UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación móvil para la gestión de manga en el mercado español

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jesús Enrique Sangonzalo Martínez

Tutor: José Vicente Soler Bayona

2019/2020

Resumen

El manga o cómic japonés constituye una parte importante de la industria editorial de Japón, donde representa más del 25% de todos los materiales impresos en el país. El mercado español lleva activo desde varias décadas y ha batido récords de ventas en los últimos años. A raíz de esta expansión, es cada vez más complicado para el consumidor saber qué licencias están publicadas en España. Por este motivo, este proyecto desarrolla una aplicación móvil para ofrecer al consumidor una visión actualizada del mercado español, es decir, mostrando cada una de las colecciones licenciadas, además de las novedades mensuales publicadas por cada editorial. Asimismo, se ha hecho uso de las nuevas tecnologías para realizar una API REST, la cual está destinada al apoyo de nuevos proyectos ofreciendo su información de manera pública.

Para conseguir esto, se han empleado técnicas de web scraping mediante el uso de la librería BeautifulSoup para la extracción de información de una página especializada en manga. Dicha información es adaptada y almacenada en una base de datos, la cual se comunica con una API REST dónde la información es publicada en formato JSON. Mediante una conexión HTTP, la aplicación móvil realiza una operación GET para obtener los datos y emplearlos en cada una de sus funcionalidades.

Como resultado, se obtiene una aplicación móvil con la información actualizada de la industria del manga en España, además de ofrecer una API REST para apoyar futuros proyectos relacionados con dicha temática.

Palabras clave: Manga, API REST, Python, Web Scraping, Ionic

Abstract

The Japanese manga or comic book is an important part of Japan's publishing industry, representing more than 25% of all printed materials in the country. The Spanish market has been active for several decades and has broken sales records in recent years. As a result of this expansion, it is increasingly difficult for consumers to know which licenses are published in Spain. For this reason, this project is developing a mobile application to offer the consumer an updated view of the Spanish market, i.e. showing each of the licensed collections, in addition to the monthly news published by each publisher. Likewise, new technologies have been used to create an API REST, which is intended to support new projects by offering its information publicly.

To achieve this, web scraping techniques have been employed by using the BeautifulSoup library to extract information from a specialized manga page. This information is adapted and stored in a database, which communicates with a REST API where the information is published in JSON format. Through an HTTP connection, the mobile application performs a GET operation to obtain the data and use it in each of its functionalities.

As a result, a mobile application is obtained with the updated information of manga's industry in Spain, besides offering an API REST to support future projects related to this topic.

Keywords : Manga, API REST, Python, Web Scraping, Ionic

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Manga es un término que engloba una amplia variedad de libros caracterizados principalmente por contener historietas y novelas gráficas originalmente producidas y publicadas en Japón. A diferencia de los cómics americanos o europeos, el manga japonés es casi siempre en blanco y negro. Las impresiones a todo color a menudo sólo se utilizan para lanzamientos especiales como en revistas especializadas o ediciones especiales. Por otro lado, la lectura de estas obras se realiza de derecha a izquierda a diferencia de las publicaciones occidentales. La publicación de manga se realiza generalmente mensual o semanalmente a través de revistas especializadas como la «Weekly Shōnen Jump». Si una serie es lo suficientemente popular, sus capítulos se recogen y se publican en volúmenes denominados «tankōbon». La mayoría de las series de manga son de larga duración y pueden abarcar varios volúmenes (Pagan, Amanga, 2018).

Con casi treinta años de publicación regular de cómic japonés en España, la cantidad de obras publicadas año tras año supone un número considerable en comparación a sus inicios. En el mercado español, ya se cuenta con más de tres mil obras y más de diez mil tomos publicados. Ante un crecimiento de la industria que en cada año bate récords, es complicado para un lector ocasional enterarse por sus propios medios de cada una de las novedades publicadas anualmente (Bernabé, Marc, 2019). Las propias editoriales proporcionan información a través de sus redes sociales o webs, pero se presentan ciertos impedimentos como que cada año surgen nuevas editoriales que promocionan nuevas licencias o que la información se encuentra dispersa, por lo que el lector debe tener un conocimiento muy concreto de la industria y de su funcionamiento.

Actualmente existen ciertas alternativas a este modelo como páginas webs especializadas o aplicaciones móviles. Sin embargo, son muy escasas y en el caso de las apps móviles, no hay apenas variedad. Tampoco se facilita una unificación de toda la información transmitida por las editoriales para hacerla pública y accesible. Esto se traduce en dificultades para posibles desarrolladores interesados en el mercado, y la pérdida de consumidores potenciales.

1.1 Motivación

En los últimos años, tanto la animación japonesa como el manga han influenciado considerablemente a la sociedad occidental. Se ha demostrado que es un medio de consumo muy rentable, dando lugar a novelas, videojuegos, figuras o «merchandising» que gozan de gran popularidad. En el continente europeo, esta expansión ha sido significativa en países como Francia, Italia o Alemania. No obstante, el consumo de manga en España sigue siendo algo minoritario, a pesar del auge obtenido en los últimos años. A medida que las editoriales aumentan sus licencias, es más complicado para el consumidor saber que se publica en su país y con qué regularidad. Por ese motivo, este proyecto tiene la misión de ayudar tanto a nuevos como veteranos consumidores a tener una mejor visión del medio.

