Desarrollo de una aplicación para la búsqueda y compra de libros mediante el reconocimiento de imágenes

Trabajo de Fin de Máster
Curso 2019–2020

Autor
Iván Monterrubio Cerezo

Director
Antonio Sarasa Cabezuelo

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Desarrollo de una aplicación para la búsqueda y compra de libros mediante el reconocimiento de imágenes

Trabajo de Fin de Máster en Ingeniería Informática

Autor
Iván Monterrubio Cerezo

Director
Antonio Sarasa Cabezuelo

Convocatoria: *Julio 2020*
Calificación: *8 - Notable*

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Dedicatoria

A mis padres, por hacerme ver, desde niño, que quería ser ingeniero

Agradecimientos

El Trabajo de Fin de Máster no ha sido la finalización de unos estudios superiores. Ha sido mucho más. Ha sido un éxito, un logro, alcanzar el objetivo que comenzó muchos años atrás. Es la culminación del esfuerzo, el sacrificio que hay que sufrir para llegar a ser la persona en quien me quería convertir cuando miraba hacia delante. Y es el orgullo de obtener lo que siempre he querido y que tanto ha costado.

Mis estudios universitarios han sido para mí una autopista sin áreas de descanso. Comencé el Grado en Ingeniería Informática y cuando fui consciente de que me había graduado ya estaba cursando el Máster en Ingeniería Informática. Hoy termina todo ello, de una manera tan precipitada como empezó. Por ello, esta sección habla de mucho más que del Trabajo de Fin de Máster, habla del fin de un etapa, la que me ha llevado a convertirme en lo que soy ahora, un ingeniero informático orgulloso de lo que ha conseguido, exhausto por el esfuerzo que ha llevado, e ilusionado por ver adónde me lleva el camino que he elegido.

Por lo tanto, el fin de una gran etapa como ha sido esta, va unido a un enorme agradecimiento. Quiero dar las gracias a todas las personas que en algún momento me han ofrecido su ayuda, su apoyo, han demostrado interés, o simplemente se han preocupado por cómo me iban las cosas. Afortunadamente, estas personas son demasiadas para nombrarlas a todas. En primer lugar, quiero dar las gracias a mis profesores, a todos ellos que realmente se interesan porque sus alumnos aprendan, que hacen que crezca esa emoción por seguir estudiando, aprendiendo y disfrutando con lo que se hace. Gracias por animar a vuestros alumnos a seguir adquiriendo conocimiento. Y en particular, quiero agradecer a Antonio Sarasa, director de este Trabajo de Fin de Máster, su ayuda, y sobre todo, que haya vuelto a depositar su confianza en mí para la realización de este proyecto tras haber realizado juntos el Trabajo de Fin de Grado.

Quiero dar las gracias a mis compañeros, por ayudarme a superar aquellas asignaturas que tentaban con resistirse y por hacer de las clases momentos inolvidables. Quiero hacer una mención muy especial a Carlos Gavidia y David Gorricho. Habéis conseguido que estos seis años hayan sido muchísimo más fáciles, se hubiesen hecho mucho más cuesta arriba sin vosotros. Siento decir que este Trabajo de Fin de Máster no ha tenido nada que ver con el Trabajo de Fin de Grado que hicimos juntos. Haría mil TFG's más con vosotros. Muchísimas gracias.

Lejos del ámbito académico, y aun sin tener ninguna relación con mis estudios,

para la superación de estos dos títulos han tenido un papel fundamental mis amigos, los de siempre. El apoyo y desahogo que me han proporcionado ha tenido más valor que cualquiera de las academias que hubiera podido visitar. Todo era más sencillo sabiendo que cuando cerrara el ordenador, me estarían esperando en cualquier parte para hacer cualquier cosa. Los planes, las fiestas o las charlas que nunca faltan entre nosotros me hacían liberarme de la presión que muchas veces sentía. Y que me preguntasen, aunque fuera mediante un simple mensaje, que qué tal me iba, significaba mucho para mí. Juan Carlos, Raúl, Abel, Juanjo, Iván, Ángel, Víctor, Patxy, Mario, mis chicos del 122, gracias por ser la familia que somos.

La persona de quien voy a hablar a continuación no se merece un gracias, se lo merece todo. Porque ha sido mi mayor apoyo durante este largo periodo, por ser quien mejor me entiende siempre y porque lo es todo. Gloria, gracias por haber estado siempre a mi lado, por soportarme cuando ni yo mismo lo hacía, por darme todo tu apoyo, tu consejo, por tu interés y preocupación constantes por mí, por hacer cualquier cosa que estuviera en tu mano por ayudarme, por hacer que me sintiera mejor, o simplemente por sacarme una sonrisa, aunque contigo no resulte para nada difícil. Siempre has sido mi modelo a seguir, en quien me fijaba cuando quería hacer las cosas bien, quien me indicaba cuál era el camino cuando circulaba a ciegas por esta amplia autopista. Cuando me faltaba motivación, tú me la dabas. Cuando me faltaban las fuerzas, me entregabas las tuyas. Solo puedo darte las gracias, gracias por ser como eres. Gracias por todo.

Como se suele decir, por último y no por ello menos importante, me queda dar las gracias a mi familia. Desde que tengo memoria, recuerdo que mi madre me decía: "¿Tú de mayor quieres tener un Ferrari? Pues tienes que estudiar mucho y tienes que hacerte ingeniero". No sé si en algún momento llegará ese Ferrari, pero desde pequeño me habéis inculcado que los estudios son lo más importante, que tenía que esforzarme si quería tener una vida cómoda, y que todo sacrificio tiene su recompensa. No os cansabais de decirme que vosotros teníais vuestro trabajo y el mío era estudiar. Me habéis enseñado a ser una persona coherente, responsable, disciplinada y trabajadora. Gracias por darme las fuerzas y la educación necesaria para lograr todo lo que me he propuesto. Todo lo que he conseguido, sin ninguna duda, es gracias a vosotros. Y no me puedo olvidar de mi hermana, que especialmente durante este Trabajo de Fin de Máster ha tenido que soportarme y me ha ayudado a desarrollar la aplicación sobre todo en la parte de la interfaz y experiencia de usuario. Pero también durante el grado ha tenido que aguantar mis quejas y ha sido un grandísimo apoyo dentro de casa, interesándose por cómo lo llevaba y de qué iban esas prácticas en las que ella solo veía un montón de palabras sin sentido en la pantalla. Muchas gracias por vivirlo conmigo.

Gracias a todos por haberme ayudado a superar mis estudios, y en particular, este proyecto.

Mamá, ya soy ingeniero.

Resumen

Desarrollo de una aplicación para la búsqueda y compra de libros mediante el reconocimiento de imágenes

Este proyecto constituye el Trabajo de Fin de Máster de Iván Monterrubio Cerezo, alumno del Máster en Ingeniería Informática de la Universidad Complutense de Madrid.

El objetivo del proyecto es ofrecer una herramienta que posibilite encontrar de manera sencilla, rápida e intuitiva las diferentes opciones con las que cuentan los amantes de la lectura a la hora de comprar un libro. Se trata de ofrecer accesibilidad a cualquier título que se desee adquirir o consultar, extendiendo el concepto de buscador de libros y aportándole un valor añadido.

El resultado principal de este proyecto ha sido el desarrollo de una aplicación móvil multiplataforma que proporciona información detallada de libros, los cuales pueden buscarse mediante la introducción manual de sus datos o a través de una foto de los mismos. Para ellos, además de datos técnicos, se ofrece una vista previa, enlaces de compra en distintas plataformas o incluso la posibilidad de leerlos completos o descargarlos en formato digital.

La aplicación hace uso de Google Books, una API de Google que ofrece acceso a una extensa biblioteca digital, para la búsqueda de los libros, así como otros servicios de automatización de búsquedas para la consulta y comparación de diversas plataformas digitales de compra.

Palabras clave

Libros, Compra, Reconocimiento de imágenes, Aplicación móvil, API, Google Books

Abstract

Development of an application for search and purchase of books through image recognition

This project is the Final Master's Project of Iván Monterrubio Cerezo, student of the Master in Computer Engineering at the Universidad Complutense de Madrid.

The objective of the project is to offer a tool that makes it easy, quickly and intuitively to find the different options available to lovers of reading when buying a book. It is about offering accessibility to any title that you want to acquire or consult, extending the concept of book search and providing an added value.

The main result of this project has been the development of a multiplatform mobile application that provides detailed information on books, which can be searched for by manually entering their data or by taking a photo of them. For them, in addition to technical data, it offers a preview, purchase links on different platforms or even the possibility of reading them fully or downloading them in digital format.

The application uses Google Books, a Google API that offers access to an extensive digital library to search for books, as well as other search automation services to query and compare various digital purchasing platforms.

Keywords

Books, Purchase, Image recognition, Mobile app, API, Google Books

Índice

Índice de figuras

Capítulo 1

Introducción

Hace ya varios años, el mundo está pasando por un proceso de revolución tecnológica que repercute en prácticamente todos los ámbitos de la sociedad. Hay incluso quienes afirman que se está produciendo la cuarta revolución industrial de nuestra historia, la que se denomina Industria 4.0. Esto está provocando un cambio sustancial a nivel empresarial, por el que la gran mayoría de empresas están digitalizando su negocio, pasando de una metodología manual a una informática, haciendo que prácticamente todos los procesos pasen por un sistema informático o estén sujetos al uso de internet. Un gran ejemplo es el negocio cinematográfico. Hace no muchos años, para ver una película, había que ir a un videoclub y alquilarla por unos días, mientras que ahora existen plataformas digitales en las que se puede acceder instantáneamente a casi cualquier título. Esto ha provocado que el cine en formato físico haya disminuido hasta casi desparecer.

En el mundo de la lectura ha ocurrido algo similar. Las bibliotecas y librerías era el lugar habitual al que se dirigían los amantes de la lectura para informarse de las últimas novelas publicadas o comprar libros. Actualmente, para estar al tanto de las novedades literarias, basta con hacer una búsqueda en internet para encontrar toda la información que se necesite. A la hora de decidirse por adquirir una obra, se pueden encontrar muchas plataformas digitales donde se venden libros en todos sus formatos, ya sea físico, de tapa dura, de tapa blanda, edición de bolsillo o digital, pero será necesario bucear entre las diferentes tiendas para encontrar la mejor opción, siendo en ocasiones una tarea algo engorrosa. LiBook, la aplicación desarrollada durante este proyecto, facilita en gran medida la comparación de las diferentes opciones con las que cuenta un usuario para comprar un libro, de la misma manera que hace mucho más sencillo encontrar el título en concreto del que se tiene interés. Al estar especializada únicamente en libros, no existe el riesgo de confundir una obra con una película que lleva el mismo nombre, o consultar un libro erróneo por accidente. Su interfaz visual, así como la búsqueda de libros mediante una imagen, dejan muy claro si el libro que se consulta es el que realmente se está buscando.

LiBook permite fotografiar la portada de un libro y automáticamente se muestra toda la información relativa a esa obra, así como una vista previa de la misma, enlaces a las principales tiendas digitales e incluso la posibilidad de leerlo por completo de manera online o descargarlo en formato *PDF* o *EPUB* para leerlos en un libro

electrónico, además de otras funciones como añadir títulos a una lista de favoritos.

1.1. Motivación

La elección de este tema para el desarrollo del Trabajo de Fin de Máster viene muy marcado por ser la lectura una de las aficiones más comunes. Internet da acceso a toda la información que se necesite, también el mundo de la literatura, pero en ocasiones puede que haya demasiada información. Resulta extremadamente útil cuando se quiere conocer, por ejemplo, cuáles son las últimas tendencias, las novedades de un escritor, o cuándo estrenarán en el cine la adaptación cinematográfica de una obra. Pero si se quiere encontrar un sitio dedicado exclusivamente a libros en el que se pueda leer una muestra o encontrar de un vistazo los lugares donde se puede comprar, puede no resultar tan sencillo, o en ocasiones, es necesario leer varios párrafos de información que no resulta relevante hasta llegar a lo que realmente se está buscando.

En lo relativo al desarrollo del proyecto, se ha decidido realizar una aplicación móvil tanto para dispositivos Android como iOS con la intención de llegar al mayor número de usuarios posibles. Las aplicaciones móviles son probablemente la herramienta más utilizada por gran parte de la población para realizar casi cualquier acción, por lo que, en general, se está muy familiarizado con este tipo de programas.

Por otra parte, el uso de las API se extiende más cada día, debido a que es una forma sencilla y eficaz de aprovechar servicios y funciones ya desarrolladas y probadas por otros desarrolladores, por lo que se ha decidido usar varias API que posibilitan acceder a la información de una extensa colección de libros y ofrecer la funcionalidad de llegar hasta esos títulos mediante el reconocimiento de texto en imágenes.

1.2. Objetivos

El principal objetivo de LiBook es ofrecer un servicio de búsqueda de libros sencillo, rápido e intuitivo que proporcione la información más relevante de ellos y ofrezca un valor añadido a los buscadores tradicionales. Los objetivos del proyecto se pueden concretar en los siguientes:

- Proporcionar un sencillo buscador de libros basado en *título* y *autor*.
- Ofrecer un buscador tomando como entrada la imagen de la portada del libro.
- Mostrar una lista con los posibles resultados para escoger el libro exacto al que el usuario se refiere.
- Consultar información relevante del libro en un solo vistazo, como el *título*, *autor*, *descripción*, *fecha de publicación*, *editorial*, *número de páginas* o su *ISBN*.
- Ofrecer la posibilidad de leer una vista previa o la totalidad del libro online.
- Descargar el libro en formato digital.

- Ver enlaces de compra del libro de los vendedores más habituales.
- Disponer de una sección de *favoritos*.

1.3. Plan de trabajo

Para la realización del proyecto se ha seguido la siguiente planificación:

- Búsqueda de información sobre el estado del arte: se comenzó haciendo una investigación sobre aplicaciones o servicios similares al de este proyecto, como apps especializadas en libros o comparadores de precios. Después, consultando diferentes maneras de acceder a colecciones de libros o alguna API que ofreciera información sobre los mismos, así como otras API que permitieran reconocer texto en imágenes.
- Especificación de requisitos: en esta fase se detallaron cuáles iban a ser todas las funcionalidades del proyecto y cómo iban a llevarse a cabo.
- Elección de API para varias funcionalidades: en este punto se evaluaron varias de las API encontradas durante el proceso de investigación del estado del arte y se determinaron cuáles eran las mejores opciones para las funcionalidades de búsqueda de libros y reconocimiento de texto en imágenes.
- Diseño y desarrollo de una aplicación móvil: durante esta fase se realizó una exploración de las diferentes opciones para la creación de una aplicación móvil multiplataforma y se llevó a cabo su desarrollo.
- Evaluación y pruebas de la aplicación: finalmente, el software resultante fue sometido a pruebas para la corrección de errores y comprobar el correcto funcionamiento de la aplicación.

En la figura 1.1 se puede apreciar un cronograma con la planificación de las diferentes actividades llevadas a cabo para el desarrollo del proyecto.

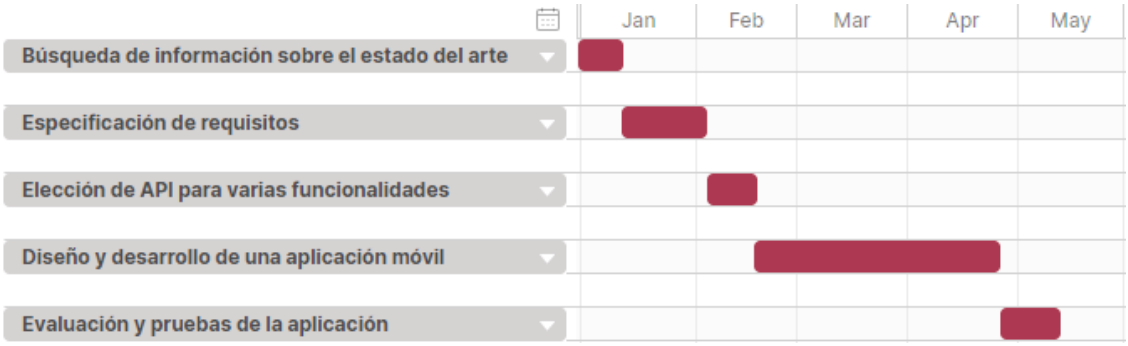


Figura 1.1: Planificación temporal

Capítulo 2

Estado del Arte

Este capítulo realiza una revisión del estado del arte en relación a los puntos principales de los que consta el proyecto y su desarrollo. Particularmente, se explicará qué es una API y cuán importantes son hoy en día, se hablará sobre el procesamiento de imágenes y el reconocimiento de texto en ellas, se mostrarán aplicaciones cuyo tema principal es la búsqueda de libros, tal y como se hace en este proyecto, y por último, se verán varios proyectos y aplicaciones que, al igual que esta aplicación, comparan las diferentes opciones que tiene un usuario al interesarse en la compra de un producto, ofreciendo varios vendedores y posibilidades.

2.1. API

API son las siglas de Interfaz de Programación de Aplicaciones. La definición de API es un conjunto de funciones y especificaciones que ejecutan diversos procedimientos y tienen como finalidad ser utilizadas por software externo. Una API permite implementar funciones externas que se emplean en nuestro proyecto sin la necesidad de programarlas nuevamente. En terminología de programación, es una capa de abstracción. Su gran atractivo consiste en poder hacer uso de funciones o la infraestructura ya implementadas en otro software reutilizando de esta manera código que tiene la garantía de estar probado y de que funciona correctamente.

Actualmente, la revolución de los servicios web ha hecho evolucionar el término API al de API REST, aunque comúnmente se le sigue conociendo únicamente como API.

REST es un estilo de arquitectura de software que se utiliza para describir cualquier interfaz entre sistemas que utilicen el protocolo HTTP para comunicarse. Sus siglas significan Transferencia de estado representacional. Esta interfaz se utiliza para obtener o generar datos en todos los formatos posibles, siendo los más comunes XML o JSON. Se trata de una alternativa sencilla a otros protocolos como SOAP. Algunas de las características principales de REST son las siguientes:

- **Protocolo cliente/servidor**: el uso de REST y de una API REST se basa en la conexión que establecen el cliente y el servidor. El cliente no necesita

saber cuáles son los detalles de la implementación del servidor, y este no tiene en cuenta cómo serán utilizados los datos que devuelva, el cliente se encarga únicamente de enviar o solicitar datos y el servidor le devuelve la respuesta según las especificaciones de la API. Por lo tanto, se dice que el cliente y el servidor están débilmente acoplados.

- **Sin estado**: cada una de las peticiones HTTP contienen toda la información necesaria para que sea ejecutada, por lo que ni el cliente ni el servidor precisan recordar estados previos para procesarla, como un inicio de sesión.
- **Operaciones**: las principales operaciones que incorpora un sistema REST y la especificación HTTP son GET (consultar), POST (crear), PUT (editar) y DELETE (eliminar), por lo que su uso se simplifica considerablemente.
- **URI**: En REST, los objetos siempre se manipulan a través de la URI. Este es el único identificador único de cada recurso, lo que facilita acceder a la información para ejecutar cualquiera de las operaciones permitidas.

