que por defecto el framework realiza esta separación. Además, también se implementan servicios que se encargan de la consulta de datos.

5.2.2. Arquitectura basada en capas

La Figura 5.8 muestra la arquitectura del sistema basada en capas.

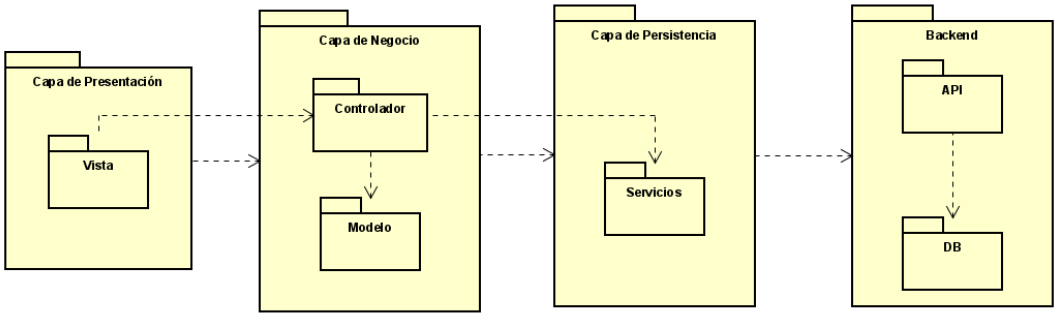

Figura 5.8: Arquitectura del sistema basada en capas

5.2.3. Diagrama de clases

La Figura 5.9 muestra el modelo de dominio inicial.

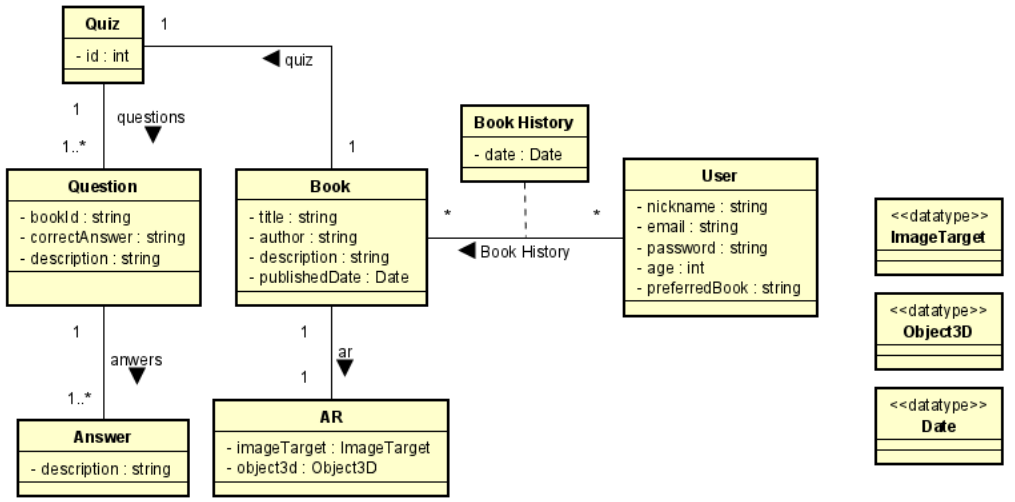

Figura 5.9: Modelo de dominio inicial

5.3. Diseño de los microservicios

Las Figuras 5.10, 5.12, 5.14 y 5.16 muestran las bases de datos de cada microservicio.

Las Figuras 5.11, 5.13, 5.15 y 5.17 muestran los diagramas de recursos REST para cada microservicio.

5.3.1. Microservicio Login

Diseño de la base de datos

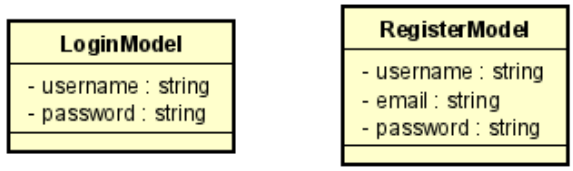

Figura 5.10: Base de datos del microservicio Login

Diagrama de recursos

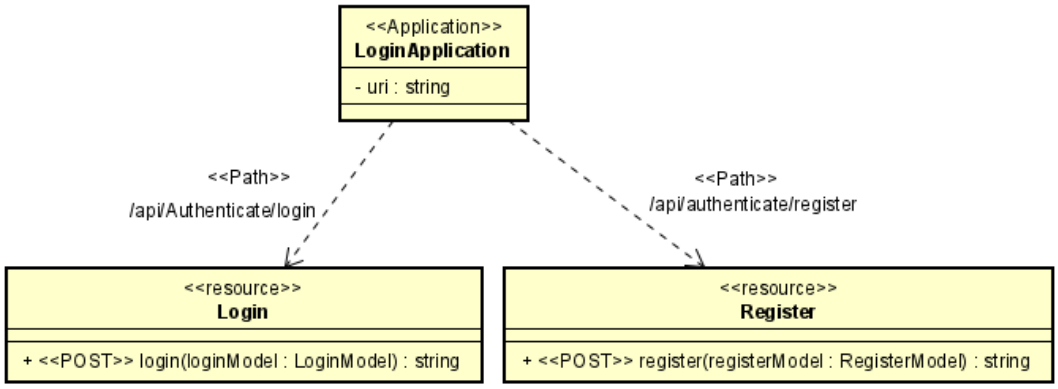

Figura 5.11: Diagrama de servicios REST del microservicio Login

5.3.2. Microservicio User

Diseño de la base de datos

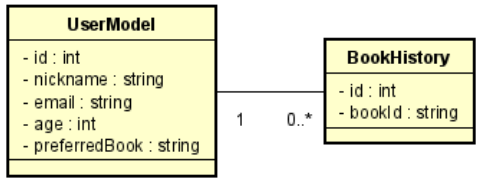

Figura 5.12: Base de datos del microservicio User

Diagrama de recursos

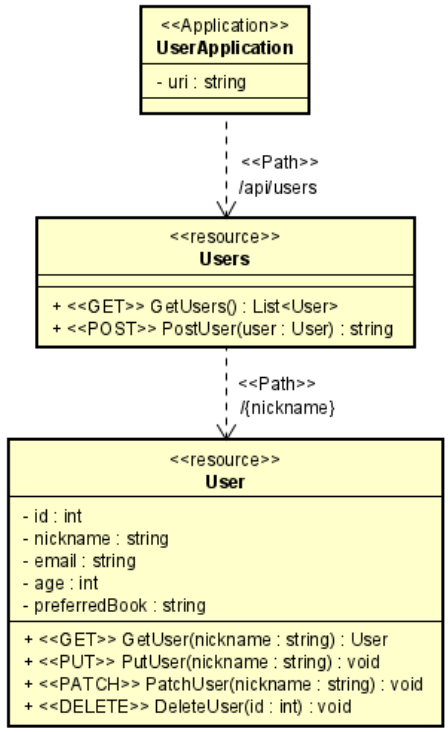

Figura 5.13: Diagrama de servicios REST del microservicio User

5.3.3. Microservicio Book

Diseño de la base de datos

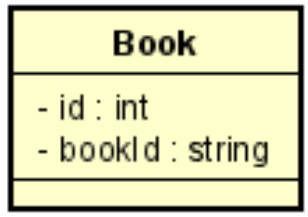

Figura 5.14: Base de datos del microservicio Book

Diagrama de recursos

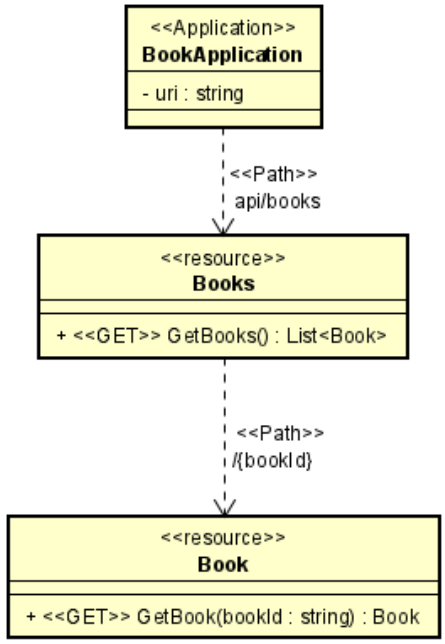

Figura 5.15: Diagrama de servicios REST del microservicio Book

5.3.4. Microservicio Quiz

Diseño de la base de datos

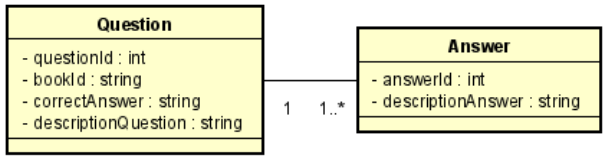

Figura 5.16: Base de datos del microservicio Quiz

Diagrama de recursos

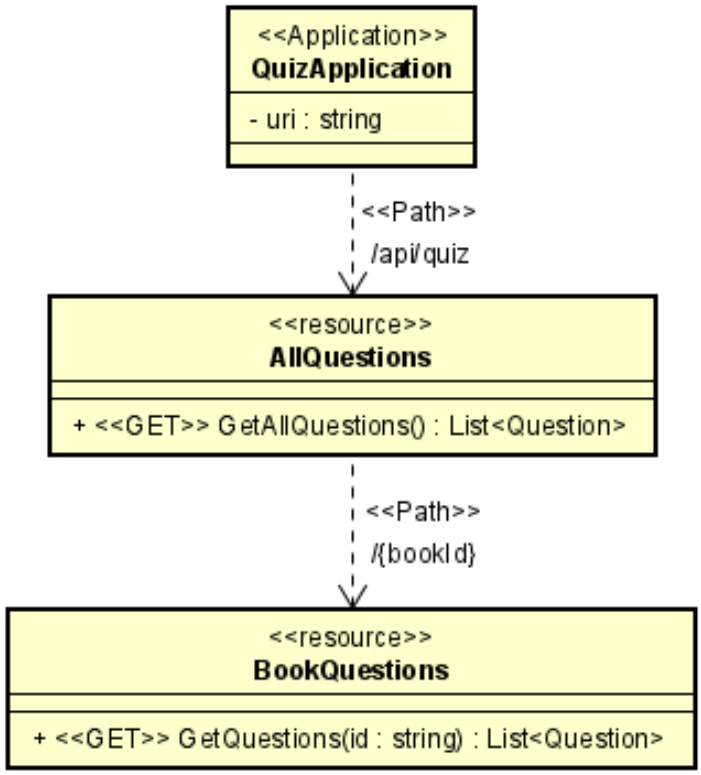

Figura 5.17: Diagrama de servicios REST del microservicio Quiz

5.4. API de búsqueda de libros

Durante el diseño del proyecto fue necesario utilizar una API pública que gestionara el módulo de libros, y que la aplicación pudiera realizar búsquedas de libros y devolviera información de ellos.

Se tomó, por lo tanto, una decisión de diseño basada en las necesidades de la aplicación. La API seleccionada para realizar este módulo fue la API de Google Books[1].

La Figura 5.18 muestra el diagrama de recursos de la API de Google Books.