Por lo tanto, se puede definir una API REST como una biblioteca apoyada completamente por el protocolo HTTP, lo que ofrece un servicio que provee de funciones que dan la capacidad de utilizar servicios web ajenos a nuestro proyecto dentro de una aplicación propia de manera segura. Esto nos ofrece algunas ventajas notables:

- **Separación entre el cliente y el servidor**: el protocolo REST divide totalmente la interfaz de usuario del servidor y el almacenamiento de los datos. Esto mejora la portabilidad a otras plataformas, incrementa la escalabilidad de los proyectos y hace posible que los diferentes componentes del desarrollo evolucionen independientemente.
- **Visibilidad, fiabilidad y escalabilidad**: esta separación entre el cliente y el servidor tiene la ventaja de que los diferentes equipos de desarrollo puedan escalar el producto sin demasiados problemas. Se puede migrar a otros servidores o hacer cualquier tipo de cambio en la base de datos siempre que el envío de los datos sea el correcto. Esta división facilita que diferentes servidores se encarguen del back y del front, haciendo que las aplicaciones sean más flexibles.
- **Independencia de plataformas y lenguajes**: La API REST es independiente de la plataforma o el lenguaje que se utilice, adaptándose al tipo de sintaxis o las plataformas con las que se esté trabajando, ofreciendo libertad para cambiar o probar nuevos entornos dentro del desarrollo. Es indiferente si el servidor de la API REST es PHP, Python, Node.js o Java, lo único necesario es que las respuestas que ofrece la API sean en el lenguaje de datos que se especifica, generalmente JSON o XML.

Hoy en día, las API es uno de los recursos más importantes para las empresas, siendo una de las mayores fuentes de ingresos actualmente. El Instituto Apigee realizó una encuesta[(1)] con la que llegó a la conclusión de que aquellas organizaciones identificadas como más competentes en el uso de las API obtenían mayores ingresos

y producían mayores niveles de satisfacción del cliente. La gran mayoría de los ejecutivos que fueron encuestados (en el ámbito de las TIC y el marketing) coincidían en que las API producirían un notable impacto en sus negocios en los 12 meses siguientes, y que podría aumentar con el tiempo.

Algunos ejemplos destacables de organizaciones que tienen a disposición de sus clientes servicios API REST son: Google, con su extensa plataforma Google Cloud Platform[(2)], en la que se pueden solicitar acceso a numerosas API de pago o gratuitas con temáticas tan variadas como la computación, el almacenamiento y bases de datos, análisis de datos, aprendizaje automático, herramientas de administración o infraestructura administrada; Twitter[(3)], que proporciona una variedad de servicios para consultar, crear, administrar o analizar sus datos, así como la creación de bots y otras herramientas que facilitan el acceso y el análisis de la información de mayor actualidad; o Amazon[(4)], que entre sus servicios está el acceso a diversas API que ofrecen herramientas de desarrollo, monetización, testing o incluso API Gateway[(5)], un servicio completamente administrado que facilita a los desarrolladores la creación, la publicación, el mantenimiento, la monitorización y la protección de API a cualquier escala.

2.2. Reconocimiento de texto en imágenes

Una de las muchísimas funcionalidades que ofrece la gran expansión tecnológica en la que vivimos es la capacidad de reconocer texto en una imagen. Cada vez es más frecuente el uso de programas en los que se hace un análisis de una imagen y se obtienen de ella datos de gran interés, como sus características cromáticas, la relación entre una imagen y datos geográficos, saber qué objetos se encuentran en ella, conocer el contexto que se percibe en la imagen, y por supuesto, descubrir si en la imagen aparece texto y saber cuál es, dónde está o qué significa.

El funcionamiento de este tipo de funcionalidades viene muy marcado por el uso del aprendizaje automático, que ha adquirido un papel fundamental en los procesos de reconocimiento de imágenes. El proceso por el cual se consigue que un programa sea capaz de reconocer elementos en una imagen se caracteriza principalmente por dotar a un algoritmo la capacidad de aprender a detectar cuándo la imagen contiene elementos destacables. Esto se consigue haciendo que un sistema pase por un proceso de aprendizaje: en primer lugar, se le muestran una gran cantidad de imágenes en los que se encuentran presentes los elementos que queremos que reconozca, ya sean objetos, colores o letras, indicándole cuáles son y dónde se encuentran. El algoritmo del sistema aprende a reconocer cuáles son esos elementos, y posteriormente, es capaz de detectarlos en imágenes nuevas.

Muchos son los servicios que cuentan con esta función, ofreciéndolos a los usuarios generalmente a través de una API. Las organizaciones que hacen de esta funcionalidad un modelo de negocio proveen a los usuarios de los mecanismos necesarios para enviarles una imagen y que reciban fácilmente los resultados que desean, aunque también existen librerías de código abierto que ofrecen esta posibilidad.

Algunos ejemplos de servicios que cuentan con reconocimiento de texto en imágenes son los siguientes:

- **API Vision**: la API Vision[6] de Google Cloud ofrece modelos de aprendizaje automático preparados previamente y muy potentes a través de las API REST y RPC. Asigna etiquetas a imágenes y las clasifica rápidamente en millones de categorías predefinidas. Detecta objetos y caras, lee texto impreso y manuscrito, y consigue metadatos de gran valor.
- **Computer Vision**: Computer Vision[7] es un conjunto de API de Microsoft que proporciona varios servicios que detectan y extraen texto manuscrito o impreso que aparece en las imágenes. Cuenta con hasta tres API que se adaptan a la necesidad del usuario: Read API, que detecta el contenido de texto de una imagen y convierte el texto identificado en una secuencia de caracteres legible por una máquina, estando optimizada para imágenes con gran cantidad de texto; OCR API, API de reconocimiento óptico de caracteres (OCR) de Computer Vision similar a Read API, pero que se ejecuta de forma sincrónica y no está optimizada para documentos de gran tamaño; y Recognize Text API, similar a OCR, pero que se ejecuta de forma asincrónica y usa modelos de reconocimiento actualizados.
- **Amazon Rekognition**: la API de Amazon Rekognition[8] facilita la adición de análisis de imagen y video con tecnología probada, altamente escalable y de aprendizaje profundo que no requiere experiencia en aprendizaje automático para su uso. Con Amazon Rekognition se pueden identificar objetos, personas, texto, escenas y actividades en imágenes y videos, además de detectar cualquier contenido inapropiado. Amazon Rekognition también proporciona análisis faciales de alta precisión y capacidades de búsqueda facial que se pueden usar para detectar, analizar y comparar rostros.
- **Tesseract**: Tesseract[9] se considera el OCR open source más preciso y está disponible para descarga en su repositorio de Github. Está entrenado en más de 100 idiomas, soporta varios formatos de imagen, lee documentos de varias páginas y funciona en GNU/Linux, Windows y macOS. Aunque carece de una interfaz gráfica nativa, se puede integrar fácilmente con ECM como OpenKM, Alfresco o Nuxeo y existen múltiples aplicaciones tanto de escritorio como en línea que usan Tesseract como back-end.

2.3. Proyectos similares

Tras un proceso de investigación, se han encontrado aplicaciones que realizan tareas similares a las presentadas en este proyecto. A continuación se muestran algunas de ellas, y serán divididas en dos ámbitos, el reconocimiento automático y búsqueda de libros, y comparadores de precios para diversos productos.

2.3.1. Búsqueda de libros

- **Quiero Libros**: Quiero Libros es una aplicación de compra-venta de libros físicos. Dispone de una aplicación web[10] y de una aplicación para dispositivos Android[11]. Su propósito es ofrecer un mercado especializado exclusivamente en libros tanto nuevos como de segunda mano. Entre sus funcionalidades está

la de añadir libros manualmente y en el caso de la aplicación móvil ofrece la posibilidad de escanear el código de barras o el código ISBN. La búsqueda de los mismos se puede hacer por título, autor o ISBN, entre otros filtros, y clasificarlos y ordenarlos por criterios como la cantidad de uso que se los ha dado o si son de tapa dura o blanda.

- **My Library**: esta aplicación para dispositivos Android[12] permite crear una biblioteca personal y realizar búsquedas dentro de ella. Las opciones que ofrece para añadir libros a la colección son tres: escaneando el código de barras, introduciendo el ISBN del libro o añadirlo manualmente. Además, se pueden realizar búsquedas dentro de la aplicación mediante títulos, autores, categorías, etc., ordenarlos atendiendo a diversos criterios o exportar la biblioteca.
- **Google Books**: el gran buscador también dispone de un buscador[13] específico para libros. Probablemente sea una de las mayores bibliotecas digitales que existen. El primer método de entrada que nos ofrece es únicamente manual, donde podemos introducir títulos, autores, géneros o el ISBN de la obra que buscamos, aunque una vez que muestra resultados aparecen otros filtros de búsqueda, como el tipo de libro (eBook, gratuito, con vista previa, etc.), si buscamos un libro, revista o periódico, o la fecha de publicación. Para cada obra se muestra información y detalles de ella, y en muchos de ellos una vista previa de su contenido o incluso, en algunos casos, la posibilidad de leerlo completo y descargarlo, así como enlaces a plataformas donde se pueden comprar. Aunque aparte de esto, una de sus funcionalidades más poderosas es la API[14] que ofrece, proporcionando a desarrolladores la oportunidad de enviar peticiones con los criterios de búsqueda y recibir en formato JSON el resultado de la misma.

2.3.2. Comparadores de precios

- **Google Shopping**: Google Shopping[15] es uno de los mejores comparadores de precios de productos similares. Este buscador de precios es especialmente popular entre aquellos que se dedican al dropshipping, un modelo de negocio que permite a los propietarios de una tienda online vender productos a sus clientes sin que sea necesario almacenarlos previamente. Desde la perspectiva del usuario, Google obtiene los datos de toda la web y los presenta de una manera sencilla. Tan solo hay que buscar el producto que se desea comparar y se muestran los resultados ordenados por su precio, aunque también ofrece otros criterios de búsqueda y ordenación. Cuando se hace clic en un producto, se muestran algunas fotos, una descripción del mismo, reseñas e información sobre los precios de diferentes vendedores.
- **idealo**: idealo[16] es un sitio web de comparación de precios de productos de Amazon, eBay y otros famosos marketplaces. Su funcionamiento es similar al del resto de comparadores de precios, se introduce el nombre del producto deseado en la barra de búsqueda y se obtienen páginas con miles de productos, junto a su precio. Al hacer clic en un artículo en particular, se muestra la descripción del producto, el número de ofertas que tiene, las reseñas e incluso los precios internacionales.

- **Kelkoo**: Kelkoo[17] es otro comparador de precios que destaca principalmente por permitir al usuario ver la evolución de los precios a lo largo del tiempo. Más que un historial de precios como tal lo que ofrece esta web es establecer alertas de precios para que, así, el usuario pueda finalizar la compra cuando el precio sea más bajo.
- **ASOS**: ASOS[18] es un buscador online especializado en ropa. Tiene relación con un gran número de tiendas y ofrece en su catálogo miles de productos. En su web, se puede buscar un producto y se muestran todos los artículos que coinciden con la búsqueda, pudiendo comparar entre los diferentes vendedores y encontrar el producto que mejor se adapta a los deseos del cliente. Además de esto, su principal semejanza con este proyecto es una funcionalidad añadida en su aplicación móvil[19]. Se ha agregado la posibilidad de buscar un producto a partir de una imagen. Con el dispositivo móvil, se selecciona la imagen en la que aparece el producto a buscar, ya sea desde la galería o haciendo una foto con la cámara, y la aplicación se encarga de hacer el procesamiento de la imagen para mostrar una serie de productos que se asemejan al de la imagen.

Capítulo 3

Desarrollo del proyecto

Este tercer capítulo hace una descripción del trabajo que se ha llevado a cabo durante la elaboración del proyecto. Se hace seguimiento formal de la especificación de los requisitos que sigue la aplicación, la arquitectura que sigue el proyecto y el modelo de datos.

3.1. Especificación de requisitos

Este apartado muestra la especificación de requisitos del proyecto. Se hace, en primer lugar, una distinción de los actores que intervienen y a continuación, una serie de casos de uso que se corresponden con las diferentes tareas de la aplicación.

3.1.1. Actores del sistema

En el sistema participan dos actores diferenciados:

- **Actor sistema**: es el actor que representa al sistema, se encarga de la búsqueda de los libros y sus enlaces de compra, y de gestionar los favoritos.
- **Actor usuario**: es el actor que representa al usuario final, se encarga entre otras cosas, de hacer las peticiones para las búsquedas, marcar y desmarcar libros como favoritos e iniciar y cerrar sesión.

3.1.2. Casos de uso

En la figura 3.1 podemos ver el diagrama de casos de uso para el actor sistema, mientras que el diagrama de casos de uso para el actor usuario se puede apreciar en la figura 3.2.

3.1.2.1. Casos de uso del sistema

En este apartado se pueden ver los casos de uso de actor sistema.

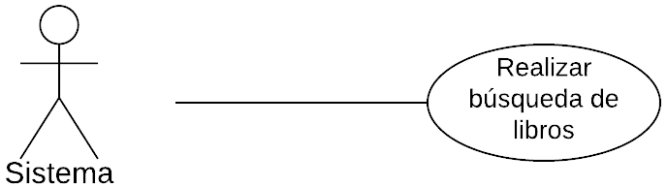


Figura 3.1: Casos de uso - Sistema

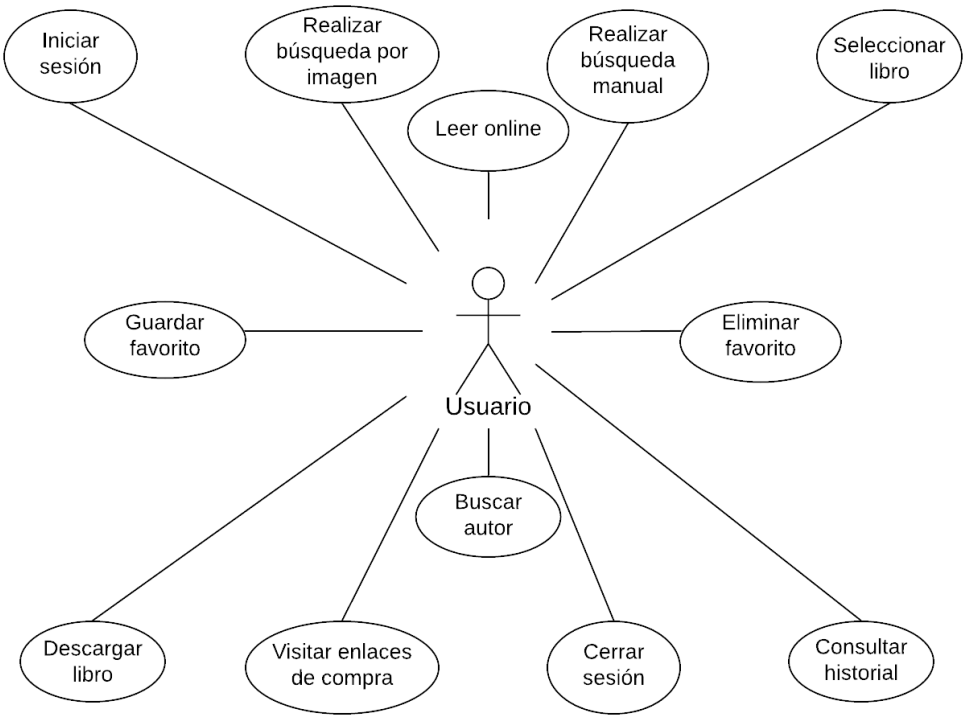


Figura 3.2: Casos de uso - Usuario

UC-0001	**Realizar búsqueda de libros**	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El sistema realizará una búsqueda de libros conforme a los criterios establecidos	
Precondición	El usuario ha establecido unos criterios de búsqueda, ya sean manuales o por imagen	
Secuencia normal	**Paso**	**Acción**
	1	El sistema recoge los datos de entrada del usuario
	2	Se realiza la búsqueda con los datos de entrada
	3	Se agrupan los datos en formato JSON y se devuelven a la aplicación
Postcondición	Los datos resultantes son devueltos a la aplicación	
Excepciones	**Paso**	**Acción**
	1	Si no hay datos de entrada, se muestra un mensaje pidiendo que se introduzcan
	2	Si no se encuentran resultados para la búsqueda, se muestra un mensaje indicándolo
Frecuencia	Por cada petición de búsqueda	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.3: Caso de uso 1 - Realizar búsqueda de libros

3.1.2.2. Casos de uso del usuario

A continuación se muestran los casos de uso del actor usuario.

UC-0002	**Iniciar sesión**	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario iniciará sesión en la aplicación	
Precondición	No hay ninguna sesión iniciada	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce sus credenciales
	2	Se validan las credenciales en el servidor
	3	Si las credenciales son validadas, se inicia la sesión en el dispositivo
Postcondición	El usuario inicia sesión	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no existe, se muestra un mensaje de error
	2	Si las credenciales no son correctas, se muestra un mensaje de error
Frecuencia	Por cada inicio de sesión	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.4: Caso de uso 2 - Iniciar sesión

UC-0003	**Realizar búqueda por imagen**	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario realiza una búsqueda indicando una imagen como entrada	
Precondición	La aplicación tiene los permisos de cámara y almacenamiento	
Secuencia normal	**Paso**	**Acción**
	1	El usuario indica que quiere realizar una búsqueda por imagen
	2	Indica si la imagen será tomada por la cámara o de la galería
	3	Selecciona la imagen
	4	La imagen se envía al servidor donde se hace el reconocimiento de texto
	5	Si se encuentran resultados, se devuelven a la aplicación en un JSON
	6	Se muestran los resultados al usuario
Postcondición	Se muestra el resultado de la búsqueda	
Excepciones	**Paso**	**Acción**
	1	Si no reconoce texto en la imagen se muestra un mensaje de error
	2	Si no se han encontrado resultados que coincidan se notifica al usuario
Frecuencia	Por cada búsqueda por imagen	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.5: Caso de uso 3 - Realizar búsqueda por imagen

UC-0004	Realizar búqueda manual	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario realiza una búsqueda introduciendo los criterios manualmente	
Precondición	El usuario inicia un búsqueda manual	
Secuencia normal	Paso	Acción
	1	El usuario indica que quiere realizar una búsqueda manual
	2	Introduce el título y/o el autor de la obra que busca
	3	Los datos se envían al servidor donde se realiza la búsqueda
	4	Si se encuentran resultados, se devuelven a la aplicación en un JSON
	5	Se muestran los resultados al usuario
Postcondición	Se muestra el resultado de la búsqueda	
Excepciones	Paso	Acción
	1	Si los dos campos están vacíos, se pide que se rellene al menos uno
	2	Si no se han encontrado resultados que coincidan se notifica al usuario
Frecuencia	Por cada búsqueda manual	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.6: Caso de uso 4 - Realizar búsqueda manual

UC-0005	Seleccionar libro	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario selecciona un libro de la lista de resultados	
Precondición	Se ha realizado una búsqueda de libros	
Secuencia normal	Paso	Acción
	1	El usuario se encuentra ante la lista de resultados de una búsqueda
	2	Selecciona el libro que desea consultar
	3	Se muestra el detalle del libro
Postcondición	Se abre el detalle del libro	
Excepciones	Paso	Acción
	-	-
Frecuencia	Por cada búsqueda	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.7: Caso de uso 5 - Seleccionar libro

UC-0006	Leer online	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Lectura online del libro seleccionado	
Precondición	Se ha seleccionado un libro	
Secuencia normal	Paso	Acción
	1	El usuario se encuentra ante la lista de resultados de una búsqueda
	2	Se abre un navegador interno con la vista previa del libro
Postcondición	Se abre el libro en el navegador	
Excepciones	Paso	Acción
	1	Si no se encuentra la dirección del lector online se muestra un error
Frecuencia	Por cada apertura del lector online	
Importancia	Alta	
Urgencia	Baja	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.8: Caso de uso 6 - Leer online