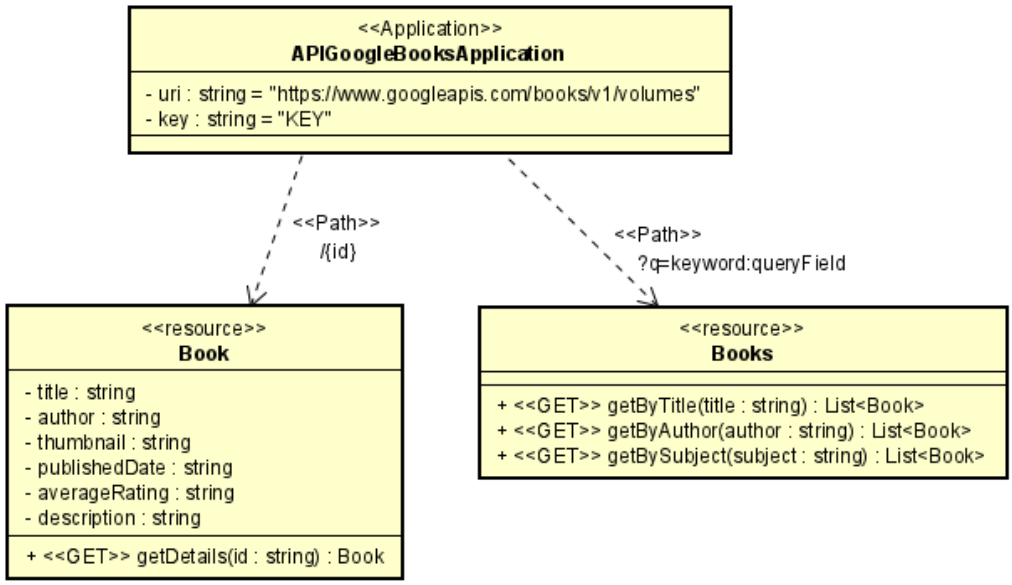

Figura 5.18: Diagrama de recursos de la API de Google Books

[1]Más información: `https://developers.google.com/books`

5.5. Desarrollo basado en componentes

El desarrollo basado en componentes es un enfoque del desarrollo de software que se centra en el diseño y el desarrollo de componentes reutilizables rompiendo la arquitectura monolítica.

5.5.1. Componentes en el proyecto Ionic

La Figura 5.19 muestra la distribución de los componentes del proyecto Ionic y la relación entre ellos.

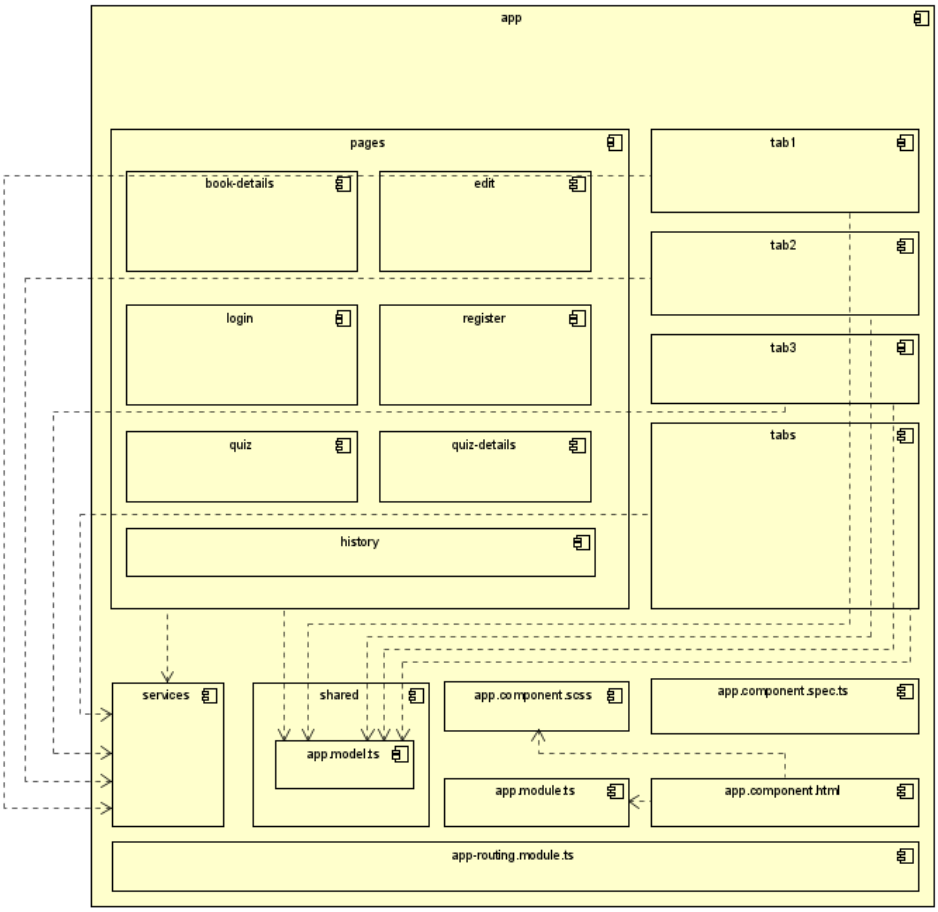

Figura 5.19: Componentes en el proyecto Ionic

5.5.2. Componente típico de Ionic

Ionic basado en Angular implica que tiene el mismo comportamiento y la misma filosofía de implementación, por lo tanto, está pensado para realizar el desarrollo basado en componentes.

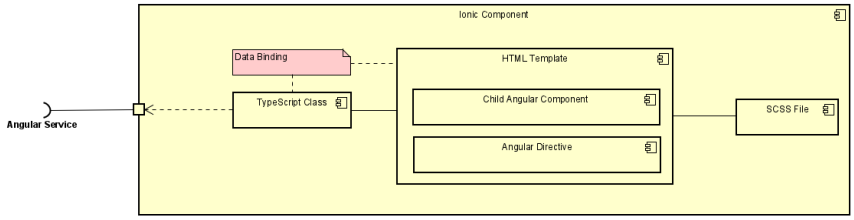

Figura 5.20: Componente típico de Ionic

5.5.3. Componentes creados en el proyecto

Los componentes creados en el proyecto Ionic son de dos tipos: página o servicio.

La Figura 5.21 muestra la estructura de un componente (página) Ionic con el contenido. Todos los componentes del proyecto de Ionic respetan esta estructura y nombrado. La única sustitución es la X que aparece en el diagrama por el nombre real de los componentes explicados abajo.

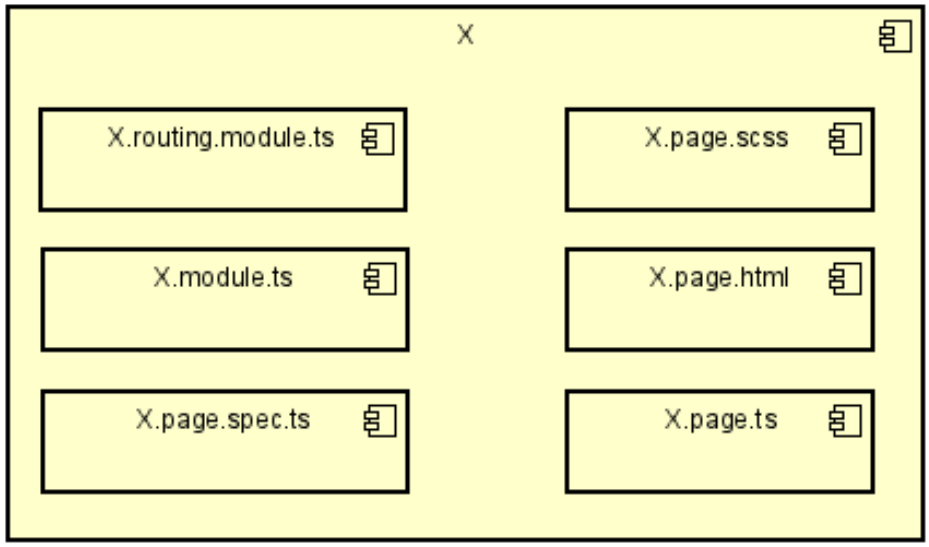

Figura 5.21: Componente Ionic y su contenido

Dichos componentes son:

- book-details: muestra los detalles del libro seleccionado como el título, el autor, portada del libro, valoración del libro, categoría y descripción.
- edit: permite editar campos del usuario.
- history: listado de los libros leídos por el usuario.
- login: pantalla para iniciar sesión en la aplicación.
- quiz: muestra el cuestionario del libro seleccionado, con sus preguntas y respuestas.
- quiz-details: muestra el informe final de lo obtenido en el cuestionario realizado anteriormente.
- register: pantalla para registrarse en la aplicación con los datos personales.
- tab1: muestra el módulo de búsqueda de libros.
- tab2: muestra el módulo de realidad aumentada.
- tab3: gestiona la parte del usuario mostrando la información personal y redirigiendo a distintas pantallas relacionadas con el mismo.
- tabs: gestiona las 3 *tabs* de la aplicación.
- app: aplicación en su conjunto. Alberga todos los componentes.

5.5.4. Modelo del *frontend*

La Figura 5.22 muestra el modelo del *frontend*.