UC-0007	Guardar favorito	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se guarda como favorito un libro seleccionado	
Precondición	Se ha seleccionado un libro y no es favorito	
Secuencia normal	Paso	Acción
	1	En la vista de detalle del libro, el usuario pulsa sobre el icono del corazón
	2	Se envía la petición al servidor
	3	Se inserta el libro en la tabla de favoritos
Postcondición	El libro es guardado como favorito y se muestra con el corazón relleno	
Excepciones	Paso	Acción
	1	Si no se ha podido insertar el libro como favorito, se muestra un error
Frecuencia	Cada vez que se añade un libro a favoritos	
Importancia	Alta	
Urgencia	Media	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.9: Caso de uso 7 - Guardar favorito

UC-0008	Eliminar favorito	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se elimina de favoritos un libro seleccionado	
Precondición	Se ha seleccionado un libro y es favorito	
Secuencia normal	Paso	Acción
	1	En la vista de detalle del libro, el usuario pulsa sobre el icono del corazón
	2	Se envía la petición al servidor
	3	Se elimina el libro de la tabla de favoritos
Postcondición	El libro es eliminado de favoritos y se muestra con el corazón vacío	
Excepciones	Paso	Acción
	1	Si no se ha podido eliminar el libro de favoritos, se muestra un error
Frecuencia	Cada vez que se elimina un libro de favoritos	
Importancia	Alta	
Urgencia	Media	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.10: Caso de uso 8 - Eliminar favorito

UC-0009	Buscar autor	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se buscan libros del mismo autor que el del libro seleccionado	
Precondición	Se ha seleccionado un libro	
Secuencia normal	Paso	Acción
	1	En la vista de detalle del libro, el usuario pulsa sobre el nombre del autor
	2	Se realiza una búsqueda con el autor del libro seleccionado
	3	Se muestran los resultados de la búsqueda
Postcondición	Se muestran los resultados de la búsqueda	
Excepciones	Paso	Acción
	1	Si no se han encontrado resultados para la búsqueda, se notifica
Frecuencia	Cada vez que se busca el autor de un libro	
Importancia	Media	
Urgencia	Media	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.11: Caso de uso 9 - Buscar autor

UC-00010	Descargar libro	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se descarga el libro seleccionado en formato digital	
Precondición	Se ha seleccionado un libro y está disponible la opción de descarga	
Secuencia normal	Paso	Acción
	1	En la vista de detalle del libro, el usuario pulsa sobre el icono de descarga
	2	Se abre el navegador para proceder a la descarga
	3	Se acepta la descarga del archivo
Postcondición	Se descarga el archivo	
Excepciones	Paso	Acción
	1	Si no existe el link de descarga, se notifica
Frecuencia	Cada vez que se descarga un libro	
Importancia	Media	
Urgencia	Media	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.12: Caso de uso 10 - Descargar libro

UC-00011	Visitar enlaces de compra	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se visitan varias plataformas digitales para la compra del ejemplar	
Precondición	Se ha seleccionado un libro	
Secuencia normal	Paso	Acción
	1	Con un libro seleccionado, el usuario pulsa sobre el enlace a la tienda en la que quiera comprar el libro
	2	Se abre el navegador interno en la aplicación con la web de la tienda
Postcondición	Se muestra el navegador con la opción de compra	
Excepciones	Paso	Acción
	-	-
Frecuencia	Cada vez que se pulse un enlace de compra	
Importancia	Vital	
Urgencia	Inmediatamente	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.13: Caso de uso 11 - Visitar enlaces de compra

UC-00012	Consultar historial	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se conulta el historial de los libros visitados por el usuario	
Precondición	Se han consultado libros previamente	
Secuencia normal	Paso	Acción
	1	El usuario se dirige a la pestaña 'Perfil' y selecciona la opción historial
	2	Se envía la petición al servidor para recuperar los últimos libros
	3	Se devuelve a la aplicación el listado de los libros que se han consultado
Postcondición	Se muestran los libros que han sido consultados	
Excepciones	Paso	Acción
	1	Si aun no se ha visitado ningún libro, se indica mediante un mensaje
Frecuencia	Cada vez que se consulte el historial	
Importancia	Media	
Urgencia	Media	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.14: Caso de uso 12 - Consultar historial

UC-00013	Cerrar sesión	
Versión	1.0	
Autores	Iván Monterrubio Cerezo	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	Se cierra la sesión del usuario	
Precondición	Se ha iniciado sesión	
Secuencia normal	Paso	Acción
	1	El usuario se dirige a la pestaña 'Perfil' y selecciona cerrar sesión
	2	Se muestra la pantalla principal de la aplicación con el inicio de sesión
Postcondición	Se cierra la sesión del usuario	
Excepciones	Paso	Acción
	-	-
Frecuencia	Cada vez que se cierre sesión	
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 3.15: Caso de uso 13 - Cerrar sesión

3.2. Arquitectura

La aplicación resultante del desarrollo del proyecto sigue una arquitectura cliente-servidor[20], siendo el cliente la aplicación móvil multiplataforma y el servidor una máquina que provee al cliente de los datos correspondientes a los libros y sus búsquedas, así como el resto de servicios.

La decisión de seguir esta arquitectura viene dada por presentar las siguientes ventajas:

- **Administración centrada en el servidor**: el cliente apenas presenta necesidades de administración.
- **Centralización de los recursos**: los recursos de todos los usuarios se encuentran en un solo servidor, evitando así la inconsistencia y la redundancia

de las bases de datos.

- **Mejora de la seguridad**: se disminuyen las posibilidades de un acceso indebido ya que se dispone de un mecanismo de autenticación centralizado.
- **Escalabilidad de la instalación**: la red y su funcionamiento no se ven afectados por la incorporación o eliminación de usuarios.

En la figura 3.16 se presenta un esquema de la arquitectura sobre la que se desarrolla el proyecto.

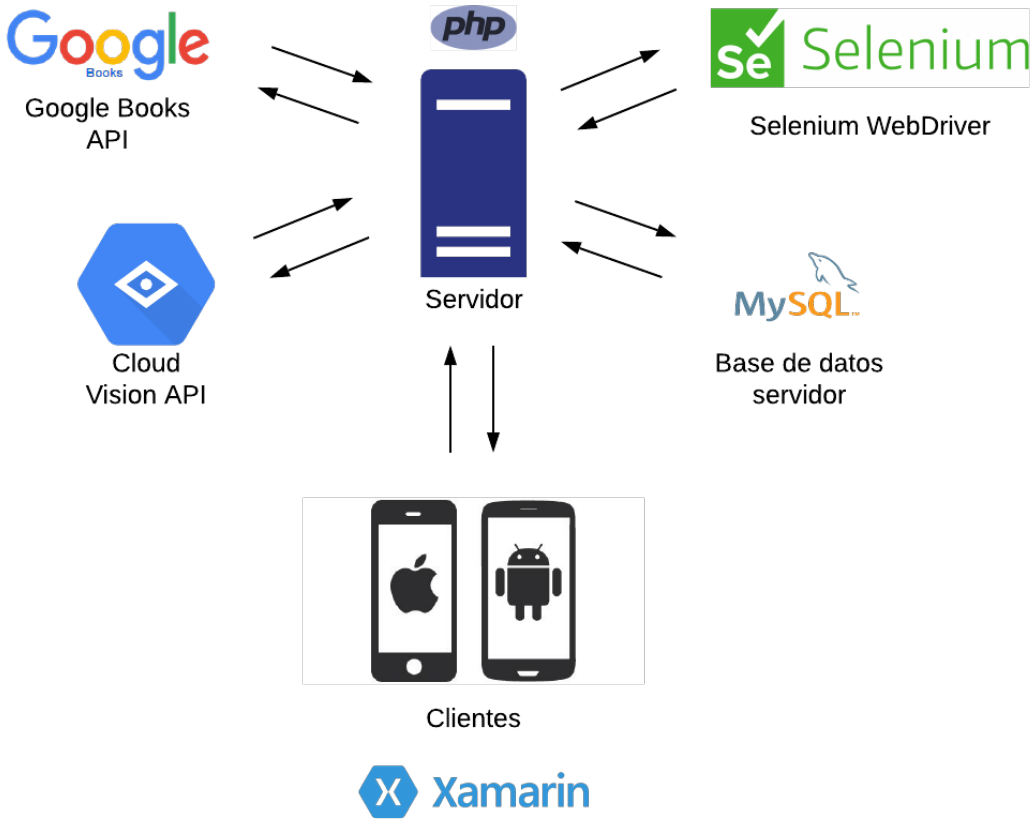


Figura 3.16: Arquitectura del proyecto

3.3. Modelo de datos

Esta sección habla de la base de datos empleada en el proyecto, su modelo entidad-relación y una descripción de su diseño.

3.3.1. Modelo Entidad-Relación

En la figura 3.17 podemos apreciar el modelo entidad-relación de la base de datos del proyecto, que se trata de una base de datos relacional[21] SQL con cuatro tablas

que almacenan los datos relativos a cada usuario.

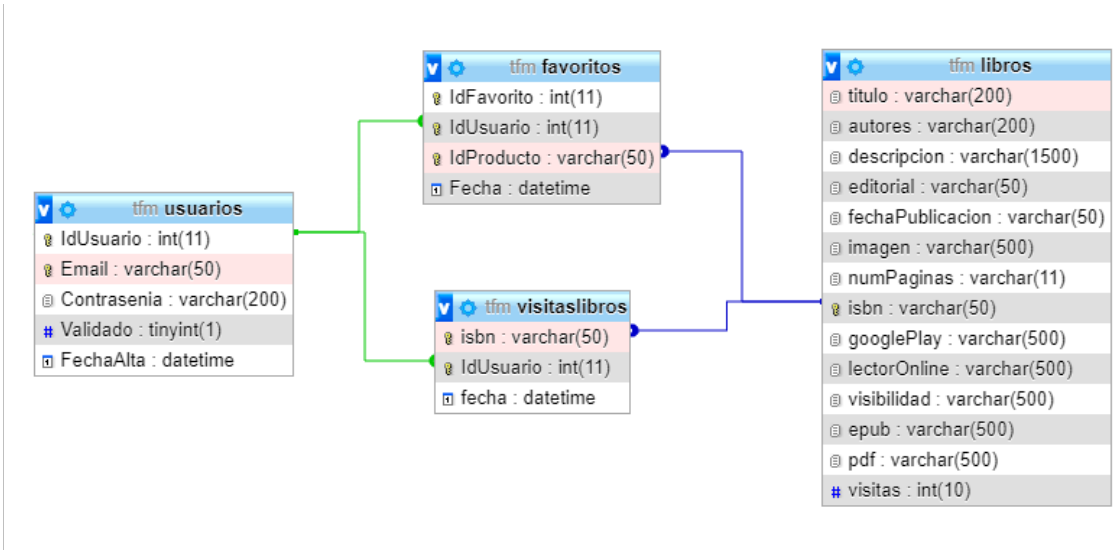


Figura 3.17: Modelo Entidad-Relación

3.3.2. Base de datos

La base de datos empleada en el proyecto es una base de datos relacional MySQL[22]. En ella, hay cuatro tablas encargadas de almacenar toda la información relativa a los usuarios y sus acciones, como los libros que han guardado como favoritos o el historial de libros consultados. A continuación se detallan cada una de las tablas.

3.3.2.1. Tabla *usuarios*

- **Descripción**: almacena todos los usuarios que se han registrado en la aplicación móvil, pudiendo estar validados o no.
- **Columnas**:
  - *IdUsuario*: identificador único de cada usuario.
  - *Email*: correo electrónico con el que se registra el usuario.
  - *Contrasenia*: contraseña con la que se registra el usuario. Se almacena encriptada.
  - *Validado*: valor que indica si el usuario se ha validado accediendo al enlace que le llega por correo electrónico, obteniendo acceso así a iniciar sesión en la aplicación.

3.3.2.2. Tabla *favoritos*

- **Descripción**: cada una de sus filas se corresponde con una relación entre un libro y un usuario, correspondiéndose a que el usuario ha marcado como favorito ese libro.

- **Columnas**:
  - *IdFavorito*: identificador único de cada favorito.
  - *IdUsuario*: identificador del usuario que ha guardado ese libro como favorito.
  - *IdProducto*: identificador del libro que ha sido marcado como favorito.
  - *Fecha*: fecha de inserción del favorito

3.3.2.3. Tabla *visitaslibros*

- **Descripción**: tabla en la que se registra cada vez que un usuario consulta un libro. Es utilizada para el historial.
- **Columnas**:
  - *isbn*: identificador del libro que se ha visitado.
  - *IdUsuario*: identificador del usuario que ha visitado ese libro.
  - *IdProducto*: identificador del libro que ha sido marcado como favorito.
  - *fecha*: fecha en que se visitó el libro.

3.3.2.4. Tabla *libros*

- **Descripción**: tabla en la que se registran cada uno de los libros que han sido visitados por algún usuario junto a todos sus atributos.
- **Columnas**:
  - *titulo*: título del libro.
  - *autores*: autor o autores de la obra.
  - *descripcion*: descripción del libro.
  - *editorial*: editorial en la que está publicado el ejemplar.
  - *fechaPublicacion*: fecha en la que se publicó el libro.
  - *imagen*: enlace a la portada del libro.
  - *numPaginas*: número de páginas.
  - *isbn*: isbn del ejemplar. Es el identificador único de cada libro.
  - *googlePlay*: enlace de compra del libro en Google Play.
  - *lectorOnline*: enlace al lector online donde se puede leer parte o la totalidad del libro.
  - *visibilidad*: valor que indica si el libro se puede leer online por completo o tan solo muestra una vista previa.
  - *epub*: enlace de descarga del libro en formato ePUB.
  - *pdf*: enlace de descarga del libro en formato PDF.
  - *visitas*: número de veces que los usuarios han visitado este ejemplar.

3.4. Herramientas tecnológicas

En este capítulo se describen las herramientas tecnológicas que han sido empleadas durante el proceso de desarrollo del proyecto.

3.4.1. Lenguajes

3.4.1.1. C#

C#[23] es un lenguaje de programación diseñado por la compañía Microsoft. Está estandarizado por ECMA e ISO, dos de las organizaciones más importantes a la hora de crear estándares para los servicios o productos. El lenguaje de programación C# está orientado a objetos. Es considerado como una evolución de sus lenguajes antecesores, C y C++, y surge ante la necesidad de crear una alternativa a los lenguajes existentes ante los problemas que la compañía tuvo con la empresa creadora del lenguaje Java. Es por esto que C# presenta los atributos positivos de C++, Java y Visual Basic y los mejora otorgando un lenguaje fuerte y actualizado para los tiempos actuales. Una de sus funcionalidades es crear sitios y aplicaciones web, así como la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles.

3.4.1.2. PHP

PHP[24] es un lenguaje de código abierto dedicado al desarrollo web. PHP se distingue de otros lenguajes en que es ejecutado en la parte del servidor, generando código HTML que se envía directamente al cliente. Este recibe el resultado tras ejecutar el script sin tener la necesidad de saber cuál es su contenido. PHP es soportado por la gran mayoría de servidores y sistemas operativos actuales. Con él se puede programar por procedimientos, usar la programación orientada a objetos o parte de ambas. Una de sus más preciadas características es ofrecer soporte y documentación con una gran variedad de bases de datos.

La aplicación más común de PHP es ejecutar scripts en el lado del servidor.

3.4.1.3. SQL

SQL[25] es un tipo de lenguaje vinculado con la gestión de bases de datos de carácter relacional que permite la especificación de distintas clases de operaciones entre éstas. Gracias a la utilización del álgebra y de cálculos relacionales, SQL brinda la posibilidad de realizar consultas con el objetivo de recuperar información de las bases de datos de manera sencilla.

3.4.2. API empleadas

3.4.2.1. Cloud Vision API

La API Vision[6] de Google Cloud ofrece modelos de aprendizaje automático preparados previamente y muy potentes a través de las API REST y RPC. Asigna etiquetas a imágenes y las clasifica rápidamente en millones de categorías predefinidas. Detecta objetos y caras, lee texto impreso y manuscrito, y consigue metadatos de gran valor.

3.4.2.2. Google Books API

La API de Google Books[14] permite realizar de manera programática la mayoría de las operaciones que se pueden realizar de manera interactiva en el sitio web de Google Books, permite realizar búsquedas de texto completo y recuperar información de libros, visibilidad y disponibilidad de libros electrónicos, así como administrar estanterías personales.

3.4.2.3. Selenium WebDriver

Selenium WebDriver[26] es una API compacta orientada a objetos que maneja un navegador de forma nativa, como lo haría un usuario real, ya sea localmente o en máquinas remotas. Está diseñado como una interfaz de programación concisa y compacta. Selenium WebDriver se refiere tanto a los enlaces de idioma como a las implementaciones del código de control del navegador individual. Esto se conoce comúnmente solo como WebDriver.

3.4.3. Librerías utilizadas

3.4.3.1. Newtonsoft.Json

Newtonsoft.Json[27] es una librería de código abierto y licencia MIT para crear, analizar, consultar y modificar datos en formato JSON. Se trata de un marco JSON de alto rendimiento popular para .NET. Con él, se puede serializar y deserializar cualquier objeto .NET, convertir a XML y viceversa, y correr en cualquier plataforma .NET como Xamarin.

3.4.3.2. PHPMailer

PHPMailer[28] es una clase destacada del ecosistema PHP que facilita el envío de email desde PHP, sobre todo, en correos especialmente complejos. Está disponible con licencia de libre distribución y uso.

3.4.3.3. AES Encryption

Advanced Encryption Standard[29] (AES), también conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos, creado en Bélgica. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001 después de un proceso de estandarización que duró 5 años. Se transformó en un estándar efectivo el 26 de mayo de 2002. Desde 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica.

3.4.4. Plataformas y entornos de desarrollo

3.4.4.1. Visual Studio

El entorno de desarrollo integrado de Visual Studio[30] es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Un entorno de desarrollo integrado (IDE) es un programa con numerosas características que se pueden usar para muchos aspectos del desarrollo de software. Más allá del editor estándar y el depurador que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.

3.4.4.2. Xamarin

Xamarin[31] es una plataforma de código abierto para compilar aplicaciones modernas y con mejor rendimiento para iOS, Android y Windows con .NET. Xamarin es una capa de abstracción que administra la comunicación de código compartido con el código de plataforma subyacente. Xamarin se ejecuta en un entorno administrado que proporciona ventajas como la asignación de memoria y la recolección de elementos no utilizados.

3.4.4.3. PHPStorm

PhpStorm[32] es un IDE comercial multiplataforma para PHP creado por la empresa JetBrains. PhpStorm proporciona un editor para PHP , HTML y JavaScript con análisis de código sobre la marcha, prevención de errores y refactorizaciones automatizadas para código PHP y JavaScript. Está escrito en Java, y los usuarios pueden extender el IDE instalando complementos creados para PhpStorm o escribiendo sus propios complementos. El software también se comunica con fuentes externas como XDebug.

3.4.4.4. XAMPP

XAMPP[33] es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El programa se distribuye con la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas.

3.4.5. Formatos de datos

3.4.5.1. JSON

JSON[34] es un formato de texto ligero de intercambio de datos completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

Capítulo 4

Diseño de la aplicación

Este capítulo explica cuáles son los principales módulos y clases del proyecto, a qué funcionalidad de la aplicación corresponden y cómo se han desarrollado.