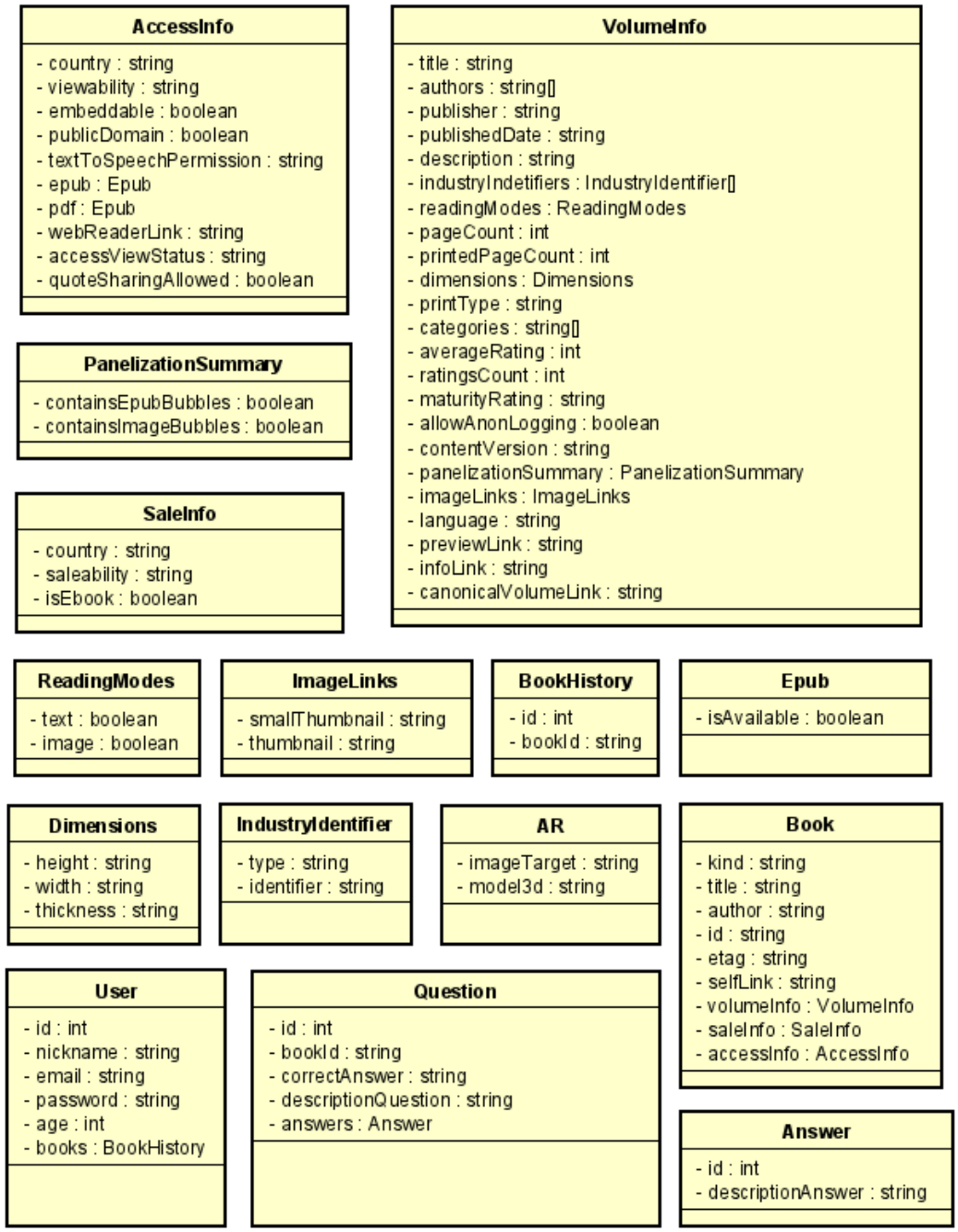

Figura 5.22: Modelo del *frontend*

5.6. Diagrama de despliegue

Un diagrama de despliegue UML es un diagrama que muestra la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que viven en ellos. Los diagramas de despliegue son un tipo de diagrama de estructura utilizado para modelar los aspectos físicos de un sistema orientado a objetos. Se suele utilizar para modelar la vista de despliegue estática de un sistema, es decir, la topología del hardware [42].

La Figura 5.23 muestra el diagrama de despliegue del sistema.

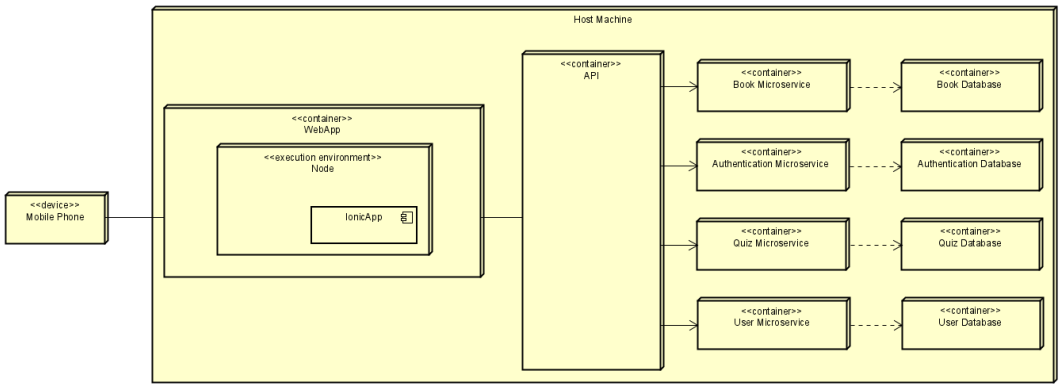

Figura 5.23: Diagrama de despliegue

Capítulo 6

Implementación

Este capítulo describe el proceso de implementación que se ha llevado a cabo en el proyecto, tanto a nivel de microservicios, como el módulo de realidad aumentada y la implementación *frontend*.

6.1. Implementación de los microservicios

La creación del esqueleto de los distintos microservicios se ha realizado de la misma forma para todos ellos. Para la explicación de las imágenes y código de esta sección se ha tomado un microservicio en concreto como ejemplo, pero la extrapolación al resto de microservicios se haría de manera similar, cambiando los nombres que hacen referencia al microservicio en concreto.

En primer lugar, es necesario Visual Studio 2022 para crear aplicaciones .NET 6.0. Al crear un nuevo proyecto en Visual Studio 2022 aparecen muchas plantillas para diferentes proyectos que se pueden crear con este IDE. En el caso de los microservicios, hay que escoger la plantilla ASP.NET Core Web API y rellenar la información adicional para la solución como el nombre del mismo y la versión .NET 6.0.

La Figura 6.1 muestra la estructura básica de los microservicios y cómo se divide en varios paquetes, que realizan diferentes funciones.

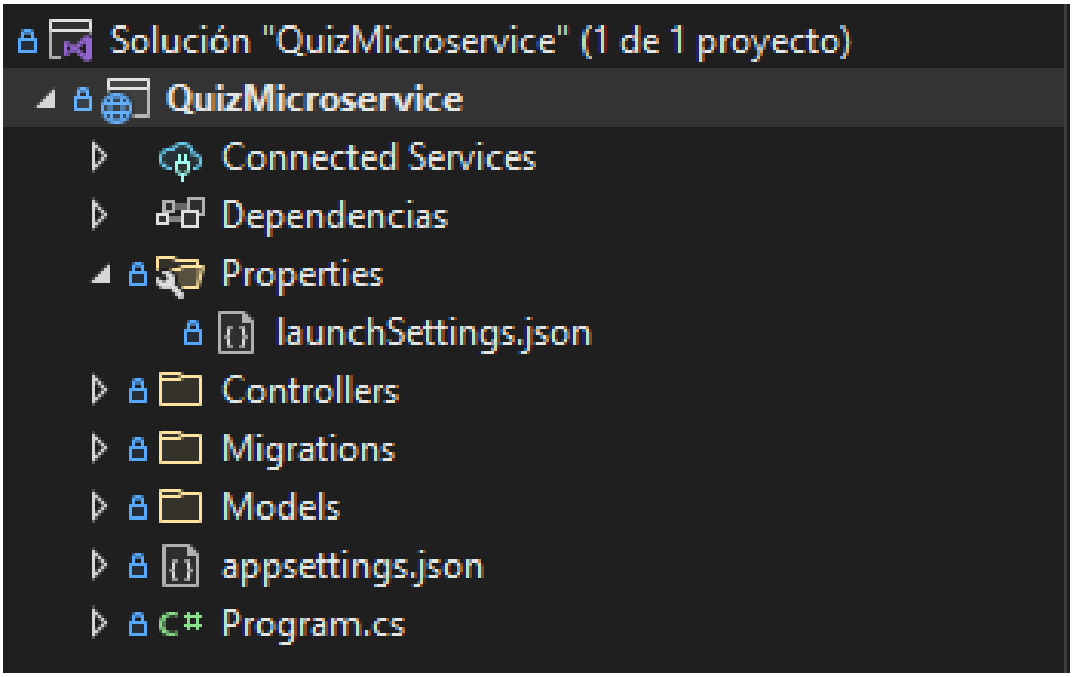

Figura 6.1: Estructura de un microservicio genérico

- Connected Services: en este módulo se gestionan los servicios conectados. En el caso de este proyecto, consiste en el servicio de base de datos de SQL Server.
- Dependencies: en este módulo se muestran y gestionan los analizadores, los marcos de trabajo y los paquetes instalados.
- Properties: este paquete contiene la información para el despliegue de la aplicación.
- Controllers: este paquete contiene los controladores del microservicio donde se implementa la lógica de las peticiones CRUD.
- Migrations: este paquete no aparece de inicio al crear el proyecto y contiene dos archivos que se crean automáticamente al realizar el comando que se explicará posteriormente. Este paquete el que se encarga de la creación y conexión con la base de datos.
- Models: este paquete contiene los modelos del microservicio con sus atributos y métodos *getter* y *setter*.
- appsettings.json: este archivo contiene la información añadida para especificar el nombre de la base de datos y el *server* al que se conecta el microservicio.
- Program.cs: este archivo sirve para definir las inyecciones de dependencias y otras configuraciones necesarias para el microservicio. En versiones anteriores de .NET ciertas configuraciones se realizaban desde un archivo denominado Startup.cs

Una vez creado el proyecto es necesario instalar ciertas librerías para la conexión con SQL Server y la gestión de CORS[1]. Se instalan mediante "Administrador de paquetes de NuGet".

- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.AspNetCore.Cors

A continuación se va a mostrar brevemente cada uno de los archivos de los que está compuesto la solución.

La estructura básica del archivo *Program.cs* es la siguiente.

```
using Microsoft.EntityFrameworkCore;
using QuizMicroservice.Models;
using System.Configuration;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddDbContext<QuizContext>(opt =>
{
    opt.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"));
});


builder.Services.AddCors(c =>
{
    c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin());
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (builder.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseCors("AllowOrigin");

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

[1] *Cross-origin resource sharing*. Más información: `https://developer.mozilla.org/es/docs/Web/HTTP/CORS`

La estructura básica del archivo *launchsettings.json* es la siguiente. Como se puede ver, se especifica la url de despliegue, variables de entorno y más información extra que se puede personalizar.

```
{
  "\$schema": "https://json.schemastore.org/launchsettings.json",

  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:17323",
      "sslPort": 44325
    }
  },
  "profiles": {
    "QuizMicroservice": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "api/quiz",
      "applicationUrl": "https://localhost:7075;http://localhost:5075",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "swagger",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

La estructura básica del archivo *appsettings.json* es la siguiente. Como se puede ver, muestra la información para la conexión al servidor y base de datos. También hay información de *logging*.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "server=localhost\\sqlexpress; database=quizdb;
    ↪  trusted_connection=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

En el paquete *Models* se alojan las diferentes clases que forman parte de la solución y se comportan como modelos. Se implementan los distintos atributos de cada clase y la relación existente con otras.

```
namespace QuizMicroservice.Models
{
    public class QuestionQuiz
    {
        public int Id { get; set; }
        public string? BookId { get; set; }
        public string? CorrectAnswer { get; set; }
        public string? DescriptionQuestion { get; set; }
        public virtual ICollection<AnswerQuiz>? Answers { get; set; }
    }
}
```

En el paquete *Models* también se aloja la clase *DataContext*. Deriva de la clase *DbContext* de *Entity Framework* y permite la conexión entre el modelo (entidades y relaciones) con el paquete instalado anteriormente denominado *Entity Framework*, que sirve para consultar, insertar, actualizar y eliminar datos mediante objetos .NET. Se utiliza, por lo tanto, para acceder a los datos de la aplicación a través de *Entity Framework*.

```
using Microsoft.EntityFrameworkCore;
using System.Diagnostics.CodeAnalysis;

namespace QuizMicroservice.Models
{
    public class QuizContext : DbContext
    {
        public QuizContext(DbContextOptions<QuizContext> options)
            : base(options)
        {}

        public DbSet<QuestionQuiz> QuestionItems { get; set; } = null!;
        public DbSet<AnswerQuiz> AnswerItems { get; set; } = null!;
    }
}
```

Una vez que ya se ha definido el modelo y el contexto, hay que realizar los siguientes comandos para crear la base de datos. Para esto es necesario tener instalado los paquetes *Microsoft.EntityFrameworkCore.Tools* y *Microsoft.EntityFrameworkCore.SqlServer* [43]. mencionado anteriormente.

El comando siguiente genera las migraciones *EF Core*. Estas migraciones crean la base de datos y las tablas en la API .NET Core.

```
dotnet ef migrations add InitialCreate
```

Tras la ejecución del comando anterior se crea el paquete *Migrations* en la solución que contiene dos archivos.

El primero de ellos tiene como nombre *MicroserviceNameContextModelSnapshot.cs*. Esta clase hereda de *ModelSnapshot* y tiene un método llamado *BuildModel()* que se encarga de construir el modelo con las anotaciones definidas, la columna identidad, el tipo de dato de los atributos y las multiplicidades y navegación entre los modelos.

El segundo de ellos tiene como nombre *Fecha_InitialCreate* y consiste en dos métodos, uno de ellos llamado *Up()* que levanta la base de datos con el nombre de las tablas, las claves primarias y las restricciones y otro método llamado *Down()* que realiza el *Drop* de las tablas.

El comando siguiente ejecuta las migraciones *EF Core* y crea la base de datos y las tablas en SQL Server.

```
dotnet ef database update
```

Microservicio de autenticación

Como se ha dicho previamente, todos los microservicios se construyen como se ha explicado en la sección anterior, pero el microservicio de autenticación tiene, además de los explicados anteriormente, el módulo de *Microsoft.AspNetCore.Identity.EntityFrameworkCore* [44]

Este módulo instalado permite gestionar la autenticación fácilmente proporcionando seguridad al sistema, es decir, cifra la contraseña en la base de datos y evita posibles vulnerabilidades.

El siguiente código muestra una extensión de lo que hay que añadir en *Program.cs*.

```
// For Identity
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();

builder.Services.Configure<IdentityOptions>(options =>
{
    // Default Password settings.
    options.Password.RequireDigit = false;
    options.Password.RequireLowercase = false;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = false;
    options.Password.RequiredLength = 6;
    options.Password.RequiredUniqueChars = 1;
});
```

El código de la clase que gestiona el contexto de la base de datos también cambia ligeramente. En este caso, la clase hereda de *IdentityDbContext*.

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace AuthenticationMicroservice.Auth
{
    public class ApplicationDbContext : IdentityDbContext<IdentityUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
        ↪  base(options)
        {
        }
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
        }
    }
}
```

6.2. Implementación del módulo de realidad aumentada

Para la implementación del módulo de realidad aumentada se ha utilizado la librería web de código abierto MindAR comentada anteriormente.

Para realizar el *Image Tracking* con MindAR y poder visualizar el objeto 3D en la portada de los libros hay que preprocesar las imágenes, es decir, se necesita escanear la imagen y extraer puntos característicos para detectar y realizar el seguimiento de la imagen más adelante.

MindAR cuenta para ello una herramienta de compilación llamada *Image Targets Compiler* [45] que es muy intuitiva de utilizar y el proceso de transformar la imagen es relativamente rápido. Lo único que tiene que realizar el usuario es subir la imagen a la plataforma y descargarse el archivo *.mind* generado. Este archivo es el que es necesario para que la librería pueda realizar el seguimiento de la portada.

La Figura 6.2 muestra la portada de un libro sin la compilación.

Figura 6.2: Portada sin preprocesar

La Figura 6.3 muestra la portada compilada y se pueden apreciar los puntos característicos en color rojo.

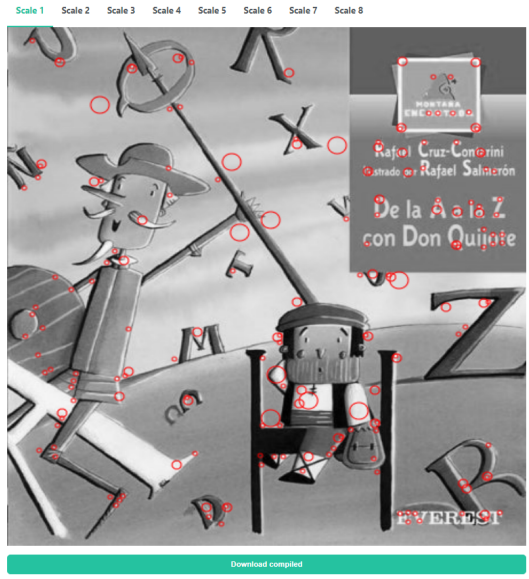

Figura 6.3: Portada preprocesada

El siguiente código muestra los *scripts* necesarios para que el módulo de realidad aumentada funcione correctamente.

```
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <script
      src="https://cdn.jsdelivr.net/gh/hiukim/mind-ar-js@1.1.4/dist/mindar-image.prod
      ↪  .js"></script>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
    <script
      src="https://cdn.jsdelivr.net/gh/hiukim/mind-ar-js@1.1
      ↪  .4/dist/mindar-image-aframe.prod.js"></script>
  </head>
</html>
```

El siguiente código HTML muestra la construcción de la escena. Dentro de la escena hay que incluir el *asset* que se va a mostrar tras escanear la imagen compilada. También se puede configurar varios parámetros del *asset* como la posición, la rotación o las animaciones.

```
<html>
  <body>
    <a-scene mindar-image="imageTargetSrc:
      https://cdn.glitch.global/360d9aee-e22b-4b1d-9b78-bae3866683fe/quijote
  ↪  .mind?v=1647182867558;"
      color-space="sRGB" renderer="colorManagement: true,
      physicallyCorrectLights" vr-mode-ui="enabled: false"
      device-orientation-permission-ui="enabled: false">
      <a-assets>
        <a-asset-item id="quijoteModel"
          src="https://cdn.jsdelivr.net/gh/hiukim/mind-ar-js@1.1
          ↪  .4/examples/image-tracking/assets/card-example/softmind/scene
          ↪  .gltf"></a-asset-item>
      </a-assets>
      <a-camera position="0 0 0" look-controls="enabled: false"></a-camera>
      <a-entity mindar-image-target="targetIndex: 0">
        <a-gltf-model rotation="0 0 0" position="0 0 0.1" scale="0.005 0.005
          0.005" src="#quijoteModel"
          animation="property: position; to: 0 0.1 0.1; dur: 1000; easing:
          easeInOutQuad; loop: true; dir: alternate"></a-gltf-model>
      </a-entity>
    </a-scene>
  </body>
</html>
```

Como muestra de ejemplo del módulo de realidad aumentada se ha realizado el escaneo de los siguientes libros:

- De la A a la Z con Don Quijote.
- El Abecedario de Don Hilario.

6.3. Implementación del *frontend*

El *frontend* de la aplicación se ha construido mediante Ionic, un framework que facilita la calidad del código y la mantenibilidad mediante el patrón arquitectónico MVC.

Para implementar el frontend hay que tener en cuenta que cada componente es una pantalla de la aplicación, y por lo tanto, cuenta con el esqueleto de la pantalla en el fichero HTML, el aspecto visual en el fichero SCSS y la lógica en el fichero TypeScript [46].

Para generar una página hay que ejecutar el siguiente comando. Éste genera la estructura del componente mencionada.

```
ionic g page pages/book-details
```

La carpeta pages contiene las vistas de la aplicación, es decir, lo que se ve en la pantalla. Tras ejecutar el comando, se genera los siguientes archivos en una carpeta llamada según lo escrito en el comando:

- *-routing.module.ts: gestiona el enrutado.
- *.module.ts: contiene el módulo de Angular para una página. Cada página es básicamente su propio módulo con importaciones y estilos.
- *.page.html: contiene la estructura HTML para una página.
- *.page.scss: gestiona el estilo para la página específica.
- *.page.spec.ts: es un archivo de pruebas añadido automáticamente para su página.
- *.page.ts: es el controlador para una página que contiene el código TypeScript que gestiona la funcionalidad.

Para generar un servicio hay que ejecutar el siguiente comando. Éste genera la estructura del servicio.

```
ionic g service services/apiquiz
```

La carpeta *services* contiene los servicios. Se realiza de esta manera para estructurar la aplicación de acuerdo con las mejores prácticas y separar las preocupaciones entre la vista y los datos reales de su aplicación. El servicio se encargará de manejar las llamadas a la API y simplemente devolverá los datos a la vista después.

El código siguiente muestra el código básico de un servicio y de cómo se realiza la petición a la API para obtener los datos.

```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Question } from 'src/shared/app.model';

@Injectable({
  providedIn: 'root'
})
export class ApiquizService {

  private static readonly BASE_URI = 'https://localhost:7075/api/quiz/';

  constructor(private httpClient: HttpClient) { }

  getByBook(bookId : string) {
    let url = ApiquizService.BASE_URI + bookId ;
    console.log(url);
    return this.httpClient.get<Question[]>(url, { observe: 'response' });
  }
}
```

A los métodos establecidos en las clases que actúan como servicio se las llama desde la lógica de cada contenedor, es decir, desde un método de la clase TypeScript de una determina pantalla.

Para la navegación entre pantallas es necesario establecer las rutas de cada componente para poder moverse entre ellos [47].

```typescript
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: 'login',
    loadChildren: () => import('./pages/login/login.module').then(m =>
    ↪  m.LoginPageModule),
  },
  {
    path: 'tabs',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule),
  }
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Existen dos maneras principales de gestionar la navegación entre componentes.

La primera de ellas se realiza con la directiva *routerLink* mientras que la segunda se realiza utilizando *router API*.

Opción 1 - Mediante la directiva *routerLink*

```
<ion-header>
  <ion-toolbar>
    <ion-title>Login</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <ion-button routerLink="/detail">Go to detail</ion-button>
</ion-content>
```