El diseño del proyecto, así como el de todas sus clases, tiene dos ámbitos: el cliente, correspondiente a la aplicación móvil e implementado en clases C# mediante la plataforma Xamarin, y el servidor, que se corresponde con todo el procesamiento de peticiones, búsquedas y comunicación con la base de datos, implementado en ficheros PHP en un servidor Apache.

Para cada funcionalidad se describe cómo interactúa el usuario con ella, qué procedimientos activa, cómo se comunica con el servidor, y cómo se devuelve la respuesta una vez que se ha terminado de ejecutar el proceso asociado a cada una de ellas. En la figura 4.1 se puede apreciar el modelo de despliegue.

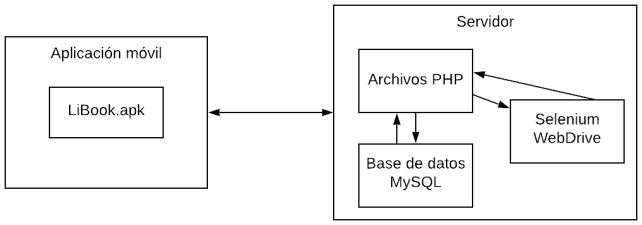


Figura 4.1: Modelo de despliegue

4.1. Registro

El registro de los usuarios por parte del cliente se encuentra implementado en la clase *Registro* del fichero *Registro.xaml.cs*. Esta recopila los datos introducidos por el usuario en el apartado *Registro* de la aplicación móvil y hace una serie de comprobaciones como que ninguno de los campos está vacío, el email tiene el formato correcto, o que las contraseñas coinciden. Si todas las comprobaciones son correctas, se encripta la contraseña con el método de encriptación bidireccional *AES* y se envía al servidor una petición de registro con los datos del usuario encapsulados en

formato JSON. En la figura 4.2 se puede ver un detalle del código que implementa esta funcionalidad en la aplicación móvil.

```
}
else
{
    Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");
    Match match = regex.Match(Email.Text);
    if (!match.Success)
    {
        DisplayAlert("Error", "Introduce un email válido.", "Aceptar");
        Email.Focus();
    }
    else
    {
        if (Contrasenia.Text.Length < 8)
        {
            DisplayAlert("Error", "La contraseña debe tener al menos 8 caracteres.", "Aceptar");
            Contrasenia.Focus();
        }
        else
        {
            if (Contrasenia.Text != ContraseniaRepetida.Text)
            {
                DisplayAlert("Error", "Las contraseñas no coinciden.", "Aceptar");
            }
            else
            {
                using (Aes myAes = Aes.Create())
                {
                    byte[] myKey = AESencryption.GenerateKey();
                    byte[] myIV = AESencryption.GenerateIV();
                    // Encrypt the string to an array of bytes.
                    byte[] encrypted = AESencryption.EncryptStringToBytes_Aes(Contrasenia.Text, myKey, myIV);

                    EnviarUsuarioServidorAsync(encrypted);
                }
```

Figura 4.2: Registro - Aplicación móvil

El servidor Apache recoge estos datos en el fichero *registro.php* del paquete *usuarios* y consulta en la base de datos si ya está registrado ese email. Si lo está, se devuelve a la aplicación un código de error y se muestra al usuario un mensaje que dice que ese email ya está registrado. En caso contrario, se inserta el nuevo usuario en la base de datos, con el campo *validado* a 0, y se envía un correo electrónico a la dirección de email facilitada con un enlace que permitirá validar el usuario, ya que hasta que el usuario no sea validado, no tendrá permiso para acceder a la aplicación.

La figura 4.3 muestra parte del código que recoge los datos, comprueba su existencia en la base de datos, y envía el email de confirmación.

La figura 4.4 muestra un ejemplo del correo electrónico que le llega al usuario para validar su cuenta, y una vez acceda a la dirección web que se le indica, su usuario pasará a estar validado.

Se puede apreciar una vista previa de esta pantalla en la figura 4.5.

4.2. Inicio de sesión

El diseño que sigue la funcionalidad de inicio de sesión es similar a la de registro en cuanto a las conexiones que establece con el servidor y la forma en que el usuario la lleva a cabo.

En primer lugar, el usuario introduce sus credenciales de acceso, las cuales son

```php
$post_vars = json_decode(file_get_contents("php://input"),true);
$email = $post_vars["Email"];
$contraseniaAPP = $post_vars["Contrasenia"];

$query="select * from Usuarios where Email='$email'";

if($resultado = db::getInstance()->get_result($query)) {
    $numregistros = $resultado->num_rows;
    if ($numregistros == 0) {
        $query = "insert into Usuarios values (null, '$email', '$contraseniaAPP', 0, now())";
        if(db::getInstance()->dbquery($query)){
            $id = db::getInstance()->insert_id;
            $aes = new AES($id, $key, $blockSize);
            $account = $aes->encrypt();

            if(enviar_email($email, $account)){
                echo 0; //Usuario insertado en la BD
            }
```

Figura 4.3: Registro - Servidor

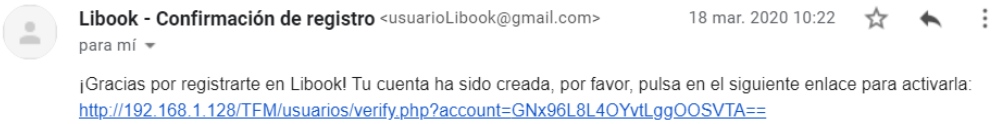


Figura 4.4: Registro - Email de confirmación

el correo electrónico y la contraseña. La aplicación comprueba que ambos campos hayan sido introducidos, se encripta la contraseña mediante el sistema de cifrado AES, y se envía la información al usuario en forma de objeto JSON. Todo esto lo lleva a cabo la clase *Login* del fichero *Login.xaml.cs*. La figura 4.6 contiene una porción del código que lleva a cabo esta acción.

El fichero *login.php* del paquete *usuarios* del servidor recoge las credenciales del usuario y comprueba en primera instancia si el usuario se encuentra en la base de datos. En caso afirmativo, se comprueba que la contraseña introducida por el usuario sea la correcta. Para ello, se recupera la contraseña encriptada almacenada en la base de datos, se procede a desencriptar tanto la que estaba almacenada como la que ha llegado en la petición, mediante el método de desencriptación por clave privada de AES, y se comparan una vez que de han descifrado. Si coinciden, se revisa que el usuario esté validado, y si es así, se devuelve un mensaje de éxito a la aplicación y el usuario inicia sesión en la aplicación. En caso de que alguna de las comprobaciones falle, se notifica al usuario con el mensaje adecuado en cada caso. La figura 4.7 muestra parte de este proceso.

Una vez que el usuario se ha logueado, la sesión permanece abierta en ese dispositivo hasta que se cierre manualmente o se borren los datos de la aplicación.

La figura 4.8 muestra la pantalla de la aplicación móvil para esta funcionalidad.

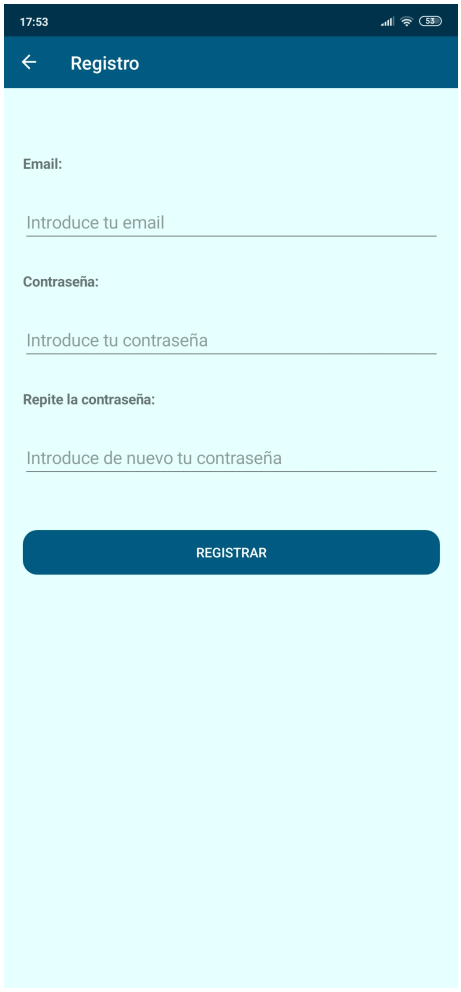


Figura 4.5: Pantalla de Registro de la aplicación móvil

```php
$query="select * from Usuarios where Email='$email'";

if($resultado = db::getInstance()->get_result($query)) {
    if ($resultado->num_rows > 0) {
        $usuario = $resultado->fetch_assoc();

        $contraseniaDB_encriptada = new AES($usuario["Contrasenia"], $key, $blockSize);
        $contraseniaDB_desencriptada = $contraseniaDB_encriptada->decrypt();

        $contraseniaAPP_encriptada = new AES($contraseniaAPP, $key, $blockSize);
        $contraseniaAPP_desencriptada = $contraseniaAPP_encriptada->decrypt();

        if ($contraseniaDB_desencriptada == $contraseniaAPP_desencriptada) {
            //Las contraseñas coinciden
            if($usuario["Validado"] == 0)
                //Usuario no validado
                echo json_encode(array("usuario_id" => $usuario["IdUsuario"],"estado" => 2));
            else //Usuario validado
                echo json_encode(array("usuario_id" => $usuario["IdUsuario"],"estado" => 0));
        } else
            //Las contraseñas no coinciden
            echo json_encode(array("usuario_id" => $usuario["IdUsuario"],"estado" => 1));
    }
```

Figura 4.7: Inicio de sesión - Servidor

```
if (String.IsNullOrWhiteSpace(email.Text))
{
    DisplayAlert("Error", "Introduce tu email.", "Aceptar");
    email.Focus();
}
else
{
    if (String.IsNullOrWhiteSpace(contra.Text))
    {
        DisplayAlert("Error", "Introduce tu contraseña.", "Aceptar");
        contra.Focus();
    }
    else
    {
        using (Aes myAes = Aes.Create())
        {
            byte[] myKey = AESencryption.GenerateKey();
            byte[] myIV = AESencryption.GenerateIV();
            // Encrypt the string to an array of bytes.
            byte[] encrypted = AESencryption.EncryptStringToBytes_Aes(contra.Text, myKey, myIV);

            EnviarUsuarioServidorAsync(encrypted);
        }
    }
}
```

Figura 4.6: Inicio de sesión - Aplicación móvil

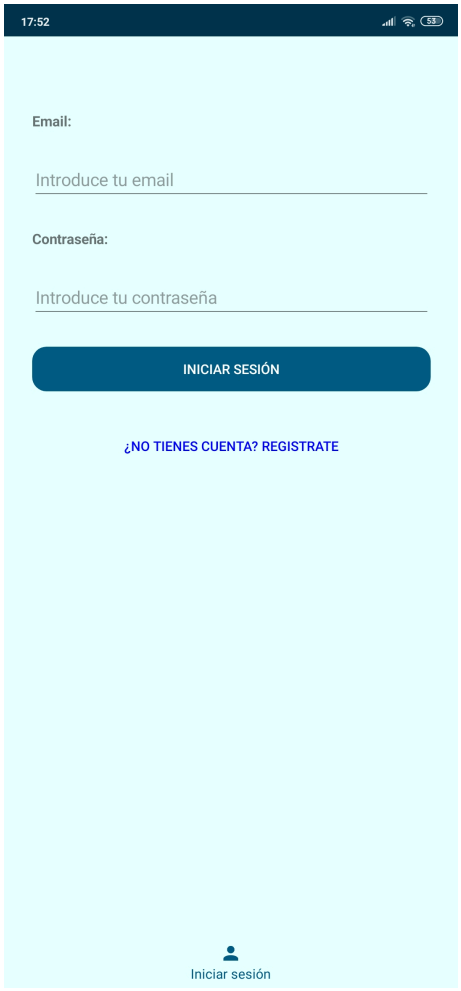


Figura 4.8: Pantalla de Inicio de Sesión de la aplicación móvil

4.3. Cierre de sesión

El cierre de la sesión es la única funcionalidad en la aplicación móvil que carece de una conexión con el servidor. Toda su implementación es llevada a cabo localmente, ya que la sesión se guarda únicamente en el dispositivo donde se ejecuta la aplicación y no tiene ninguna relación con el servidor. Por tanto, su diseño se limita a al ámbito local de la propia aplicación, donde la clase *Usuario* encargada de la sección *Perfil*, envía un mensaje interno a la clase *MainTabbedPage*, la cual se encarga de la gestión de las pestañas, en el que se indica que el usuario quiere cerrar sesión. Entonces, se procede a borrar los datos de usuario almacenados en el dispositivo y se cambian las pestañas a la pestaña de *Inicio de sesión*, tal y como aparece la primera vez que se inicia la aplicación.

En la figura 4.9 se puede apreciar el comportamiento de la aplicación cuando se genera un aviso de cierre de sesión.

```
MessagingCenter.Subscribe<Usuario>(this, "Out", (sender) =>
{
    Xamarin.Forms.Application.Current.Properties.Clear();

    Page perfil = new Login("login");
    perfil.Title = "Iniciar sesión";

    Children.Clear();
    Children.Add(perfil);
});
```

Figura 4.9: Cierre de sesión - Aplicación móvil

4.4. Búsqueda de libros por imagen

La búsqueda de libros por imagen es una de las dos funcionalidades principales y que, junto a la búsqueda de enlaces de compra, dotan a LiBook de una personalidad propia.

Esta funcionalidad tiene parte de desarrollo local y parte en el servidor, por lo que como es habitual en el diseño del proyecto, la aplicación recopila los datos con los que quiere hacer la búsqueda, estos se envían al servidor y él se encarga de llevarla a cabo y devolver los resultados. En esta ocasión, los datos de entrada para la búsqueda es una imagen que el usuario proporciona a través de la aplicación y que se corresponde con la portada del libro a buscar. Esta imagen podrá ser seleccionado mediante la cámara del dispositivo o escogiendo una imagen existente en su galería, por lo que previamente, se habrán debido de aceptar los permisos de acceso a la cámara y al almacenamiento del dispositivo. La aplicación cuenta con dos botones para la elección de la imagen por cualquiera de los dos métodos. Una vez que se ha seleccionado la imagen, se convierte a formato JSON junto al ID del usuario que hace la petición y se envía al servidor. La figura 4.10 muestra la función por la que se toma una imagen desde la cámara y se envía al proceso que se conecta con el

servidor, que tiene lugar en la clase *Cámara* del fichero *Camara.xaml.cs*.

```
takePhoto.Clicked += async (sender, args) =>
{
    if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported)
    {
        await DisplayAlert("No Camera", ":( No camera available.", "OK").ConfigureAwait(false);
        return;
    }

    var file = await CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions
    {
        Directory = "Test",
        //para guardarlo en la galeria tambien
        SaveToAlbum = true,
        CompressionQuality = 75,
        CustomPhotoSize = 50,
        PhotoSize = PhotoSize.MaxWidthHeight,
        MaxWidthHeight = 2000,
        DefaultCamera = CameraDevice.Front
    }).ConfigureAwait(false);

    if (file == null)
        return;

    App.Current.Resources["pathImagenTomada"] = file.Path;

    await Navigation.PushAsync(new ImageResult(new ItemImagen(file))).ConfigureAwait(true);
};
```

Figura 4.10: Búsqueda de libros por imagen - Aplicación móvil

Cuando el objeto JSON que contiene la imagen llega al servidor, es capturado por el fichero *recortarImagenesServidor.php* del paquete *imagenes* y se envía a la API de Google Cloud Vision para su procesamiento. Este servicio nos ofrece información relevante de la imagen como el texto que aparece en ella, del que se identifica el autor y el título de la obra y se envía a otra función del paquete *books*. El fichero *buscarLibro.php* de este paquete envía estos datos a la API Google Books solicitando la búsqueda de libros que coincidan con el autor y el título extraídos de la imagen proporcionada por el usuario. La API devuelve el resultado de la búsqueda en formato JSON con los libros que coinciden con la búsqueda aportando todos los datos que posteriormente se pueden visualizar en la aplicación, como una descripción, la editorial, el número de hojas o el enlace a un lector online en el que se podrá leer un fragmento o, en ocasiones, la totalidad del libro. Cuando se tiene el resultado de la búsqueda, se parsean los datos que contiene el JSON para seleccionar estos datos que se devolverán a la aplicación y además, con el ID del usuario que inició la petición, se realiza una búsqueda en la base de datos para comprobar si alguno de los libros encontrados están marcados como favorito por el usuario. Por último, se convierten toda esta información a un nuevo objeto JSON que es devuelto a la aplicación para mostrar la lista con los resultados obtenidos.

La figura 4.11 muestra una porción del fichero *recortarImagenesServidor.php* en el que se captura la imagen, se envía a Cloud Vision, y se pasa el resultado al fichero *buscarLibro.php* para su posterior solicitud a Google Books. La figura 4.12 muestra, por su parte, un detalle de este proceso en el que se hace la búsqueda de los libros.

Como se puede apreciar en la figura 4.13, el usuario cuenta con las dos opciones de búsqueda por imagen.

```php
$fichero = $ruta_imagenOriginal;
$imageResource = fopen($fichero, 'r');
$response = $imageAnnotatorClient->annotateImage($imageResource, $features);
if(is_null($response->getFullTextAnnotation())){
    //No se ha reconocido texto en la imagen
    $respuesta["Estado"] = -1;
    echo json_encode($respuesta);
    exit;
}
$textoFoto = str_replace("\n", ' ', $response->getFullTextAnnotation()->getText());

$respuesta["Libros"] = buscarLibro($textoFoto, $idUsuario);

echo json_encode($respuesta);
```

Figura 4.11: Búsqueda de libros por imagen - Servidor

```php
function buscarLibro($textoFoto, $idUsuario){
    $client = new Google_Client();
    $client->setApplicationName("Client_Library_Examples");
    $client->setDeveloperKey("**********************");

    $service = new Google_Service_Books($client);
    $optParams = array('printType' => 'books');
    $results = $service->volumes->listVolumes($textoFoto, $optParams);
```

Figura 4.12: Búsqueda de libros - Servidor

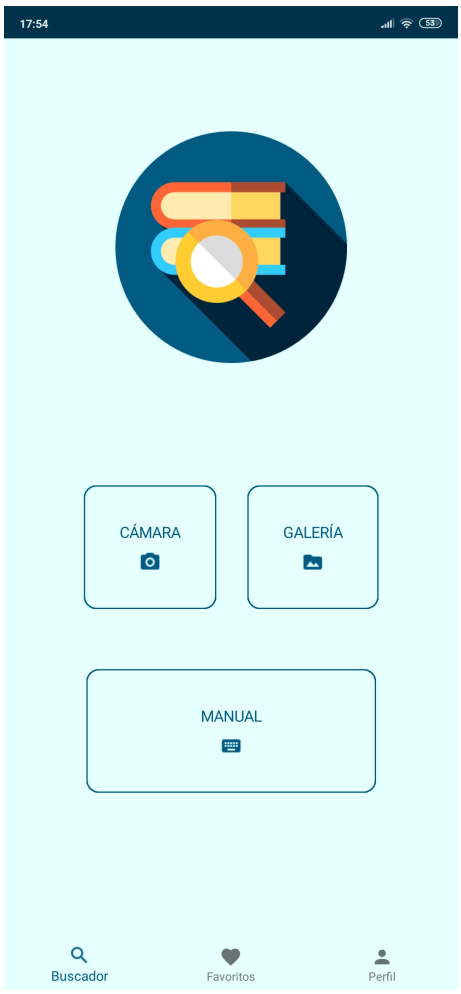


Figura 4.13: Pantalla del Buscador de la aplicación móvil

4.5. Búsqueda de libros manual

La búsqueda de libros manual es una funcionalidad que da un sentido más completo a la aplicación, ofreciendo la posibilidad de realizar una búsqueda introduciendo el título del libro, el autor o ambas. El diseño que sigue esta funcionalidad es similar al de la búsqueda de libros por imagen, aunque en esta ocasión se salta el paso de enviar la imagen a Cloud Vision, directamente realiza la búsqueda en Google Books con los criterios facilitados por el usuario. Su implementación es llevada a cabo en la clase *Formulario* del fichero *Formulario.xaml.cs*. Esta clase habilita dos campos para que el usuario pueda introducir los datos del libro que busca y los formatea en un JSON que se envía al servidor. Se puede apreciar este proceso en la figura 4.14.

```
private void Btn_buscar(object sender, EventArgs e)
{
    if (String.IsNullOrWhiteSpace(titulo.Text) && String.IsNullOrWhiteSpace(autor.Text))
    {
        DisplayAlert("Error", "Introduce un título o un autor.", "Aceptar");
    }
    else
    {
        Navigation.PushAsync(new ResultadoFormulario(titulo.Text, autor.Text));
    }
}
```

Figura 4.14: Búsqueda de libros manual - Aplicación móvil

El servidor recoge el JSON de la solicitud en el fichero *formulario.php* del paquete *books*, extrae los datos de búsqueda y realiza una petición a Google Books de la misma manera que se hace para la búsqueda de libros por imagen, obteniendo de esta consulta una lista de libros coincidentes con los criterios facilitados, que son parseados y devueltos al usuario, comprobando antes, si alguno de los resultados está entre sus favoritos. En la figura 4.15 se puede apreciar una muestra del proceso seguido.

```
$post_vars = json_decode(file_get_contents("php://input"),true);
$idUsuario = $post_vars["IdUsuario"];
$idUsuario = str_replace('"', '', $idUsuario);
$titulo = $post_vars["titulo"];
$autorBuscado = $post_vars["autor"];

$client = new Google_Client();
$client->setApplicationName("Client_Library_Examples");
$client->setDeveloperKey("**********************");

$service = new Google_Service_Books($client);
$optParams = array('printType' => 'books');
$results = $service->volumes->listVolumes($titulo . " " . $autorBuscado, $optParams);
```

Figura 4.15: Búsqueda de libros manual - Servidor

En la figura 4.16 se aprecia el formulario de entrada para este tipo de búsqueda.

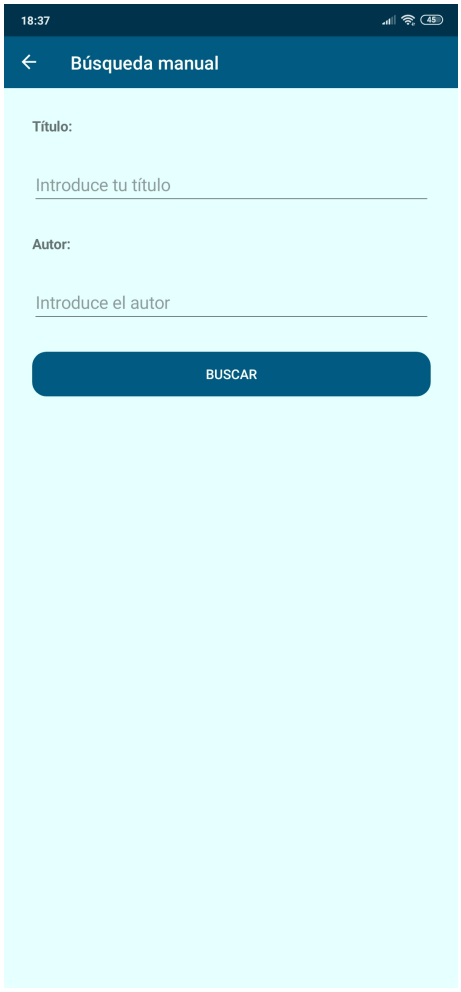


Figura 4.16: Pantalla del Buscador manual de la aplicación móvil

4.6. Búsqueda de libros por autor

Este apartado es una funcionalidad añadida para los casos en los que el usuario esté visualizando un libro en concreto y quiera conocer, de manera rápida, mas obras de ese autor. Mientras se muestran los detalles del libro en cuestión, el usuario podrá pulsar sobre el nombre del autor, que se muestra en modo de enlace, y automáticamente se realiza una búsqueda de libros que se corresponden con el autor seleccionado y se muestran los resultados en una lista de la misma manera que ocurre con el resto de búsquedas. Por lo tanto, esta función sigue el mismo patrón habitual de recopilar los datos que proporciona el usuario, incorporarlos a un objeto JSON y enviarlos al servidor. Este proceso se produce en la clase *BuscarAutor* del fichero *BuscarAutor.xaml.cs*, pudiéndose ver parte del mismo en la figura 4.17. De nuevo, el objeto JSON incorpora el ID del usuario para poder realizar la búsqueda entre sus favoritos.

El servidor recoge esta información en el fichero *buscarAutor.php* del paquete *books*. Con el autor a buscar, se lanza una petición a Google Books indicándole expresamente que lo que se buscan son libros de ese autor en concreto, devolviendo

```
1 referencia
public BuscarAutor(string autor)
{
    InitializeComponent();
    Title = autor;
    errorLabel.IsVisible = false;
    CVAutor.IsVisible = false;
    spinner.IsVisible = true;
    spinner.IsRunning = true;
    if (App.Current.Properties.ContainsKey("Id"))
    {
        PedirAutorServidorAsync(App.Current.Properties["Id"].ToString(), autor);
    }
}
```

Figura 4.17: Búsqueda de libros por autor - Aplicación móvil

de la misma manera que en otras peticiones una lista de resultados con libros, en este caso, del autor solicitado. A continuación, se comprueba si alguno de los libros devueltos son favoritos del usuario y se devuelven a la aplicación en otro objeto JSON. La figura 4.18 muestra cómo se lleva a cabo este proceso.

```
$post_vars = json_decode(file_get_contents("php://input"),true);
$idUsuario = $post_vars["IdUsuario"];
$idUsuario = str_replace('"', '', $idUsuario);
$autorBuscado = 'inauthor:' . $post_vars["Autor"];

$client = new Google_Client();
$client->setApplicationName("Client_Library_Examples");
$client->setDeveloperKey("**************************");

$service = new Google_Service_Books($client);
$optParams = array('printType' => 'books');
$results = $service->volumes->listVolumes($autorBuscado, $optParams);
```

Figura 4.18: Búsqueda de libros por autor - Servidor

4.7. Búsqueda de enlaces de compra

La búsqueda de enlaces de compra para los libros es la segunda característica principal de LiBook y otorga a la aplicación un carácter distintivo que le hace distanciarse del resto de aplicaciones con un funcionamiento similar.

Esta funcionalidad permite conocer distintas opciones de compra para los libros en los que esté interesado el usuario. Cuando consulta un libro en concreto, entre sus datos podrá encontrar enlaces a las principales plataformas de venta online, algunas de ellas especializadas en libros. Al ser seleccionado alguno de ellos, se abrirá dentro de la propia aplicación un navegador que visitará la web de compra del libro en concreto en esa tienda, pudiendo realizar en ella las acciones habituales de compra online. Las tiendas disponibles para la compra de los libros son:

- *Google Play*
- *Amazon*
- *Casa del Libro*
- *Fnac*
- *El Corte Inglés*

Este catálogo de tiendas puede ampliarse a cualquier tienda online.

El procesamiento de esta funcionalidad comienza en la aplicación móvil, concretamente en la clase *InfoLibro* del fichero *InfoLibro.xaml.cs*, cuando el usuario pulsa sobre un libro para consultarlo en detalle. En ese momento, se recoge el título y el autor de ese ejemplar y se envían al servidor, como siempre, a través de un objeto JSON. En la figura 4.19 se puede apreciar la función que lleva este proceso a cabo.

```
1 referencia
public async Task EnviarLibro(string titulo, string autor)
{
    await App.ItemManager.SaveTaskAsync(new ItemLibro(titulo, autor), true);

    //Cuando se recibe el resultado
    spinner.IsVisible = false;
    spinner.IsRunning = false;

    googlePlay.IsVisible = true;
    amazon.IsVisible = true;
    casadellibro.IsVisible = true;
    fnac.IsVisible = true;
    elcorteingles.IsVisible = true;
}
```

Figura 4.19: Búsqueda de enlaces de compra - Aplicación móvil

El servidor recoge estos datos en el fichero *ejecutarSelenium.php* del paquete *selenium*. En él, el título y el autor del libro a buscar se tratan para darles el formato necesario para proceder a su búsqueda. En este punto entra en juego Selenium, la API que hará posible buscar el ejemplar deseado en las diferentes tiendas online. Para ello, se ejecuta desde el propio fichero PHP el script de Python *busquedaGoogle.py* que ejecuta el servidor de Selenium. Esta acción se puede apreciar en la figura 4.20.

Como se ha dicho, Selenium es una API que, dotada de un servidor propio, es capaz de automatizar tareas en un navegador y realizar así de manera automática, procesos, consultas, o demás acciones en un navegador. Para cada navegador existe un pequeño programa que debe estar ejecutándose para realizar las tareas en dicho navegador, aunque para evitar tener que lanzar ese proceso manualmente, también existe la posibilidad de ejecutar Selenium con un servidor standalone, que es la opción ideal para automatizar proceos en proyectos alojados en un servidor. Por lo tanto, esta es la manera en la que este proyecto ejecuta Selenium. Se puede apreciar en la figura 4.21 este servidor en ejecución.

Con el servidor ejecutándose, el fichero Python *busquedaGoogle.py* realiza la búsqueda de las diferentes tienda utilizando la mencionada API de Selenium. Por cada

```php
$post_vars = json_decode(file_get_contents("php://input"),true);
$titulo=$post_vars["Titulo"];
$autor=$post_vars["Autor"];

$libro = $titulo . " " . $autor;
$libro = str_replace(" ", "+", $libro);
$script = 'python busquedaGoogle.py ' . $libro;
$command = escapeshellcmd($script);
$output = shell_exec($command);
echo $output;
```

Figura 4.20: Búsqueda de enlaces de compra - Servidor

```
Microsoft Windows [Versión 10.0.18363.900]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\xampp\htdocs\TFM\selenium>java -jar selenium-server-standalone-3.141.59.jar
22:01:20.235 INFO [GridLauncherV3.parse] - Selenium server version: 3.141.59, revision: e82be7d358
22:01:22.908 INFO [GridLauncherV3.lambda$buildLaunchers$3] - Launching a standalone Selenium Server on port 4444
2020-06-25 22:01:26.490:INFO::main: Logging initialized @25211ms to org.seleniumhq.jetty9.util.log.StdErrLog
22:01:34.496 INFO [WebDriverServlet.<init>] - Initialising WebDriverServlet
22:01:39.026 INFO [SeleniumServer.boot] - Selenium Server is up and running on port 4444
```

Figura 4.21: Selenium - Servidor standalone

una de las tiendas establecidas, se realiza una petición al buscador de Google cuya solicitud incorpora como palabras clave el nombre de la tienda, el título del libro y su autor. Para cada una de las búsquedas, se consultan los resultados obtenidos examinando su URL para comprobar si se tratan realmente de la web en la que se anuncia el ejemplar solicitado o por el contrario el resultado no es el deseado. Cuando se tienen los resultados verificados correspondientes a las páginas web donde se anuncia el libro, se transforman a un objeto JSON que será devuelto a la aplicación para que el usuario pueda acceder a esos enlaces.

La figura 4.22 muestra una pequeña parte en la que se realiza la búsqueda en Google de las diferentes tiendas.

```python
for tienda in tiendas:
    driver.get("https://www.google.co.ve/search?q=" + tienda + palabra)

    result_xpath = "//div[@class='r']/a[@href]"
    WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.XPATH, result_xpath)))

    results = driver.find_elements(By.XPATH, result_xpath)
    links = [result.get_attribute("href") for result in results]
```

Figura 4.22: Selenium - Búsqueda en la web

En la figura 4.23 se aprecia cómo se muestran los enlaces para cada uno de los vendedores.

Figura 4.23: Pantalla con los diferentes enlaces de compra

4.8. Inserción y eliminación de favoritos

La funcionalidades de inserción y eliminación de favoritos comparten diseño e implementación, ya que una operación es la antagónica de la otra. La función que lleva a cabo estas operaciones se encuentra en la clase *InfoLibro* del fichero *InfoLibro.xaml.cs*. Con un libro seleccionado, el usuario puede pulsar sobre el icono del corazón para alternar la condición de favorito del libro, añadiéndolo a ese grupo cuando el icono está seleccionado o eliminándolo del mismo cuando no lo está. Al pulsar el corazón, se consulta cuál era el estado de ese libro y se pasa al contrario, recogiendo el código ISBN, que es el identificador único de cada libro, el ID del usuario y un parámetro que indica si se quiere insertar o eliminar el libro de favoritos para enviárselo al servidor. En la figura 4.24 se puede apreciar el código que se encarga de esta parte en la aplicación móvil.

El servidor captura estos datos en el fichero *putFavoritos.php* del paquete *favoritos*. La inserción o eliminación del libro en la sección de favoritos viene marcado por el valor del parámetro que se envía desde la aplicación. Si la petición es de inserción, se crea el libro en la tabla favoritos de la base de datos, y si hay que eliminarlo, se

```
private async void botonFav_Clicked(object sender, EventArgs e)
{
    string isbn = libro.isbn;
    if ((sender as ImageButton).Source.ToString().Equals("File: corazon_vacio.png"))
    {
        if(isbn == "desconocido")
        {
            await DisplayAlert("Error", "Este libro no se puede guardar en favoritos porque no tiene ISBN", "Aceptar");
        }
        else
        {
            (sender as ImageButton).Source = "corazon_lleno.png";
            (sender as ImageButton).HeightRequest = 80;
            (sender as ImageButton).WidthRequest = 80;
            (sender as ImageButton).MinimumHeightRequest = 80;
            (sender as ImageButton).MinimumWidthRequest = 80;
            ItemPutFavorito putFavorito = new ItemPutFavorito(true, App.Current.Properties["Id"].ToString(), isbn, true);
            await App.ItemManager.SaveTaskAsync(putFavorito, true);
        }

    }
    else
    {
        (sender as ImageButton).Source = "corazon_vacio.png";
        (sender as ImageButton).HeightRequest = 80;
        (sender as ImageButton).WidthRequest = 80;
        (sender as ImageButton).MinimumHeightRequest = 80;
        (sender as ImageButton).MinimumWidthRequest = 80;
        ItemPutFavorito putFavorito = new ItemPutFavorito(false, App.Current.Properties["Id"].ToString(), isbn, true);
        await App.ItemManager.SaveTaskAsync(putFavorito, true);
    }
}
```

Figura 4.24: Favoritos - Aplicación móvil

busca ese libro en la tabla para borrarlo. La figura 4.25 muestra este proceso.

```
$post_vars = json_decode(file_get_contents("php://input"),true);
$insertar=$post_vars["Insertar"];
$idUsuario=$post_vars["IdUsuario"];
$idFavorito=$post_vars["IdFavorito"];

//INSERTAR
if($insertar == true){
    $query = "INSERT INTO Favoritos(IdFavorito, IdUsuario, IdProducto,
     Fecha) VALUES (null, $idUsuario,'$idFavorito', now())";
    if ($resultado = db::getInstance()->dbquery($query))
        echo 0;
    else
        echo -1;
}
//BORRAR
else{
    $query = "DELETE FROM Favoritos WHERE IdProducto = $idFavorito";
    if ($resultado = db::getInstance()->dbquery($query))
        echo 0;
    else
        echo -1;
}

echo json_encode($respuesta);
```

Figura 4.25: Favoritos - Servidor

La figura 4.26 muestra un ejemplo de un libro con el botón de favorito en forma de corazón desmarcado.

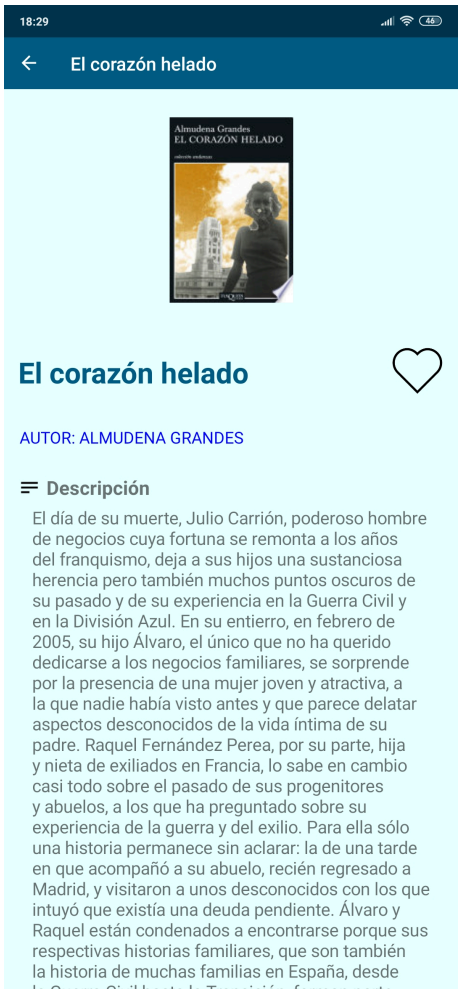


Figura 4.26: Pantalla de un libro que no está marcado como favorito

4.9. Consultar el historial

El historial es una funcionalidad añadida para la aplicación con la que el usuario puede consultar los últimos libros que ha seleccionado. Esta opción se encuentra en el apartado *Perfil* y se accede a ella mediante un botón. Cuando el usuario lo pulsa, se envía una petición al usuario en la que se indica el ID del usuario que solicita saber su historial. Esta función está implementada en la clase *Historial* de la clase *Historial.xaml.cs*, y puede apreciarse en la figura 4.27.

El servidor se encarga de esta funcionalidad en el fichero *historial.php* del paquete *books*. Cuando recibe la petición, recoge el ID del usuario y consulta en la base de datos cuáles son los últimos libros que ha visitado ese usuario, recopilándolos y devolviéndolos a la aplicación móvil mediante un objeto JSON, haciendo, al igual que en el resto de funcionalidades que devuelven una lista de libros, la comprobación de si alguno de ellos forma parte de la sección de favoritos de ese usuario. La figura 4.28 muestra parte del código que realiza esta función.

```
1 referencia
public async Task PedirHistorialServidorAsync(string id)
{
    ItemHistorial historial = new ItemHistorial(id);
    await App.ItemManager.SaveTaskAsync(historial, true);

    if (App.Current.Resources.ContainsKey("historial"))
    {
```

Figura 4.27: Historial - Aplicación móvil

```
$query = "SELECT * FROM visitaslibros NATURAL JOIN libros WHERE
 visitaslibros.IdUsuario = $idUsuario ORDER BY fecha DESC";
if($resultado = db::getInstance()->get_result($query)){
    $favoritos = null;
    $queryHis = "SELECT IdProducto FROM favoritos WHERE IdUsuario = $idUsuario";
    if($resultadoHis = db::getInstance()->get_result($queryHis)) {
        $favoritos = array();
        while ($filaHis = $resultadoHis->fetch_assoc()) {
            array_push($favoritos, $filaHis["IdProducto"]);
        }
    }
    while ($fila = $resultado->fetch_assoc()){
        $autores = explode(", ", $fila["autores"]);
        $resultado_aux = array(
            "titulo" => $fila["titulo"],
            "autores" => $autores,
            "descripcion" => $fila["descripcion"],
```

Figura 4.28: Historial - Servidor

Se puede apreciar un ejemplo de historial en la figura 4.29.