Opción 2 - Mediante *router API*

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  ...
})
export class LoginComponent {

  constructor(private router: Router){}

  navigate(){
    this.router.navigate(['/detail'])
  }
}
```

Capítulo 7

Pruebas

Este capítulo describe las pruebas realizadas para probar el correcto funcionamiento del sistema. Se han realizado pruebas unitarias, pruebas de integración, pruebas de aceptación y pruebas de caja negra.

7.1. Objetivos de las pruebas

La realización de pruebas es un proceso muy importante para el desarrollo de un sistema ya que sirve para comprobar que la implementación del código es correcta y que el sistema se va a comportar de la manera esperada.

El flujo de procesos de las pruebas de un proyecto software incluye varios procesos que sirven para medir la calidad del sistema, que está asociada a los requisitos de sistema [48].

Realizar pruebas de software consta de los siguientes puntos:

- Especificar casos de prueba para verificar el cumplimiento de los requisitos de sistema.
- Ejecutar pruebas de caja negra, pruebas unitarias, pruebas de integración y pruebas de aceptación.
- Asegurar los objetivos de las pruebas de software.

7.2. Tipos de pruebas

Existe una gran variedad de tipos de pruebas que se pueden realizar para probar el funcionamiento del programa. En este proyecto se han realizado pruebas de caja negra, pruebas unitarias, pruebas de integración y pruebas de aceptación.

7.2.1. Pruebas de caja negra

Las pruebas de caja negra son una serie de pruebas de software donde se verifica la funcionalidad tomando en cuenta únicamente la entrada y salida del sistema. La estructura interna o la implementación son opacas en este tipo de pruebas.

Para este tipo de pruebas se utilizan diversas estrategias para obtener el mejor resultado en este proceso. Algunas de las estrategias son:

- Utilización de clases de equivalencia: minimizar el número de casos de test posibles manteniendo la mayor cobertura posible.
- Análisis de valores límite: analizar cuáles son los valores límites de los distintos casos de test y ejecutar el test para estos valores debido a que tiene mayor probabilidad de fallo.
- Tablas de decisión: establece el efecto según la combinación de entradas.

Las Tablas 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 7.10 y 7.11 muestran las distintas pruebas de caja negra realizadas para probar el funcionamiento del sistema.

PCN01	**Identificar a un usuario**
Prerrequisito	El usuario debe estar registrado en el sistema
Datos de entrada	Datos del formulario correctos
Resultado esperado	El usuario se identifica en el sistema
Resultado de la prueba	Correcto

Tabla 7.1: Prueba de caja negra de la identificación de usuario

PCN02	**Registrar a un usuario**
Prerrequisito	El usuario no debe estar registrado en el sistema
Datos de entrada	Datos del formulario correctos
Resultado esperado	El usuario se registra en el sistema
Resultado de la prueba	Correcto

Tabla 7.2: Prueba de caja negra del registro de usuario

PCN03	**Buscar libro**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Título o autor del libro
Resultado esperado	Se muestran los libros que corresponden al nombre establecido como dato de entrada
Resultado de la prueba	Correcto

Tabla 7.3: Prueba de caja negra de la búsqueda de libros

PCN04	**Visualizar detalles de un libro**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema y con un libro concreto seleccionado
Datos de entrada	Libro concreto
Resultado esperado	Se muestra el detalle del libro con su información relevante
Resultado de la prueba	Correcto

Tabla 7.4: Prueba de caja negra de los detalles de un libro

PCN05	**Escanear portada de un libro**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Portada de un libro
Resultado esperado	Se muestra objeto 3D asociado al libro
Resultado de la prueba	Correcto

Tabla 7.5: Prueba de caja negra del escaneo de un libro

PCN06	**Realizar quiz sobre un libro**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema y con un libro concreto seleccionado
Datos de entrada	Libro concreto
Resultado esperado	Se muestra el quiz sobre el libro con preguntas y respuestas
Resultado de la prueba	Correcto

Tabla 7.6: Prueba de caja negra del quiz

PCN07	**Visualizar resultados del quiz sobre un libro**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema y con un libro concreto seleccionado
Datos de entrada	Formulario con las respuestas a las preguntas
Resultado esperado	Se muestra la puntuación del quiz
Resultado de la prueba	Correcto

Tabla 7.7: Prueba de caja negra de la visualización de resultados del quiz

PCN08	**Cerrar sesión**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Sesión iniciada
Resultado esperado	Se cierra la sesión
Resultado de la prueba	Correcto

Tabla 7.8: Prueba de caja negra del cierre de sesión

PCN09	**Visualizar los datos personales**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Sesión iniciada
Resultado esperado	Se muestra los datos del usuario
Resultado de la prueba	Correcto

Tabla 7.9: Prueba de caja negra de la visualización de datos personales

PCN10	**Editar datos personales**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Datos personales
Resultado esperado	Los datos personales se actualizan con los nuevos datos introducidos
Resultado de la prueba	Correcto

Tabla 7.10: Prueba de caja negra de la edición de datos personales

PCN11	**Cerrar sesión**
Prerrequisito	El usuario debe estar con sesión iniciada en el sistema
Datos de entrada	Acción de marcar libro como leído
Resultado esperado	El libro queda marcado como leído y se almacena en los libros leídos del usuario
Resultado de la prueba	Correcto

Tabla 7.11: Prueba de caja negra de marcar libro como leído

7.2.2. Pruebas unitarias

Las pruebas unitarias consisten en probar métodos, clases o módulos de la aplicación de forma aislada para comprobar el correcto funcionamiento de ese subsistema en concreto. Las pruebas unitarias se pueden realizar de forma automatizada mediante integración continua o de forma manual.

Durante el transcurso del proyecto se ha ido realizando pruebas unitarias de forma manual de los distintos subsistemas para su validación:

- Probar el funcionamiento de una petición CRUD del microservicio. Tras implementar la lógica del método o la clase, se realizaban peticiones desde Postman para validar la respuesta.
- Probar el funcionamiento de un componente de Angular para validar su comportamiento frente a determinadas situaciones. Al no estar integrado aún con los datos del microservicio se realizó Mocking Testing para realizar las pruebas con datos falsos.

7.2.3. Pruebas de integración

Las pruebas de integración consisten en verificar que dos o más módulos integrados funcionan de forma correcta en conjunto. Este proceso sería el siguiente paso después de las pruebas unitarias ya que en este punto se conectaría el frontend de la aplicación con el microservicio implementado previamente. Para ello, había que desplegar el microservicio y probar la funcionalidad de la pantalla que se conectaba a él.

7.2.4. Pruebas de aceptación

Las pruebas de aceptación consisten en la verificación total del sistema y comprobar que cumple los requisitos funcionales en su totalidad. Por lo tanto, para ejecutar este tipo de pruebas es necesario el despliegue de toda la aplicación.

Se ha realizado este tipo de pruebas para probar y enseñar a los clientes, es decir, a los tutores, como se comportaría toda la aplicación. Las pruebas de aceptación se han ejecutado de forma manual navegando por las pantallas como lo utilizaría un determinado usuario.

Capítulo 8

Conclusiones

8.1. Conclusiones

Este documento presenta el proceso y resultados del Trabajo Fin de Grado del autor, en el que se ha desarrollado una aplicación por cuenta propia pasando por todas las etapas del ciclo de vida de un proyecto software.

Tras la finalización del mismo, se ha cumplido el objetivo planteado, que consistía en desarrollar una aplicación multiplataforma con diversas formas de aproximación interactiva a la lectura, cumpliendo los aspectos mencionados en la sección 1.3.

En cuanto a un punto de vista técnico se ha podido explorar diversas tecnologías y herramientas como la API de Google Books o el módulo de realidad aumentada con MindAR y su integración con el resto del proyecto. La poca experiencia con las tecnologías utilizadas en el proyecto ha supuesto un desafío extra, pero con una gran recompensa al final.

A nivel personal, lo que más me ha llamado la atención es la conexión entre las asignaturas de la carrera de Ingeniería Informática, las cuales he podido plasmar en un único proyecto. He podido aplicar todos los conocimientos adquiridos durante los cuatro años de carrera mezclando los conceptos y el aprendizaje de las diferentes asignaturas para desarrollar un sistema completo de forma individual por primera vez.

Se podría definir este proyecto como extenso y duro, pero, lo más importante es la satisfacción de ver el proyecto finalizado después de todo el tiempo invertido en él durante el cuatrimestre, valorando todos los aspectos positivos que me ha otorgado tanto a nivel académico como a nivel personal y que permanecerán durante toda la vida. Además, un proyecto de este estilo ha otorgado experiencia para el futuro laboral que recién empieza.

8.2. Líneas de trabajo futuras

Con la finalización de este Trabajo Fin de Grado se ha desarrollado una aplicación móvil multiplataforma con las funcionalidades necesarias que cumplen los requisitos iniciales del proyecto. Al tratarse de un Trabajo de Fin de Grado se han implementado datos suficientes para realizar pruebas y comprobar que el sistema funciona, pero para un sistema real sería necesario introducir más datos en la base de datos como libros o *quiz* para éstos. Con la implementación realizada la escalabilidad de los datos sería fácil de realizar.

Aunque este proyecto cuenta con varias funcionalidades interesantes como el módulo de filtrado de libros, el módulo de realidad aumentada o el módulo de gamificación, es cierto que es posible añadir nuevas funcionalidades como líneas de trabajo futuro personalizando la aplicación existente. Esto supondría obtener una aplicación más completa y con un mayor valor de mercado. Algunas funcionalidades posibles para un trabajo futuro son las siguientes:

1. Implementar un sistema de comunidades de usuarios. Esta funcionalidad consistiría en crear un grupo dentro de la aplicación para gestionar un pequeño grupo de personas. Este sistema podría ser útil para colegios donde las clases tengan su propia comunidad.
2. Implementar un sistema de logros. Esta funcionalidad consistiría en que cada usuario podría disponer de un sistema de logros donde se va premiando al usuario tras lograr ciertos objetivos dentro de la aplicación. Algunos ejemplos de logros podrían ser obtener el escaneo de un determinado número de libros o realizar el *quiz* de diversos libros.
3. Implementar un *ranking*: este sistema consistiría en desarrollar un *ranking* general donde los usuarios escalen posiciones mediante el sistema de logros citado anteriormente o con las puntuaciones de los *quiz* realizados. Si se opta por implementar el sistema de comunidades de usuario también podría crearse un *ranking* de comunidad.
4. Implementar desafíos 1 contra 1: este sistema podría extender del *quiz* implementado para este proyecto, donde además de realizar el *quiz*, también podría realizarse un duelo frente a otro usuario con desafíos por tiempo.
5. Implementar un sistema de recomendaciones de libros: este sistema dispondría de un catálogo de libros donde se obtengan los libros más recomendados para un determinado usuario según sus características. Este sistema podría requerir Inteligencia Artificial para lograr el objetivo. En la actualidad, la mayoría de los sistemas de consumición de contenido digital como *Netflix* o *Amazon Prime Video* tienen implementado esta funcionalidad.
6. Implementar un sistema de reseñas para los libros: este sistema podría consistir en que los usuarios puedan valorar el libro leído otorgando una pequeña reseña para que el resto de los usuarios puedan leerla.

Con todas estas funcionalidades se podría destacar que la aplicación móvil se convertiría en una red social muy completa y novedosa con varias funcionalidades que la harían más entretenida al público.

Bibliografía

[1] ObservatorioHP, "Idea inicial del proyecto." https://hpscds.com/observatorio-hp/observatorio-hp-21-22#LecturApp. Accessed: 2022-02-02.

[2] FGEE, "Compra de libros en 2021 en España." https://www.federacioneditores.org/lectura-y-compra-de-libros-2021.pdf. Accessed: 2022-05-03.

[3] IFEMA, "Beneficios de la lectura tanto en adultos como en niños." https://www.ifema.es/noticias/educacion/beneficios-lectura-adultos-ninos. Accessed: 2022-02-07.

[4] M. Billinghurst, *Augmented reality in education. New horizons for learning.* New Horizons For Learning, 2002.

[5] Danna-Camila Claros-Perdomo, Edwin-Eduardo Millán-Rojas, Adriana-Patricia Gallego-Torres, "Uso de la realidad aumentada, gamificación y m-learning." http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-11292020000100045. Accessed: 2022-02-07.

[6] Economipedia, "Análisis de competencia." https://economipedia.com/definiciones/analisis-de-la-competencia.html5. Accessed: 2022-02-08.

[7] Wattpad.com, "Wattpad." https://play.google.com/store/apps/details?id=wp.wattpad&hl=es&gl=US. Accessed: 2022-02-08.

[8] Amazon Mobile LLC, "Kindle." https://play.google.com/store/apps/details?id=com.amazon.kindle&hl=es&gl=US. Accessed: 2022-02-08.

[9] DEVAR Entertainment LLC, "DEVAR - Realidad Aumentada." https://play.google.com/store/apps/details?id=org.devar.android&hl=es&gl=US. Accessed: 2022-02-08.

[10] ARLOOPA Inc., "ARLOOPA: AR Realidad Aumentada." https://play.google.com/store/apps/details?id=com.arloopa.arloopa&hl=es&gl=US. Accessed: 2022-02-08.

[11] Vidibond S.L., "Flapp - Realidad Aumentada." https://play.google.com/store/apps/details?id=com.vidibond.appvidibond&hl=es&gl=US. Accessed: 2022-02-08.

[12] Ta-tum, "Web de información de Ta-tum." https://ta-tum.com/#welcome. Accessed: 2022-02-08.

[13] César Pablo Gutiérrez y Jesús Vegas Hernández, "Metodología ágil Scrum." https://drive.google.com/file/d/1oA2Q2lqUTIVs8z86rgjpa9kYcA4I8yUQ/view?usp=sharing. Accessed: 2022-03-10.

[14] César Pablo Gutiérrez y Jesús Vegas Hernández, "Marco de referencia Scrum." https://drive.google.com/file/d/1L-3emQimLRdhcNIqye8XdXhWky0MOza6/view?usp=sharing. Accessed: 2022-03-10.

[15] Universidad de Valladolid, "Proyecto docente del trabajo de fin de grado 2021-2022 (Mención Ingeniería de Software)." https://albergueweb1.uva.es/guia_docente/uploads/2021/545/46976/1/Documento.pdf. Accessed: 2022-03-11.

[16] B. Hughes and M. Cotterell, *Software Project Management*. The McGraw-Hill Companies, 2009.

[17] Xiaomi, "Precio Redmi 9T." https://www.mi.com/es/product/redmi-9t/. Accessed: 2022-04-01.

[18] PCExpansion, "Precio portátil Dell." https://www.pcexpansion.es/dell-vostro-3590-2tf9k.php. Accessed: 2022-04-01.

[19] Glassdoor, "Sueldos para el puesto de Programador Júnior en España." https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm. Accessed: 2022-06-19.

[20] MIT, "Licencia MIT." https://choosealicense.com/licenses/mit/. Accessed: 2022-05-14.

[21] Snyk, "What is the MIT License?." https://snyk.io/learn/what-is-mit-license/. Accessed: 2022-05-14.

[22] John Hammersley and John Lees-Miller, "Web oficial de Overleaf." https://es.overleaf.com/. Accessed: 2022-04-19.

[23] Change Vision, "Web oficial de Astah." https://astah.net/. Accessed: 2022-04-19.

[24] Balsamiq Studios, "Web oficial de Balsamiq." https://balsamiq.com/. Accessed: 2022-04-19.

[25] Microsoft, "Documentación oficial de .NET6." https://docs.microsoft.com/es-es/dotnet/core/whats-new/dotnet-6. Accessed: 2022-04-19.

[26] Microsoft, "Características de .NET Core." https://enmilocalfunciona.io/net6/. Accessed: 2022-04-19.

[27] Microsoft, "Página Web de descarga de Microsoft SQL Server." https://www.microsoft.com/es-es/sql-server/sql-server-downloads. Accessed: 2022-04-19.

[28] Intelequia, "What is Microsoft SQL Server and what is it for?." https://intelequia.com/en/blog/post/2948/what-is-microsoft-sql-server-and-what-is-it-for. Accessed: 2022-04-19.

[29] Microsoft, "Documentación oficial de SQL Server Management Studio." https://docs.microsoft.com/es-es/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15. Accessed: 2022-04-19.

[30] Microsoft, "Características de SQL Server Management Studio." https://blog.devart.com/general-review-of-microsoft-sql-server-management-studio-ssms.html. Accessed: 2022-04-19.

[31] Google Inc, "Web oficial de Angular." https://angular.io/. Accessed: 2022-04-19.

[32] Google Inc, "Características principales de Angular." https://angular.io/features. Accessed: 2022-04-19.

[33] Max Lynch and Ben Sperry, "Web oficial de Ionic." https://ionic.io/about. Accessed: 2022-04-19.

[34] JavaTpoint, "Características principales de Ionic." https://www.javatpoint.com/ionic-features. Accessed: 2022-04-19.

[35] MindAR, "Web oficial de MindAR." https://hiukim.github.io/mind-ar-js-doc/. Accessed: 2022-04-19.

[36] Microsoft, "Web oficial de Visual Studio." https://visualstudio.microsoft.com/es/. Accessed: 2022-04-19.

[37] Microsoft, "Comparación de ediciones de Visual Studio." https://visualstudio.microsoft.com/es/vs/compare/. Accessed: 2022-04-19.

[38] Microsoft, "Características de Visual Studo Code." https://code.visualstudio.com/docs/editor/whyvscode. Accessed: 2022-04-19.

[39] RGPD, "Web de RGPD." https://rgpd.es/. Accessed: 2022-05-14.

[40] PEGI, "Web de PEGI." https://pegi.info/es. Accessed: 2022-05-14.

[41] D. Stone, *User Interface Design and Evaluation*. Elsevier, 2005.

[42] Visual Paradigm, "What is Deployment Diagram?." https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/. Accessed: 2022-05-20.

[43] Jason Watmore, ".NET 6.0 - Connect to SQL Server with Entity Framework Core." https://jasonwatmore.com/post/2022/03/18/net-6-connect-to-sql-server-with-entity-framework-core. Accessed: 2022-03-20.

[44] Sarathlal Saseendran, "JWT Authentication And Authorization In .NET 6.0 With Identity Framework." https://www.c-sharpcorner.com/article/jwt-authentication-and-authorization-in-net-6-0-with-identity-framework/. Accessed: 2022-05-01.

[45] MindAR, "Image Targets Compiler." https://hiukim.github.io/mind-ar-js-doc/tools/compile/. Accessed: 2022-06-19.

[46] FreeCodeCamp, "How to Build Your First Ionic 4 App with API Calls." `https://www.freecodecamp.org/news/how-to-build-your-first-ionic-4-app-with-api-calls-f6ea747dc17a/`. Accessed: 2022-02-01.

[47] Ionic Framework, "Angular Navigation." `https://ionicframework.com/docs/angular/navigation`. Accessed: 2022-02-01.

[48] University NTT DATA, "Formación de Testing." Documentación interna de la empresa. Accessed: 2022-05-22.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

Para obtener el código fuente de la aplicación hay que clonar el repositorio alojado en la siguiente URL: `https://gitlab.com/HP-SCDS/Observatorio/2021-2022/lecturapp/uva-lecturapp.git`. En esta URL se alojan los microservicios desarrollados y el sistema basado en Ionic.

Para instalar el *backend* con .NET 6.0 es necesario instalar Visual Studio 2022 con la carga de trabajo de ASP.NET y desarrollo web[1].

Para instalar el *frontend* con Ionic se ha realizado con el editor de código Visual Studio Code. Para descargarlo hay que ir a la página oficial[2]. Aunque cualquier editor de código es válido, se recomienda Visual Studio Code por las extensiones y su fácil configuración.

Una vez instalado el editor de código, se abre el proyecto con él y desde el directorio correspondiente al código de *frontend* se ejecutan los siguientes comandos.

El comando siguiente instala Ionic. Para poder ejecutar este comando es necesario tener instalado Node.js[3].