Figura 4.29: Pantalla del historial en la aplicación móvil

Capítulo 5

Conclusiones y Trabajo Futuro

Tras la finalización del proyecto, una buena manera de evaluar el trabajo realizado y poder sacar unas conclusiones correctas es echar la vista atrás fijándose en la primera idea que se tenía del proyecto, cuáles eran sus objetivos y adónde se quería llegar, y compararlo con el resultado final haciendo una valoración de si los objetivos que se propusieron se han alcanzado.

La idea original consistía en hacer una aplicación que ofreciera a los usuarios la posibilidad de consultar y comparar precios de libros de una manera sencilla, rápida y práctica. Técnicamente, se buscaba conseguir este funcionamiento haciendo uso de las tecnologías existentes e investigando sobre diferentes servicios que aportaran valor y que pudieran unirse en una misma arquitectura.

Finalmente, se puede decir que se han logrado los objetivos propuestos. El resultado final del proyecto es una aplicación que contiene todas las funcionalidades pensadas desde el inicio, además de alguna función añadida que se incorporó durante el desarrollo. Y en lo relativo al desarrollo técnico, se ha logrado investigar sobre tecnologías novedosas como son las API, comunicar servicios muy diferentes, y hacer que todos ellos trabajen en armonía dando unidad y solidez a la aplicación. El objetivo principal era desarrollar un único producto que aunara las funcionalidades de diferentes servicios y ofrecerlo a los usuarios para que puedan tener diferentes funcionalidades en un mismo lugar, y este objetivo se ha cumplido.

Concretamente, la aplicación móvil multiplataforma se ha logrado desarrollar usando la plataforma Xamarin de Microsoft, y de los objetivos principales del proyecto, el reconocimiento de imágenes se lleva a cabo haciendo uso de la API Vision de Google Cloud Platform, mientras que la búsqueda de los libros se consigue gracias a la API Google Books. Por último, se ha conseguido ofrecer a los usuarios diferentes enlaces de compra de los libros utilizando la API de Selenium WebDriver, que automatiza búsquedas en Google consultando cuál es el enlace de venta de un libro en concreto en las principales tiendas online.

En lo relativo al trabajo futuro, aun hay aspectos que se pueden mejorar y nuevas funcionalidades que se pueden añadir. En primer lugar, puesto que la máxima prioridad del proyecto era ofrecer algo que ya existía pero otorgándole un valor añadido, se ha incidido muy especialmente en esta parte de experimentación e inno-

vación, lo que ha provocado que algunos detalles muy comunes en la gran mayoría de aplicaciones no se hayan llegado a implementar, como por ejemplo la función de restablecimiento de contraseña en caso de que el usuario la olvide o una interfaz de usuario atractiva y moderna. Un apartado que se incluye en la aplicación pero podría extenderse es el de las estadísticas. Existe un historial donde el usuario puede ver los últimos libros que ha visitado, pero podrían añadirse otros puntos interesantes como el autor más consultado, un registro de los libros que se han descargado, o el número de libros que se han consultado en un periodo de tiempo específico. También podría perfeccionarse la sección de favoritos añadiendo la posibilidad de organizar los libros en colecciones, de tal manera que el usuario pudiera tener colecciones de libros organizadas por géneros, autores, extensión o los que ya ha leído, por ejemplo. Otro aspecto que puede resultar extremadamente interesante, y posiblemente fuese muy bien recibido por los usuarios en este tipo de aplicaciones, son las recomendaciones. Se podría extender la funcionalidad de la aplicación dotando al proyecto de un sistema de aprendizaje automático que recopilara información sobre cuáles son los libros que visita cada usuario, de manera completamente anónima, y pudiera informar a los usuario de libros similares a los que está consultando, otros ejemplares que también visitan otros usuarios con gustos parecidos o cuáles son los las obras más visitadas.

Por último, sería totalmente necesario alojar el servidor del proyecto en un sitio web para que cualquier usuario pueda acceder a él en todo momento, ya que para el desarrollo del mismo el servidor se ha alojado en una máquina local.

Chapter 6

Introduction

For some years now, the world has been going through a process of technological revolution that affects practically all areas of society. There are people who say that the fourth industrial revolution in our history is taking place, the one called Industry 4.0. This is causing substantial change at the business level. Most companies are digitizing their business, going from a manual methodology to a computer one, making practically all processes go through a computer system or depend on the use of the internet. A good example is the film business. Not many years ago, to see a movie, you had to go to a video store and rent it for a few days, while now there are digital platforms where you can instantly access almost any title. This has caused that the cinema in physical format has decreased until it almost disappeared.

Something similar has happened in the world of reading. Libraries and bookstores were the usual place for lovers of reading to find out about the latest published novels or buy books.Currently, to be aware of literary news, it is enough to do an internet search to find all the information you need. When deciding to buy a work, you can find many digital platforms where books are sold in all their formats, whether physical, hardcover, softcover, pocket or digital edition, but it will be necessary to dive between the different stores to find the best option. LiBook, the application developed during this project, makes it easy to compare the different options that a user has to buy a book, in the same way that it makes it much easier to find the title that interests them. Being specialized only in books, there is no risk of mistaking a book for a movie with the same name, or accidentally consulting a wrong book. Its visual interface, as well as the search for books through an image, make it very clear if the book being consulted is the one that is actually being searched.

Libook allows you to photograph the cover of a book and automatically displays all the information related to that book, as well as a preview of it, links to the main digital stores and even the possibility of reading it completely online or downloading it in format PDF or EPUB to read them in an electronic book, in addition to other functions such as adding titles to a favorites list.

6.1. Motivation

The choice of this topic for the development of the Final Master's Project is very marked because reading is one of the most common hobbies. Internet gives access to all the information you need, including the world of literature, but sometimes there may be too much information. It is extremely useful when you want to know, for example, what are the latest trends, what is new from a writer, or when a film adaptation of a book will premiere in the cinema. But if you want to find a website dedicated exclusively to books where you can read a preview or find the places where you can buy it, it may not be so easy, or sometimes you need to read several paragraphs of information that is not relevant until you get to what you are really looking for.

Regarding the development of the project, it has been decided to make a mobile application for both Android and iOS devices with the intention of reaching as many users as possible. Mobile applications are probably the most used tool by a large part of the population to do almost any action, so people are very familiar with this type of program.

On the other hand, the use of APIs grows more and more every day, because it is a simple and effective way to use services and functions already developed and tested by other developers, so it has been decided to use several APIs that allow access to information from an extensive collection of books and offer the functionality to reach those titles through text recognition in images.

6.2. Objectives

The main objective of LiBook is to offer a simple, fast and intuitive book search service that provides the most relevant information about them and offers an added value to traditional search engines. The objectives of the project are specified in the following list:

- Provide a simple book searh engine based on title and author.
- Offer a search engine based on the cover of the book.
- Show a list with the possible results to choose the exact book to which the user refers.
- Consult relevant information of the book, such as the title, author, description, publication date, publisher, number of pages or its ISBN.
- Offer the possibility of reading a preview or the full book online.
- Download the book in digital format.
- See purchase links for the book from the most common sellers.
- Have a favorites section.

6.3. Workplan

To carry out the project, the following planning has been followed:

- Search for information on the state of the art: at the beginning an investigation was made of applications or services similar to this project, such as specialized book apps or price comparators. Later, different ways of accessing book collections or some API that offered information about them were consulted, as well as other APIs that allowed recognizing text in images.
- Specification of requirements: in this phase, all the functionalities of the project were going to be detailed and how they were going to be carried out.
- Choice of APIs for various functionalities: at this point, several of the APIs found during the state of the art research process were evaluated and the best options for the book search and text recognition in images functionalities were determined.
- Design and development of a mobile application: during this phase, an exploration of the different options for the creation of a multiplatform mobile application was carried out and its development was carried out.
- Evaluation and testing of the application: finally, the resulting software was subjected to tests to correct errors and verify the correct operation of the application.

Figure 6.1 shows a schedule with the planning of the different activities carried out for the development of the project.

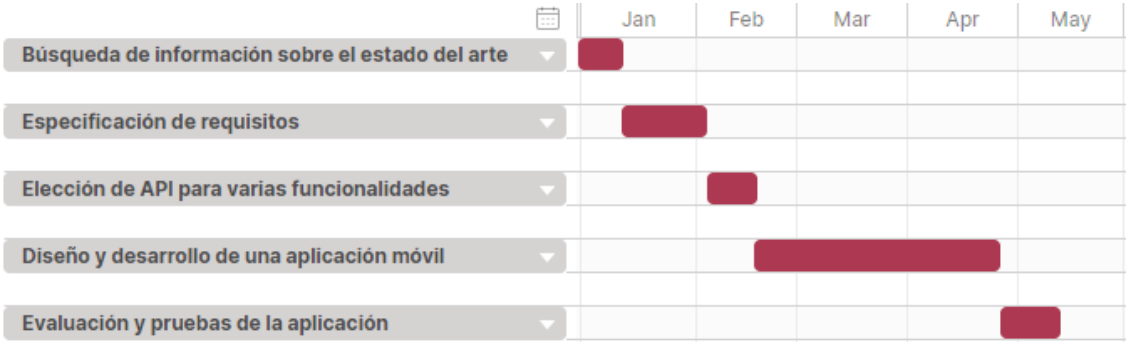


Figure 6.1: Workplan

Chapter 7

Conclusions and Future Work

After the completion of the project, a good way to evaluate the work done and to be able to draw correct conclusions is to look back, looking at the first idea you had of the project, what its objectives were and where you wanted to get to, and compare it with the final result making an assessment of whether the objectives that have been set have been achieved.

The original idea was to make an application that offered users the possibility of consulting and comparing book prices in a simple, quick and practical way. Technically, the aim was to achieve this performance by making use of existing technologies and researching different services that added value and could be united in the same architecture.

Finally, it can be said that the proposed objectives have been achieved. The final result of the project is an application that contains all the functionalities thought from the beginning, in addition to some added function that was incorporated during development.And regarding technical development, research has been carried out on innovative technologies such as APIs, communicating very different services, and making them all work in harmony, giving unity and strength to the application. The main objective was to develop a single product that would combine the functionalities of different services and offer it to users so that they can have different functionalities in one place, and this objective has been met.

Specifically, the multiplatform mobile application has been developed using Microsoft's Xamarin platform. Of the main objectives of the project, image recognition is carried out using the Google API Vision, while searching for books is achieved thanks to the Google Books API. Finally, it has been possible to offer users different book purchase links using the Selenium WebDriver API, which automates Google searches by consulting the sale link of a specific book in the main online stores.

Regarding future work, there are still aspects that can be improved and new functionalities that can be added. In the first place, the highest priority of the project was to offer something that already existed but giving it added value, for this reason, it has had a very special impact on this part of experimentation and innovation, which has caused some very common details in the vast majority applications have not been implemented, such as the password reset function in case the user forgets

it or an attractive and modern user interface. A section that is included in the application but could be extended is that of statistics. There is a history where the user can see the last books they have visited, but other interesting points could be added such as the most consulted author, a record of the books that have been downloaded, or the number of books that have been consulted in a period of specific time. The bookmarks section could also be improved by adding the possibility of organizing the books into collections, so that the user could have book collections organized by genres, authors, extension or those he has already read, for example. Another aspect that can be extremely interesting, and possibly was very well received by users in this type of applications, are the recommendations. The functionality of the application could be extended by providing the project with a machine learning system that would collect information about which books each user visits, completely anonymously, and could inform users of books similar to the ones they are consulting, other copies that other users with similar tastes also visit or which are the most visited books.

Finally, it is absolutely necessary to host the project server on a website so that any user can access it at all times, because for its development the server has been hosted on a local machine.

Bibliografía

[1] Encuesta del Instituto Apigee sobre las API - http://pages.apigee.com/rs/apigee/images/Apigee%20Institute%20Survey%20Report.pdf

[2] Google Cloud Platform - https://cloud.google.com/

[3] Twitter Developers - https://developer.twitter.com/en

[4] Amazon Web Services - https://aws.amazon.com/es/

[5] Amazon API Gateway - https://aws.amazon.com/es/api-gateway/

[6] Google Cloud Vision - https://cloud.google.com/vision?hl=es

[7] Azure Computer Vision - https://azure.microsoft.com/es-es/services/cognitive-services/computer-vision/

[8] Amazon Rekognition - https://aws.amazon.com/es/rekognition/

[9] Github de Tesseract OCR - https://github.com/tesseract-ocr/tesseract

[10] Quiero Libros Aplicación Web - https://www.quierolibros.com/

[11] Quiero Libros Aplicación Móvil - https://play.google.com/store/apps/details?id=com.quierolibros&hl=es

[12] My Library Aplicación Móvil - https://play.google.com/store/apps/details?id=com.vgm.mylibrary&hl=es

[13] Buscador de Google Books - https://books.google.es/

[14] API de Google Books - https://developers.google.com/books

[15] Google Shopping - https://www.google.es/shopping?hl=es_ES

[16] idealo - https://www.idealo.es/

[17] Kelkoo - https://www.kelkoo.es/

[18] ASOS Web - https://www.asos.com/es/

[19] ASOS Aplicación Móvil - https://play.google.com/store/apps/details?id=com.asos.app&hl=es

[20] Arquitectura Cliente-Servidor - https://www.ecured.cu/Arquitectura_Cliente_Servidor

[21] Bases de Datos Relacionales - https://www.oracle.com/es/database/what-is-a-relational-database/#:~:text=Las%20bases%20de%20datos%20relacionales,una%20ID%20%C3%BAnica%2C%20llamada%20clave.

[22] MySQL - https://www.mysql.com/

[23] C# - https://docs.microsoft.com/es-es/dotnet/csharp/

[24] PHP - https://www.php.net/

[25] SQL - https://www.w3schools.com/sql/

[26] Selenium WebDriver - https://www.selenium.dev/

[27] Newtonsoft.Json - https://www.newtonsoft.com/json

[28] PHPMailer - https://github.com/PHPMailer/PHPMailer

[29] AES Encryption - https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

[30] Microsoft Visual Studio - https://visualstudio.microsoft.com/es/

[31] Xamarin - https://dotnet.microsoft.com/apps/xamarin

[32] PHPStorm - https://www.jetbrains.com/es-es/phpstorm/

[33] XAMPP - https://www.apachefriends.org/es/index.html

[34] JSON - https://www.json.org/json-en.html

Apéndice A

Manual de usuario

En el Apéndice A se detalla un manual de usuario en el que se explican todas las funcionalidades de la aplicación, se muestra la pantalla asociada a cada una de ella y se explica cómo hacer uso de las mismas. Esta descripción se realiza sobre la aplicación Android, aunque la aplicación también está disponible para dispositivos iOS.

En primer lugar, al abrir la aplicación el usuario visualizará una pantalla con el logo de LiBook, visible en la figura A.1, que se mantiene visible durante los pocos segundos que dura la inicialización de la aplicación y las conexiones iniciales que se realizan con el servidor.

A.1. Inicio de sesión y Registro

Una vez que la aplicación está iniciada, el usuario verá la pantalla de inicio de sesión, donde podrá iniciar sesión o registrarse si es la primera vez que usa la aplicación. Para ello, deberá pulsar en el enlace que se muestra en la parte inferior y que le llevará a la pantalla de registro. Ambas pantallas se pueden observar en la figura A.2.

En la pantalla de Registro deberá introducir su email y una contraseña, que se le pedirá que repita, y que debe tener al menos 8 caracteres. Si alguna de las dos condiciones no se cumple, se le mostrará un mensaje de error, que se pueden apreciar en la figura A.3. Si el registro se ha efectuado correctamente, se le notifica mediante un mensaje que se le ha enviado un correo electrónico desde el que podrá activar su usuario para poder iniciar sesión.

Una vez que el usuario se ha registrado, podrá volver a la pantalla de Inicio de sesión pulsando en la flecha que aparece en la esquina superior izquierda o el botón Atrás del sistema, y deberá introducir sus credenciales para iniciar la sesión. En este punto, pueden surgir dos posibles errores: que alguno de los dos campos a introducir estén vacíos, o que el usuario no esté verificado. Se pueden apreciar ambos errores en la figura A.4.

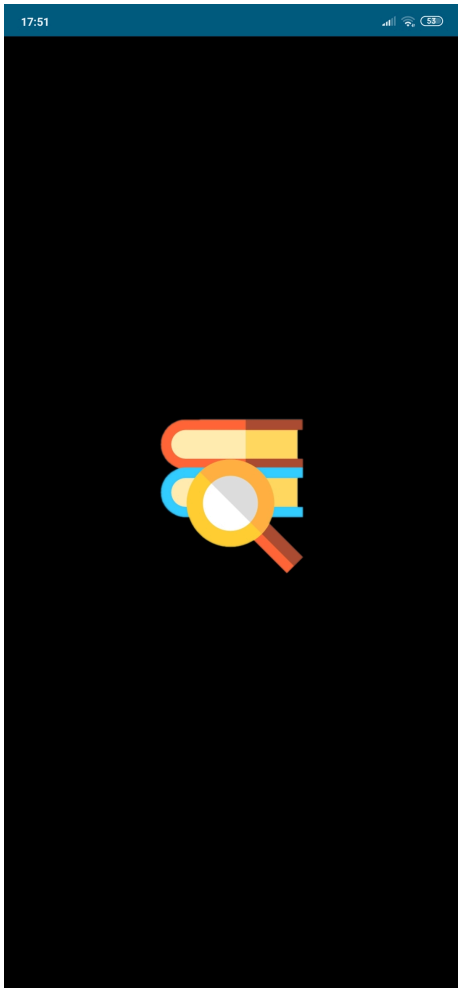
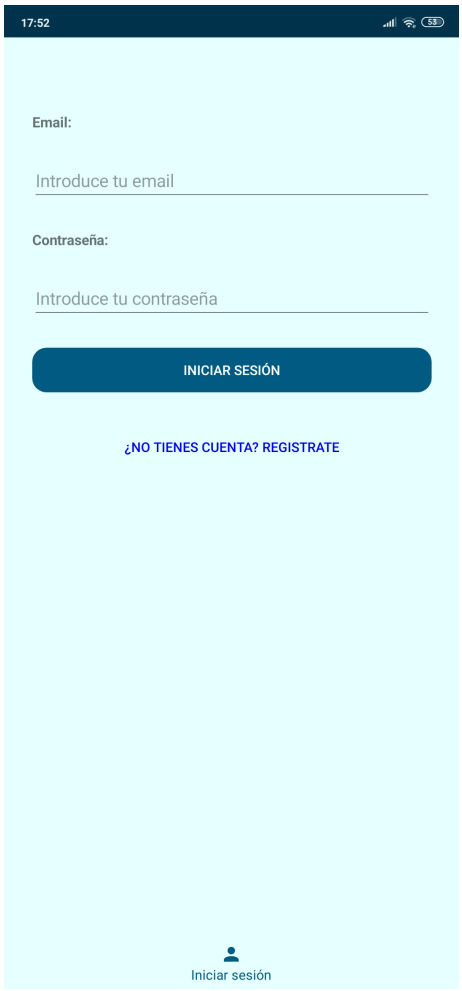
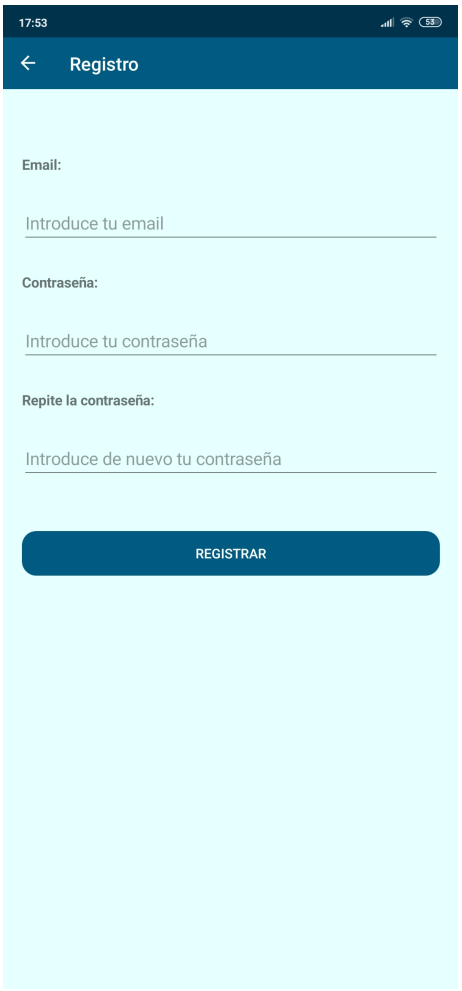