```
1 npm install -g ionic
```

El comando siguiente sirve para poner en marcha la aplicación.

```
1 ionic serve
```

[1]Más información sobre ASP.NET Core: `https://docs.microsoft.com/es-es/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio`

[2]Más información: `https://code.visualstudio.com/`

[3]Más información sobre Node: `https://nodejs.org/en/`

Si se desea generar el proyecto Ionic para una determinada plataforma, es necesario instalar Capacitor[4].

```
npm install @capacitor/core
npm install @capacitor/cli --save-dev
```

Una vez instalado Capacitor, es necesario ejecutar el siguiente comando[5].

```
ionic capacitor build
```

Con el comando anterior se realizan las siguientes acciones:

- Realizar la construcción de Ionic.
- Copiar *Web Assets* a la plataforma nativa especificada.
- Abrir el IDE para el proyecto nativo: Xcode para iOS o Android Studio para Android.

A.2. Manual de mantenimiento

Para mantener la aplicación, es necesario tener conocimientos de las tecnologías utilizadas en este proyecto y entender el funcionamiento básico, además de haber leído las explicaciones de este documento.

Para añadir nuevos componentes (pantallas) a la aplicación o modificar los ya existentes es necesario haber leído los capítulos 5 y 6 para aprender a gestionar el componente y como se realiza la conexión al servicio para que se realice la consulta al microservicio correspondiente.

Al haber aplicado MVC en el proyecto Ionic y tener la separación correspondiente entre microservicios, la adicción de nuevos componentes consistiría en ejecutar los comandos mencionados en el capítulo 6 para crear un nuevo componente o servicio y crear la lógica y la vista en los archivos correspondientes.

Para añadir nuevos microservicios o modificar los ya existentes es necesario haber leído los capítulos 5 y 6 para gestionar los modelos y el controlador de los microservicios.

Si se desea añadir nuevos datos a la aplicación, hay que añadirlos mediante la base de datos específica según los datos que se quieran añadir. Por ejemplo, si se desea añadir nuevas preguntas para un determinado libro, se debería añadir las líneas deseadas y ejecutar el *script SQL*.

[4]Más información sobre la instalación de Capacitor: `https://capacitorjs.com/docs/getting-started`
[5]Más información sobre Ionic Capacitor: `https://ionicframework.com/docs/cli/commands/capacitor-build`

```
USE [quizdb]
GO
SET IDENTITY_INSERT [dbo].[QuestionItems] ON

INSERT [dbo].[QuestionItems] ([Id], [BookId], [CorrectAnswer], [DescriptionQuestion])
↪  VALUES (1, N'SM_xAAAAMAAJ', N'Quijote', N'¿Qué personaje sale en la letra I?')
INSERT [dbo].[QuestionItems] ([Id], [BookId], [CorrectAnswer], [DescriptionQuestion])
↪  VALUES (2, N'SM_xAAAAMAAJ', N'27', N'¿Cuántas letras tiene el abecedario?')
INSERT [dbo].[QuestionItems] ([Id], [BookId], [CorrectAnswer], [DescriptionQuestion])
↪  VALUES (3, N'SM_xAAAAMAAJ', N'Verde', N'¿De qué color es el gigante de la letra
↪  G?')
INSERT [dbo].[QuestionItems] ([Id], [BookId], [CorrectAnswer], [DescriptionQuestion])
↪  VALUES (4, N'SM_xAAAAMAAJ', N'Blanco', N'¿De qué color es el caballo/letra de Don
↪  Quijote en la portada?')

SET IDENTITY_INSERT [dbo].[QuestionItems] OFF
GO
SET IDENTITY_INSERT [dbo].[AnswerItems] ON

INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (1,
↪  N'Quijote', 1)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (2,
↪  N'Dulcinea', 1)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (3,
↪  N'Sancho Panza', 1)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (4,
↪  N'20', 2)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (5,
↪  N'27', 2)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (6,
↪  N'28', 2)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (7,
↪  N'Azul', 3)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (8,
↪  N'Naranja', 3)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (9,
↪  N'Verde', 3)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (10,
↪  N'Blanco', 4)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (11,
↪  N'Negro', 4)
INSERT [dbo].[AnswerItems] ([Id], [descriptionAnswer], [QuestionQuizId]) VALUES (12,
↪  N'Rojo', 4)
```

Como muestra de ejemplo del módulo de *quiz* se ha realizado las preguntas y respuestas de los siguientes libros:

- De la A a la Z con Don Quijote.
- El Abecedario de Don Hilario.
- Micaela, una rana ridícula.

A.3. Manual de usuario

La Figura A.1 muestra el diagrama de estados de las pantallas de la aplicación y como se navega por las mismas.

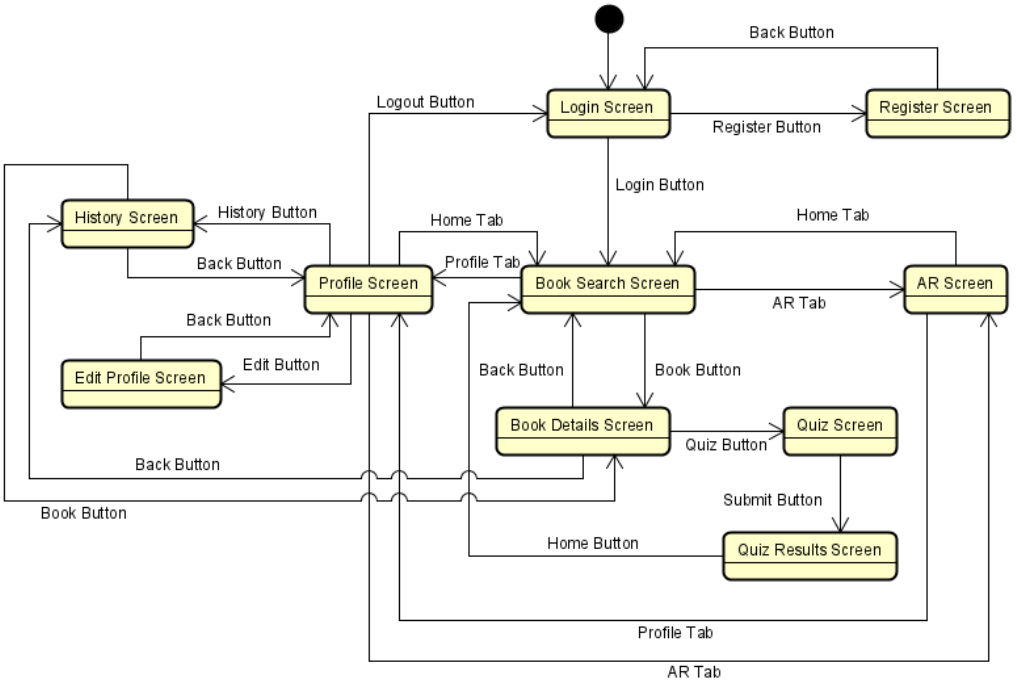

Figura A.1: Diagrama de estados de las pantallas de la aplicación

Pantalla de inicio de sesión

La primera vez que se inicia la aplicación aparece la pantalla de login. Desde esta pantalla se puede iniciar sesión al sistema solo si previamente el usuario ya se había registrado.

Si el usuario no está registrado en la aplicación, hay que ir a la pantalla de registro para introducir los datos personales básicos.

La Figura A.2 muestra la pantalla de inicio de sesión.

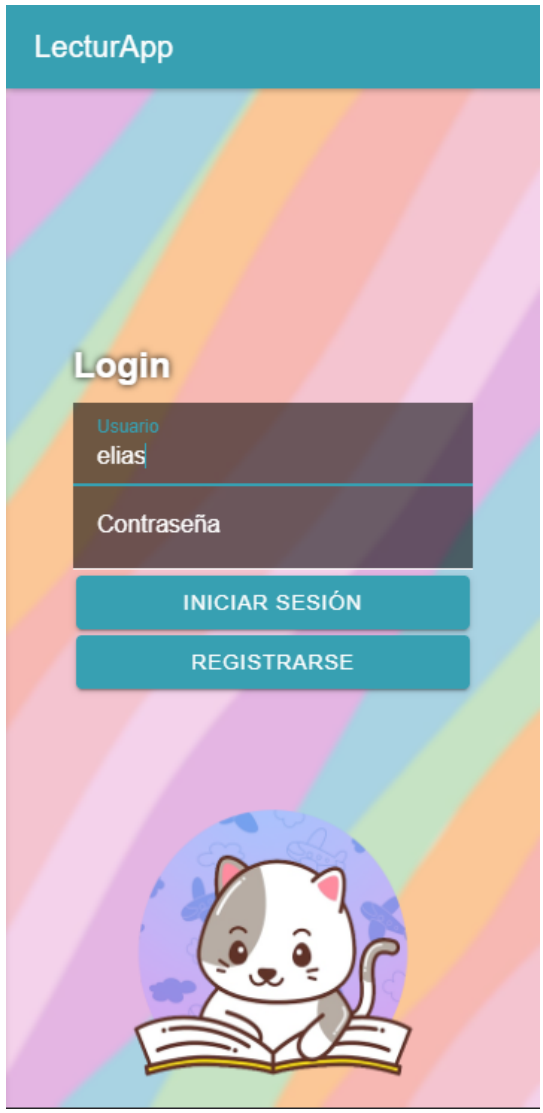

Figura A.2: Pantalla de inicio de sesión

Pantalla de registro

En la pantalla de registro el usuario de la aplicación se registra en el sistema y entra a la pantalla principal.

La Figura A.3 muestra la pantalla de registro.

Figura A.3: Pantalla de registro

Pantalla principal

Desde la pantalla principal se puede navegar por las distintas *tabs* de la aplicación. Cada una de ellas contiene un módulo.

- tab1: gestiona el módulo de búsqueda y filtrado de libros.
- tab2: muestra la cámara para el módulo de realidad aumentada.
- tab3: gestiona el apartado de perfil de usuario y todas las tareas relacionadas con ello.

La Figura A.4 muestra la pantalla de inicio con las *tabs*.

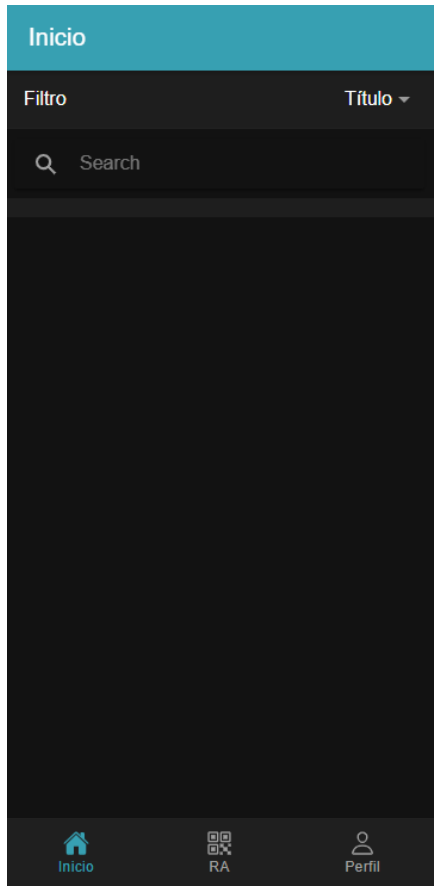

Figura A.4: Pantalla de inicio de sesión

Pantalla de Inicio

En esta pantalla se puede realizar la búsqueda de libros mediante la barra de búsqueda. Se puede filtrar según título y autor. Una vez hecha la búsqueda se puede seleccionar en el libro para obtener más detalles sobre el mismo.