Figura A.1: Pantalla inicial de la aplicación

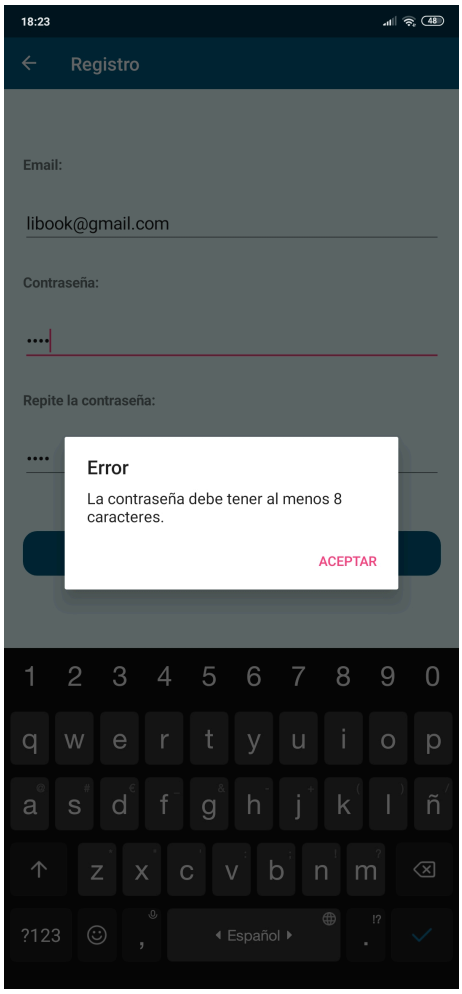


(a) Inicio de sesión

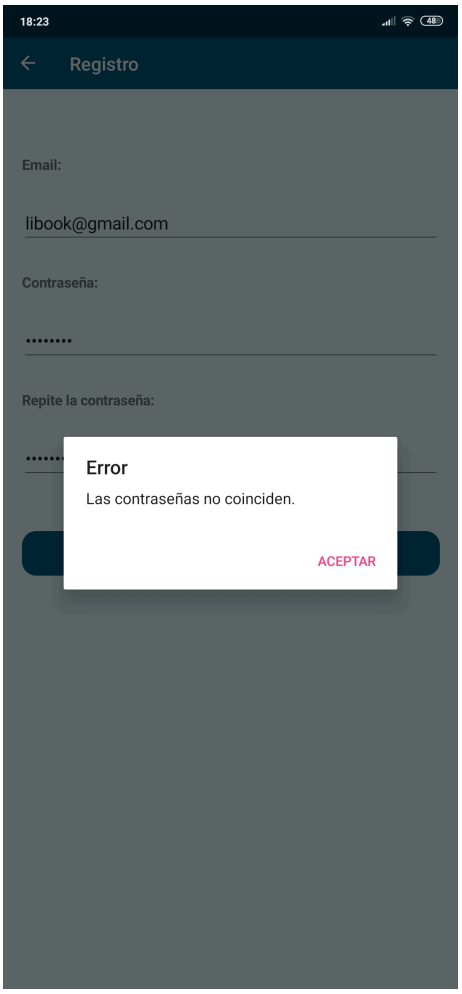


(b) Registro

Figura A.2: Inicio de sesión y Registro

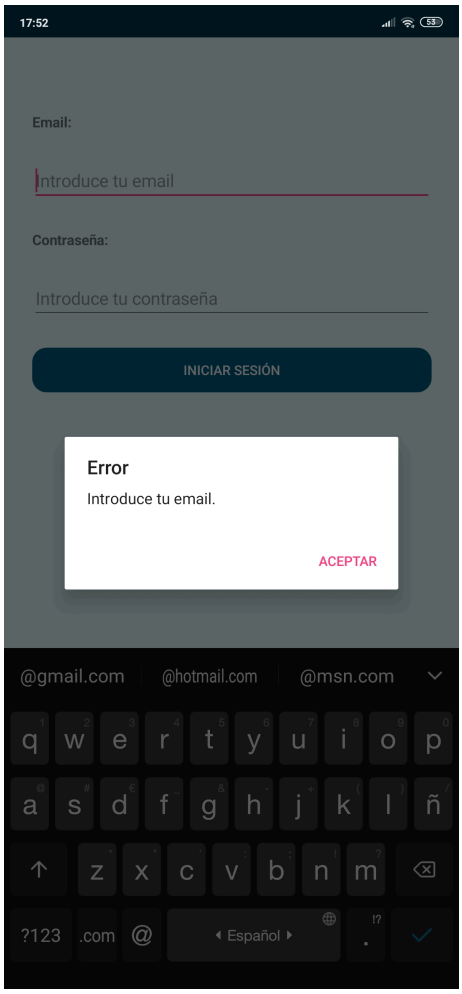


(a) Error Registro 1

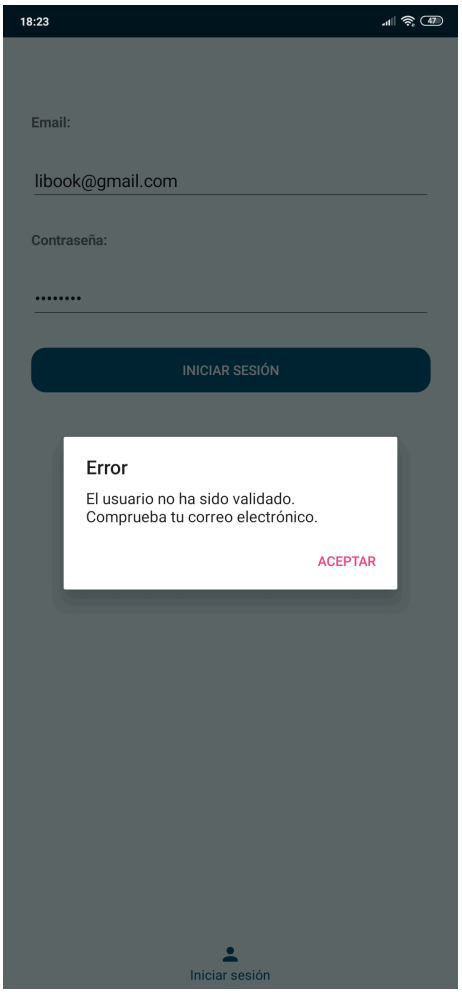


(b) Error Registro 2

Figura A.3: Posibles errores en el Registro



(a) Error Inicio de Sesión 1



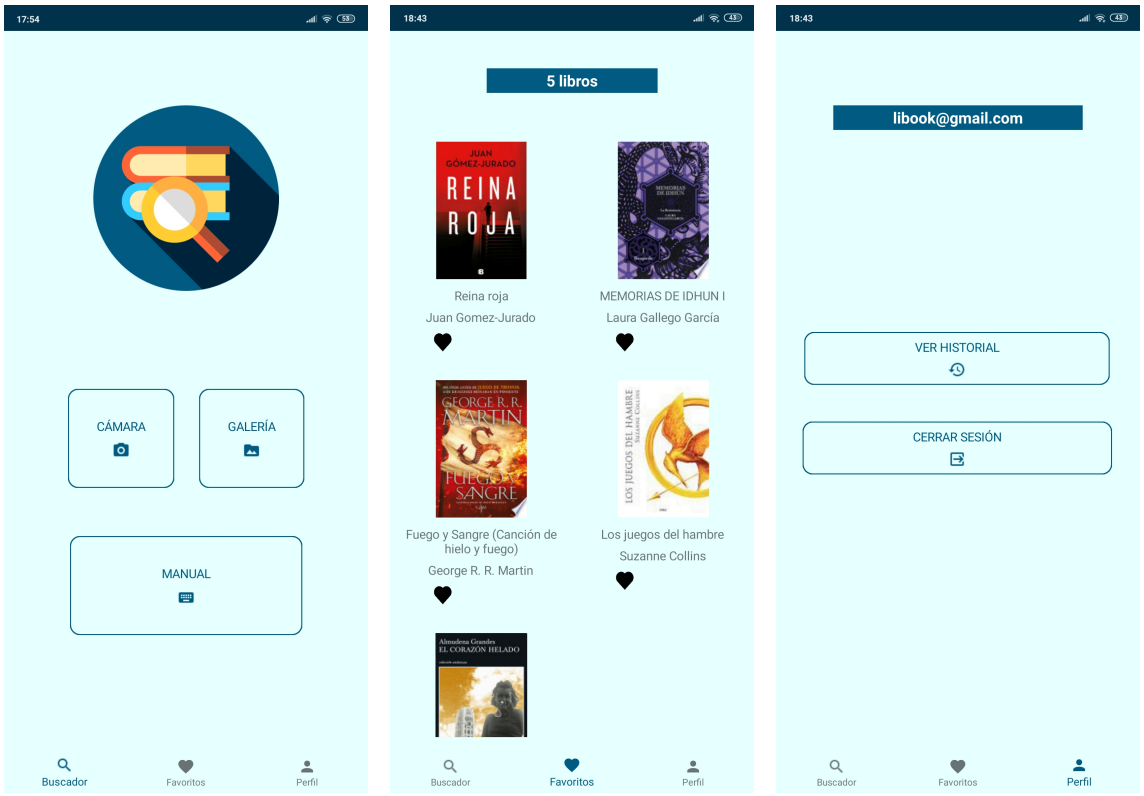
(b) Error Inicio de Sesión 2

Figura A.4: Posibles errores en el Inicio de Sesión

A.2. Búsqueda por imagen

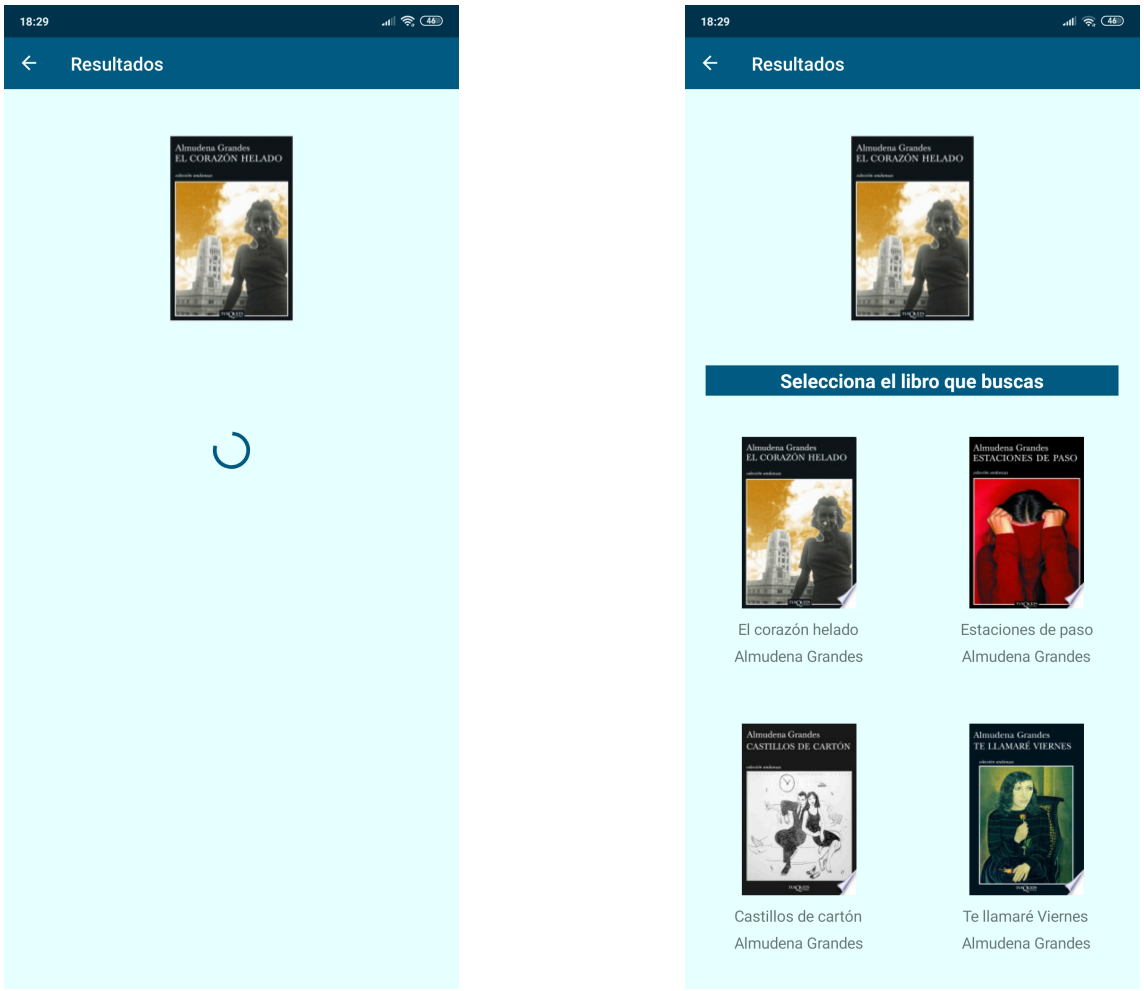
Al iniciar sesión, la aplicación mostrará la pestaña Buscador, aunque podrá apreciar dos pestañas más que se corresponden con la sección de favoritos y de perfil. Podrá navegar entre ellas pulsando sobre el nombre de la que quiera visitar o deslizando el dedo por la pantalla. La figura A.5 muestra las tres pestañas de las que dispone la aplicación.

En la primera de las pestañas se encuentran las tres opciones de búsqueda de libros, que puede ser mediante una imagen, tomada con la cámara o desde la galería del dispositivo, o escribiendo el título o el autor del libro manualmente. Para realizar una búsqueda por imagen, el usuario deberá pulsar sobre una de las dos opciones disponibles y hacer o elegir la foto. A continuación, se mostrará un *spinner* que indica que se está procesando la petición y rápidamente aparecerán los diferentes resultados de la búsqueda. La figura A.6 representa este proceso. Cuando haya aparecido la lista de resultados, el usuario podrá seleccionar el libro que deseé para verlo en detalle.



(a) Buscador (b) Favoritos (c) Perfil

Figura A.5: Pestañas de la aplicación



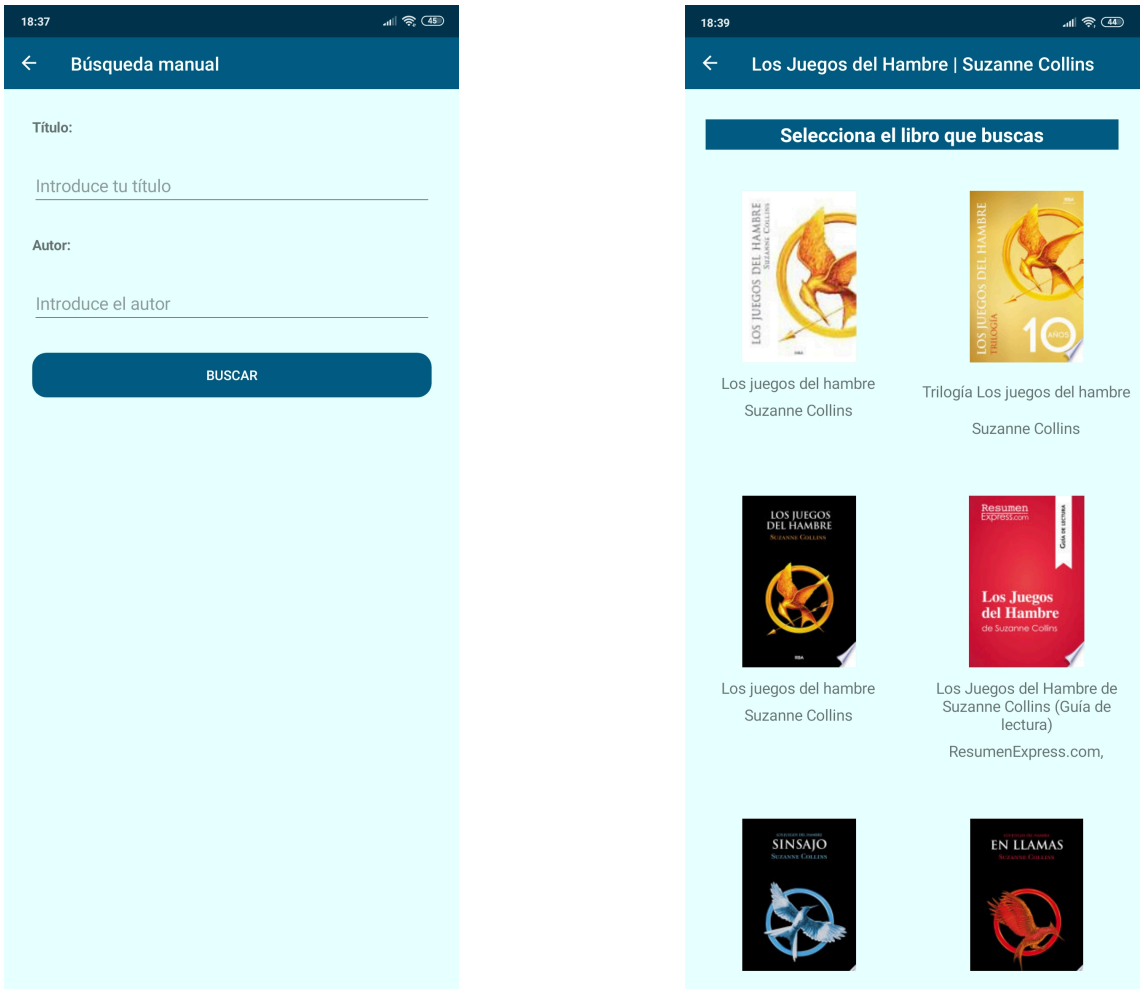
(a) Cargando los resultados

(b) Lista de resultados

Figura A.6: Búsqueda de libros por imagen

A.3. Búsqueda manual

La otra opción de búsqueda de libros es la manual, en la que el usuario podrá introducir en un formulario los datos relativos al libro que busca. Podrá elegir entre indicar únicamente el libro, el autor, o ambos. Una vez introducidos, deberá pulsar en el botón que indica *Buscar* y de nuevo, aparecerá un *spinner* que indica que se está realizando la búsqueda y, posteriormente, la lista con los resultados, tal y como se observa en la figura A.7. En la pantalla de los resultados, se puede apreciar cómo en la barra superior se indican cuáles han sido los parámetros de la búsqueda.