La Figura A.5 muestra la pantalla de búsqueda de libros.

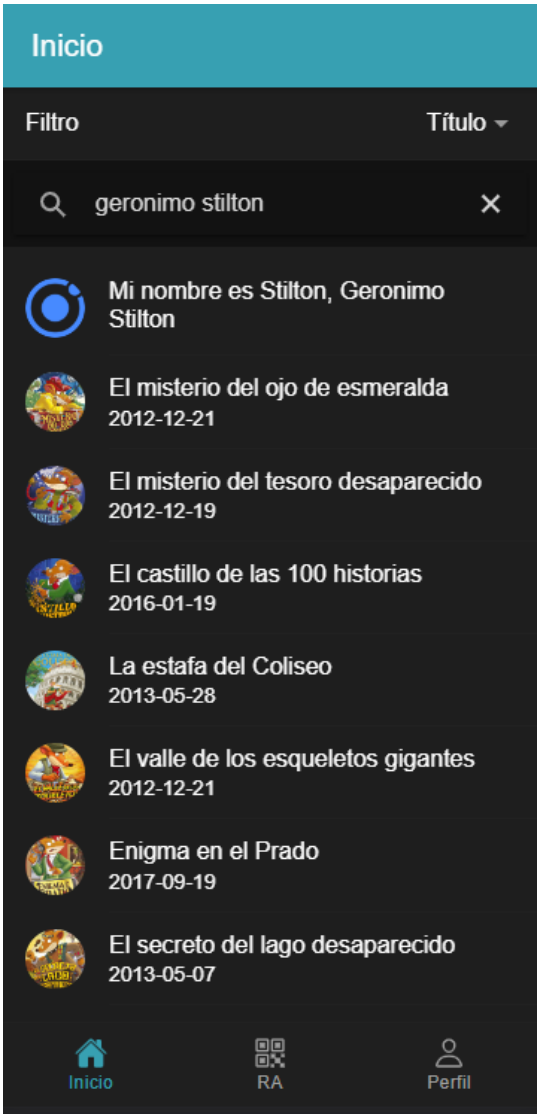

Figura A.5: Pantalla de búsqueda

Pantalla de detalles del libro

En esta pantalla aparece información sobre el libro seleccionado. Esta información es el título, autor, imagen de la portada, categoría del libro, valoración del libro y descripción.

Además, se puede añadir un libro a la lista de "Libros leídos" mediante el botón *toggle*.

La Figura A.6 muestra la pantalla de detalles de un libro.

Figura A.6: Pantalla de detalles del libro

Pantalla de *quiz*

Al marcar la opción "Marcar libro como leído" se hace visible un botón para acceder al *quiz* del libro.

En esta pantalla se realiza el *quiz* seleccionando las respuestas que el usuario considere correctas. Al terminar se pulsaría el botón y se cambiaría a la pantalla de visualización de resultados.

La Figura A.7 muestra la pantalla de *quiz*.

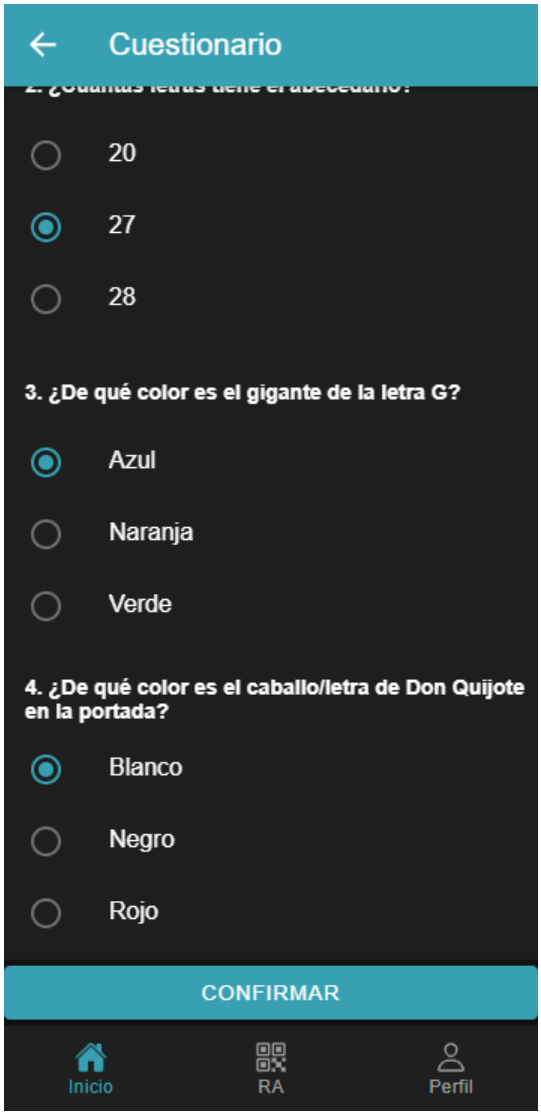

Figura A.7: Pantalla de *quiz*

Pantalla de visualización de resultados del *quiz*

En esta pantalla aparece el porcentaje de aciertos del *quiz* realizado. Una vez terminada la visualización del resultado se pulsaría al botón para volver al menú de inicio.

La Figura A.8 muestra la pantalla de visualización de los resultados del *quiz*.

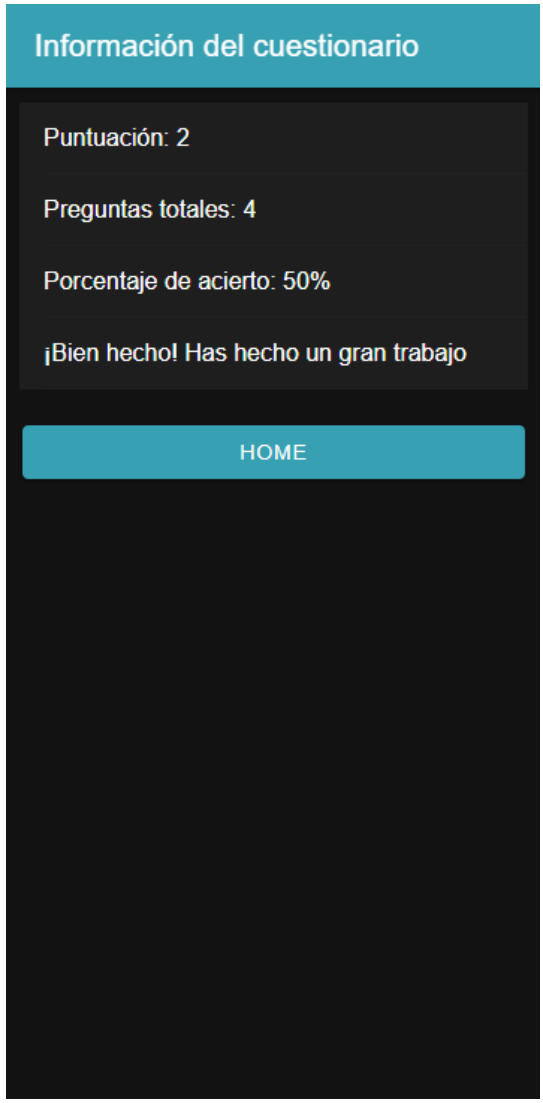

Figura A.8: Pantalla de visualización de resultados del *quiz*

Pantalla de realidad aumentada

En esta pantalla se muestra la cámara del móvil para el escaneo de la portada del libro correspondiente.

La Figura A.9 muestra la pantalla de realidad aumentada.

Figura A.9: Pantalla de realidad aumentada

Pantalla de perfil de usuario

En esta pantalla el usuario puede visualizar la información personal como *nickname*, correo, edad o libro preferido.

Además, se puede acceder a la pantalla de edición de datos personales del usuario o a la lista de libros leídos pulsando en los botones correspondientes.

La Figura A.10 muestra la pantalla de perfil de usuario.

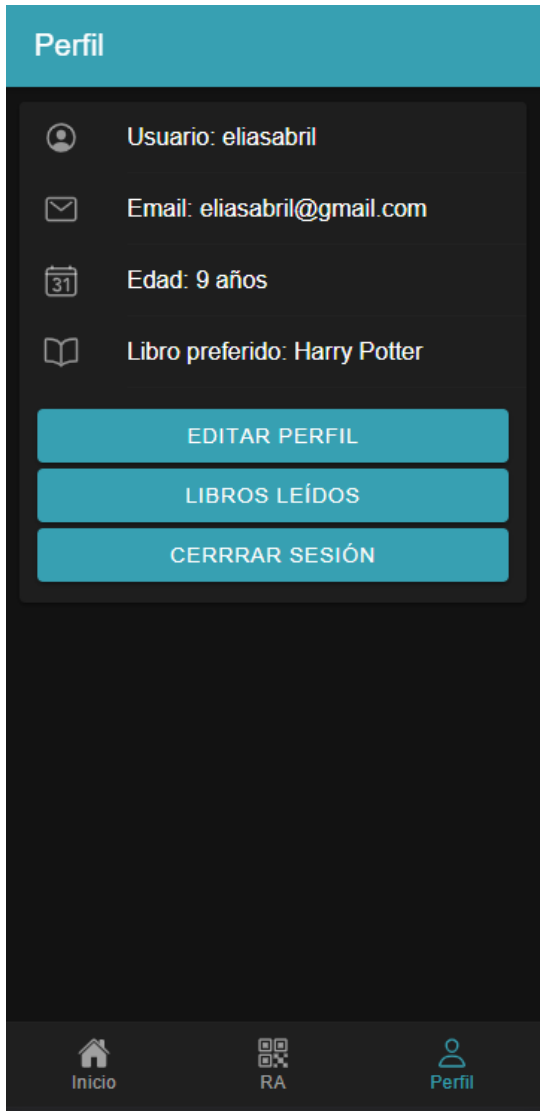

Figura A.10: Pantalla del perfil de usuario

Pantalla de edición de datos personales

En esta pantalla se puede editar los datos del usuario.

La Figura A.11 muestra la pantalla de edición de datos personales.

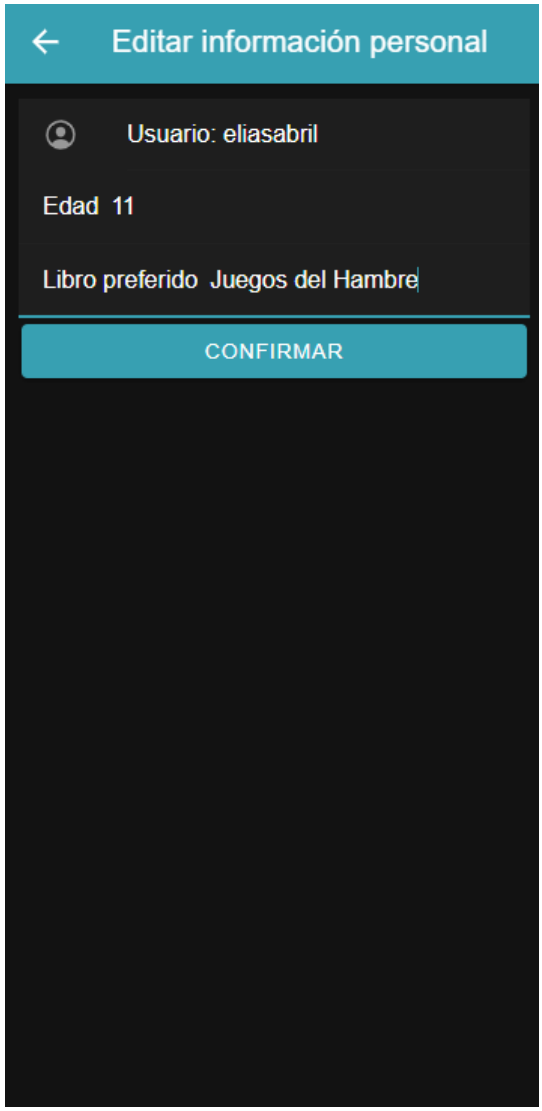

Figura A.11: Pantalla de editar datos personales

Pantalla de lista de libros leídos

En esta pantalla se puede visualizar la lista de los libros marcados como leídos en la pantalla de detalles del libro.

La Figura A.12 muestra la pantalla de libros leídos.

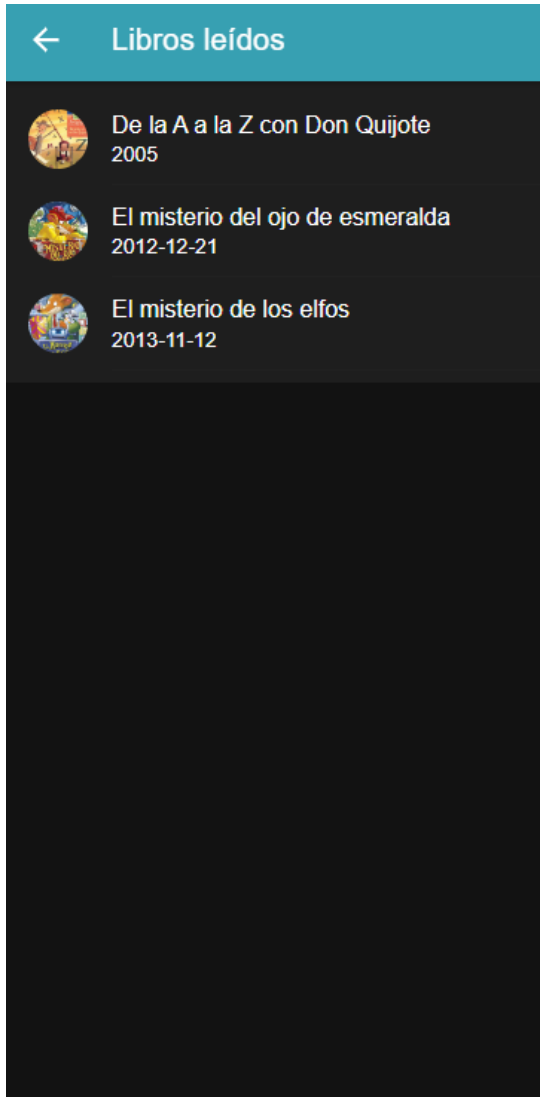

Figura A.12: Pantalla de libros leídos

Apéndice B

Resumen de enlaces adicionales

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

- Repositorio Gitlab del código: `https://gitlab.com/HP-SCDS/Observatorio/2021-2022/lecturapp/uva-lecturapp.git`.
- Demo de la aplicación: `https://drive.google.com/file/d/1XNT5J7_v-QKCIF7kKnMZiMx7DsaeEX0S/view?usp=sharing`