(a) Formulario de la búsqueda

(b) Lista de resultados

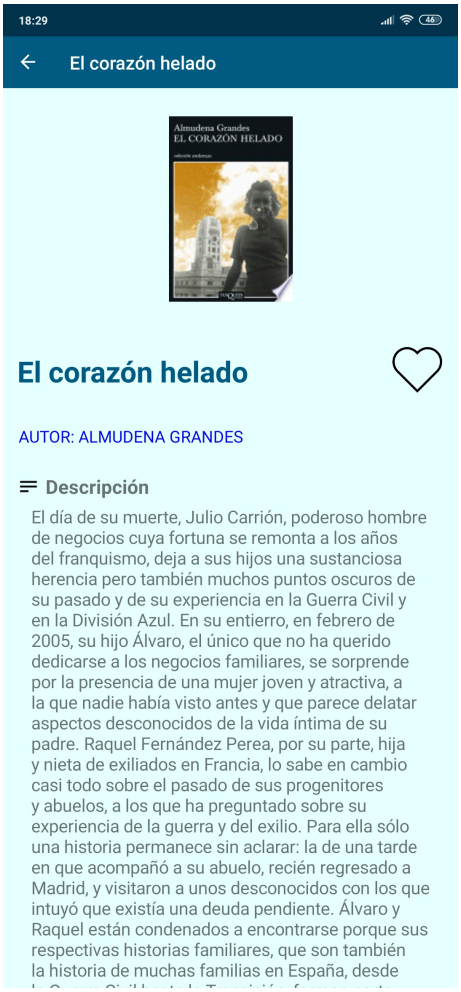
Figura A.7: Búsqueda de libros manual

A.4. Consulta de un libro

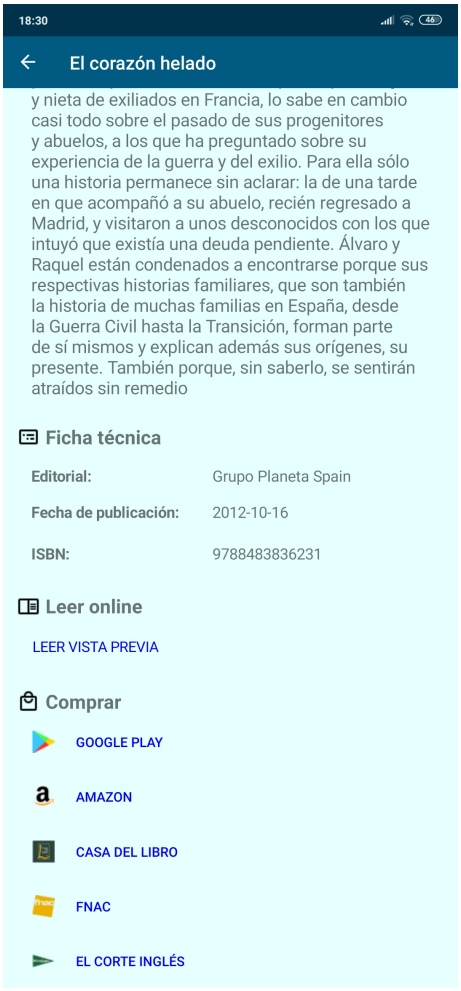
Cuando el usuario se encuentra ante una lista de libros, ya sean los resultados de una búsqueda, la lista de sus libros favoritos, o el historial, podrá seleccionar cualquiera de ellos para verlo en detalle. Al pulsar un libro, se abrirá una nueva pantalla con las características del mismo. En ella, se puede apreciar el título del libro en la barra superior, una imagen de su portada, el título y el autor, que a su vez se trata de un enlace que realiza una nueva búsqueda sobre ese autor, un botón en forma de corazón para añadir o eliminar el ejemplar de favoritos, la descripción de la obra, una tabla con detalles técnicos como la editorial, la fecha de publicación, su ISBN o el número de páginas, enlaces con opciones de compra de ese ejemplar y la posibilidad de descargarlo en formato PDF o EPUB. Aunque estos son todos los datos que se pueden consultar de un libro, no siempre serán visibles todos ellos, ya que solo aparecerán aquellos datos con los que cuenta el libro en concreto que se está visitando. La figura A.8 muestra un ejemplo de esta pantalla.

En esta pantalla el usuario puede consultar las diferentes tiendas donde se da la opción de compra de ese ejemplar. Para ello, deberá pulsar en el enlace correspondiente a la plataforma que quiere consultar. Al hacerlo, se abrirá un navegador dentro de la aplicación que mostrará la web indicada. La figura A.9 muestra un ejemplo de estas opciones de compra.

Esta pantalla ofrece otras funcionalidades como la búsqueda de otros libros de este autor, para lo que tendrá que pulsar el nombre del autor de este libro para que aparezca un nueva pantalla con los resultados, o la lectura de una vista previa del libro, o el libro completo si está disponible esta opción. Si esto último ocurre, se indica en la vista de resultados, al igual que se informa si un libro está disponible para su descarga. Cuando lo está, se encuentra la sección debajo de los enlaces para la compra del libro, con un icono en PDF o en EPUB que comenzará la descarga del ejemplar al pulsarlo. Todas estas funcionalidades se pueden apreciar en la figura A.10.

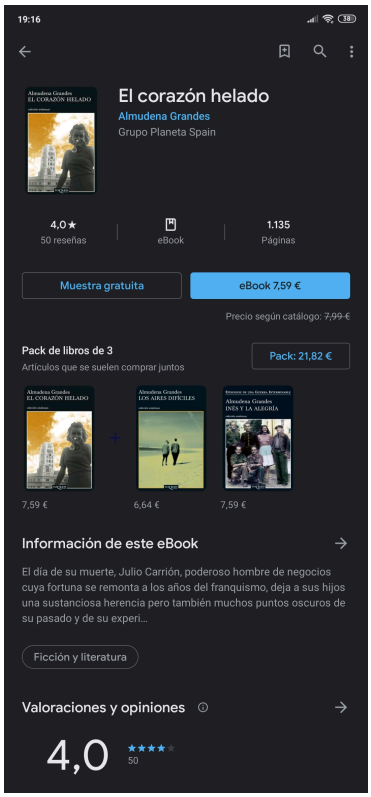


(a) Información de un libro 1



(b) Información de un libro 2

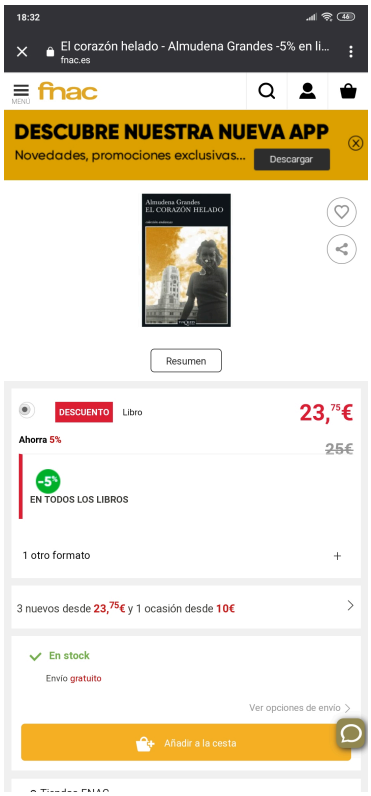
Figura A.8: Pantalla de un libro seleccionado



(a) Google Play

(b) Amazon

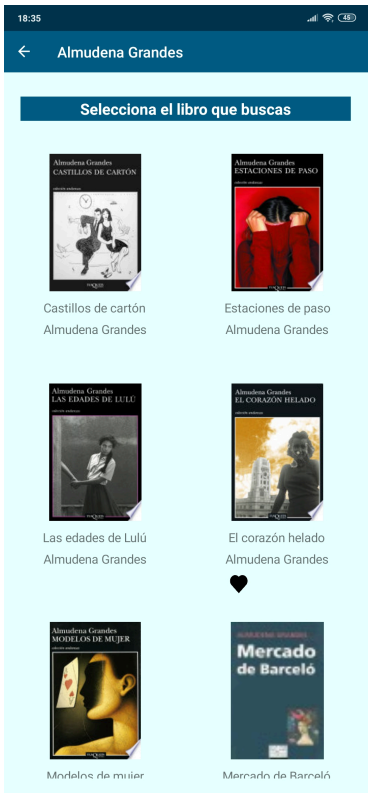
(c) Casa del Libro



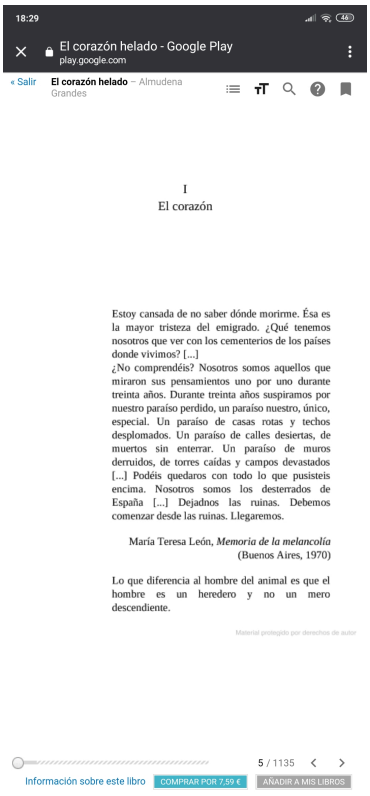
(d) Fnac

(e) El Corte Inglés

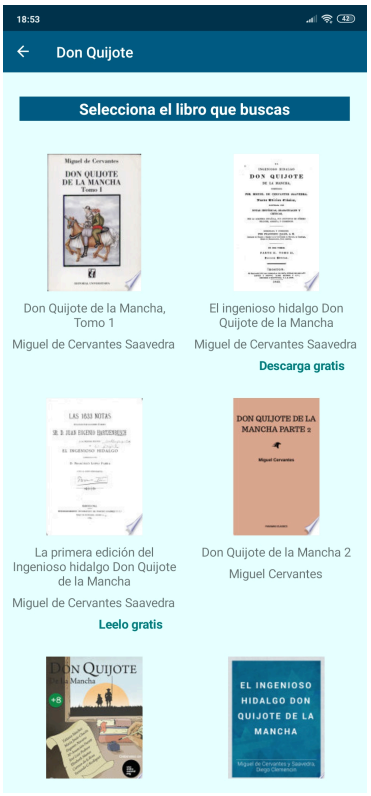
Figura A.9: Opciones de compra para los libros



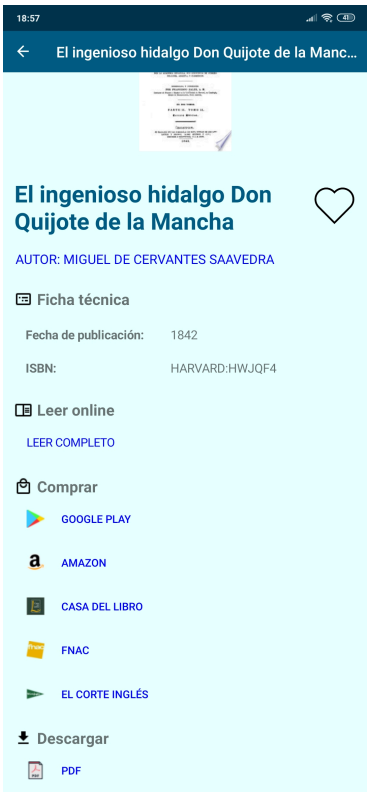
(a) Búsqueda de un libro por su autor



(b) Vista previa de un libro



(c) Información de libro disponible para leer y descargar



(d) Descarga de un libro

Figura A.10: Funcionalidades en la consulta de un libro

A.5. Favoritos

El usuario dispone de una sección de Favoritos, en la que podrá consultar aquellos ejemplares en los que ha marcado el botón en forma de corazón para incluirlos en esta categoría de favoritos. Como se puede ver en la figura A.5, se muestra una lista con los libros que el usuario tiene como favoritos, o un mensaje que indica que aun no ha marcado ningún libro como favorito en caso contrario. En esta lista podrá seleccionar cualquier libro para abrir la pantalla de información de ese libro, la cual se corresponde con la figura A.8, y desde la que podrá hacer todas las funciones habituales, entre las que está desmarcar el libro de favoritos pulsando de nuevo en el icono del corazón.

A.6. Historial

El historial es una de las dos opciones disponibles en la sección de Perfil. Para acceder a él, el usuario deberá pulsar sobre el botón Historial, el cual se puede apreciar en la figura A.5, para visualizar una lista de los últimos libros que ha visualizado, ordenados cronológicamente empezando por el más reciente. En esta lista, de manera similar al resto de listas que se pueden encontrar en la aplicación, el usuario podrá pulsar sobre cualquiera de estos ejemplares para abrir el libro, consultar sus detalles y elegir alguna de sus opciones. La figura A.11 muestra un ejemplo de esta sección.

A.7. Cerrar sesión

La segunda opción que ofrece la sección Perfil es el cierre de sesión, tal y como se muestra en la figura A.5. Al pulsar sobre el botón de cierre de sesión, se borrarán los datos de acceso del usuario en ese dispositivo y aparecerá de nuevo la pantalla de inicio de sesión, que se puede apreciar en la imagen izquierda de la figura A.2. Para volver a iniciar sesión, el usuario deberá introducir de nuevo sus credenciales.

Figura A.11: Historial de un usuario

Apéndice B

Manual de instalación

El Apéndice B explica como se ha de proceder para la instalación del proyecto, que cuenta con dos partes independientes: la instalación del servidor en un entorno local, junto a la base de datos, y la instalación de la aplicación móvil que se deberá realizar de manera manual.

B.1. Repositorio

El código resultante del desarrollo del proyecto puede encontrarse en dos repositorio de GitHub, uno dedicado a la parte del servidor y otro a la aplicación móvil. En el repositorio del servidor se puede encontrar un archivo SQL que contiene la exportación de la base de datos del proyecto. Se pueden encontrar ambos repositorios en estos enlaces:

- **Servidor**: `https://github.com/Ivan8Mt/Servidor`
- **Aplicación móvil**: `https://github.com/Ivan8Mt/TFM`

B.2. Servidor

Durante el desarrollo del proyecto, el servidor se alojó en un entorno local. Para ello se utilizó la plataforma XAMPP, por lo que los pasos que se detallan a continuación serán para realizar una instalación en este entorno.

La parte del servidor del proyecto se encuentra en la carpeta *TFM*, que contiene varios paquetes los cuales incluyen diversos ficheros PHP para cada una de las funcionalidades de la aplicación. El primer paso es instalar XAMPP en la máquina que hará de servidor de la aplicación. Se puede descargar esta plataforma desde su sitio web oficial(33). Con XAMPP instalado, se debe acceder su carpeta de instalación, entrar en la carpeta *htdocs*, y colocar en esa ubicación la carpeta *TFM*. Por otra parte, es necesario importar la base de datos. Para ello, se deberá levantar el servidor Apache y phpMyAdmin de XAMPP, lo cual se puede hacer desde el panel de control de

XAMPP, programa que viene incorporado en su instalación. Con ambas herramientas funcionando, se deberá acceder a la dirección *http://localhost/phpmyadmin/*, donde se encontrará el panel de administración de la base de datos de XAMPP. En el panel lateral, se deberá seleccionar *Nueva*, después en *Importar* y por último *Seleccionar archivo*, eligiendo el fichero *tfm.sql*. Con los archivos PHP ubicados en la carpeta *htdocs* y la base de datos importada, el servidor estará listo.

B.3. Aplicación móvil

La instalación de la aplicación móvil conlleva el uso del IDE Visual Studio, ya que al estar el servidor en un entorno local, es necesario ajustar manualmente la dirección IP donde se encuentra alojado para poder realizar la conexión. Visual Studio se puede descargar e instalar desde su sitio web oficial[(30)]. Una vez instalado, será necesario añadir la plataforma Xamarin, mediante la cual se ha hecho el desarrollo del proyecto. Para su instalación se pueden seguir los pasos que se indican en la https://docs.microsoft.com/es-es/xamarin/get-started/installation/?pivots=windows. Con ambas herramientas correctamente instaladas, se procederá a importar el proyecto.

Nota: este manual sigue los pasos para la instalación de la aplicación en un dispositivo Android. Para instalarla en un dispositivo iOS será indispensable contar con un ordenador y un dispositivo Apple.

Para ello, se deberá abrir Visual Studio y seleccionar del panel lateral derecho la opción *Abrir un proyecto o una solución*, y seleccionar el archivo *TFM.sln* de la carpeta *TFM*. Pasados unos segundos, se abrirá la solución del proyecto donde se puede ver todo el código de la aplicación. Antes de poder ejecutarlo, es necesario instalar algunos paquetes NuGet de los que depende la aplicación. Para ello, se debe abrir el panel *Explorador de soluciones*, pulsar con el botón derecho del ratón sobre el proyecto *TFM* y seleccionar *Administrar paquetes NuGet*. Se abrirá una nueva ventana donde se podrá comprobar qué paquetes están ya instalados e instalar aquellos que falten. La lista completa de los paquetes necesarios es la siguiente:

- Microsoft.CodeAnalysis.FxCopAnalyzers
- Microsoft.CodeQuality.Analyzers
- Microsoft.CSharp
- NETStandard.Library
- Newtonsoft.Json
- Plugin.Multilingual
- Plugin.Permissions
- Rg.Plugins.Popup
- Xam.Plugin.Media
- Xam.TabView

- Xamarin.Auth
- Xamarin.Essentials
- Xamarin.Forms

Con todo ello instalado, el último paso consiste en asignar la dirección IP de la máquina en la que se encuentra alojado el servidor. Este valor se encuentra en una variable llamada *BaseAdress* dentro del fichero *Constants.cs*, ubicado en el paquete *Servidor*. La figura B.1 muestra el contenido de este archivo.

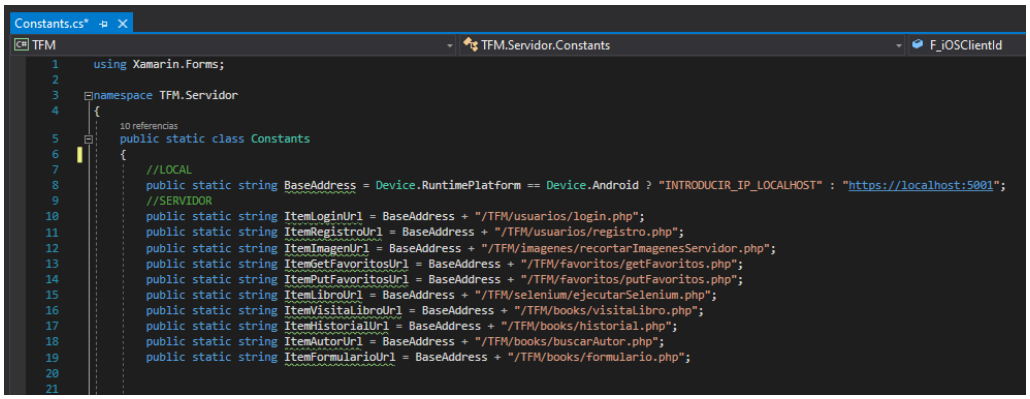


Figura B.1: Archivo donde se asigna la dirección IP del servidor

Tras todos estos pasos, tan solo habrá que conectar al ordenador el dispositivo móvil donde se quiere instalar la aplicación y pulsar en el botón de ejecución que se encuentra en la parte superior de la pantalla para comenzar la instalación de la aplicación en ese dispositivo.