Asimismo, se pretende adquirir nuevos conocimientos acerca de las diferentes tecnologías sobre el desarrollo de aplicaciones móviles para posibles proyectos futuros.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación móvil para gestionar de manera unificada y actualizada las colecciones y novedades licenciadas por las editoriales en el mercado español. Para poder cumplirlo, se deberán alcanzar los siguientes subobjetivos:

- Desarrollar el mecanismo para obtener de manera automática los datos bibliográficos de páginas webs especializadas, además de analizar y adaptar los datos para obtener un resultado más acotado.
- Diseñar y desarrollar una base de datos relacional para dotar de persistencia al modelo de datos y almacenar los datos correctamente adaptados previamente por técnicas de web scraping.
- Implementar una API REST para facilitar en formato JSON los datos recogidos en el modelo de datos relacional.
- Conectar todos los subsistemas para establecer transparencia en el sistema resultante y proporcionar al usuario la sensación de estar utilizando un único sistema.

1.3 Estructura de la memoria

El contenido de la memoria comienza por el estado del arte, donde se analizan y se obtienen una serie de conclusiones acerca de diferentes aplicaciones móviles del mercado. En el capítulo siguiente, se establecen los casos de uso y los prototipos preliminares de la aplicación. Posteriormente, se especifica la arquitectura del sistema y un diseño detallado mediante distintos modelos. A continuación, se aborda el desarrollo de los scripts empleados para realizar técnicas de «web scraping», la construcción y población de la base de datos, la implementación de una API REST y una aplicación móvil desarrollada por medio de diversos «frameworks». Finalmente, se presentan los resultados finales y se incluye un apartado de conclusiones junto a mejoras, nuevas funcionalidades o líneas de desarrollo para futuras versiones de la aplicación.

2. Estado del arte

Se pueden encontrar varias aplicaciones móviles destinadas a la gestión de medios literarios en las principales plataformas. Sin embargo, la mayoría basan sus datos en mercados extranjeros y lo que publiquen las editoriales del país en cuestión. Principalmente, tomando como referencia al mercado inglés o americano. Por este motivo, el mercado español a día de hoy no cuenta con casi ningún tipo de variedad en este tipo de herramientas digitales. A continuación, se analizarán varios proyectos similares basados tanto en el mercado español como en el extranjero y, posteriormente, se realizará una priorización de requisitos mediante el método MoSCoW.

2.1 Proyectos similares

2.1.1 Whakoom

2.1.1.1 Descripción

Aplicación móvil dedicada a la gestión de cómics y manga en el mercado español principalmente, aunque también proporciona servicio en otros mercados internacionales como el americano o el japonés. Actualmente, se encuentra disponible en las tiendas digitales de Android y Apple. (*Whakoom*, s. f.)

Figura 1. *Whakoom*

2.1.1.2 Funcionalidades principales

Esta herramienta permite añadir y organizar las diferentes colecciones del usuario, mostrando los tomos que han sido adquiridos y los que le faltan por comprar. También proporciona la opción de poder añadir un cómic o un manga que no esté registrado en su base de datos, a través de un formulario. La aplicación utiliza la cámara trasera para escanear el código de barras de un volumen y poder obtener su ISBN (International Standard Book Number), es decir el identificador único utilizado en libros y similares.

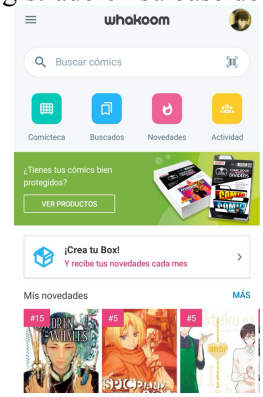

Figura 4. *Pantalla principal de Whakoom*

De igual manera, muestra las novedades más recientes publicadas por las editoriales acompañadas de una fecha lo más concreta posible. También contiene funcionalidades propias de una red social, donde los usuarios pueden compartir su lista de productos deseados, los productos adquiridos recientemente, compartir reseñas de diversas colecciones o puntuar las ediciones por diferentes categorías. Cuentan además con su propia tienda de fundas protectoras para cómics y manga en diferentes tamaños y, de vez en cuando, realizan sorteos entre los usuarios suscritos a su plan «Premium».

2.1.2 Goodreads

2.1.2.1 Descripción

Figura 5. Goodreads

Aplicación móvil destinada a la catalogación de libros y cómics para crear estanterías digitales o listas de lectura por los propios usuarios a partir del propio catálogo del que dispone la herramienta. Fue desarrollada en 2006 por Otis Chandler y actualmente es propiedad de Amazon. Actualmente se encuentra disponible en las principales plataformas digitales de Android y Apple. (*Goodreads*, s. f.)

2.1.2.2 Funcionalidades principales

Goodreads permite al usuario poder interactuar con la comunidad activamente a través de recomendaciones, novedades o libros destacados por otros usuarios. Contiene una función muy interesante dirigida a autores, donde se les permite publicar sus propios libros y mostrar actualizaciones constantes del progreso de sus libros. Asimismo, los usuarios pueden reseñar y puntuar mediante estrellas los libros leídos. Permite clasificar los libros en diferentes categorías en función de si el usuario lo ha terminado, quiere leerlo o está en proceso de finalizarlo. En caso de no poder encontrar un libro en concreto en su base de datos, es posible añadirlo manualmente a través de un formulario.

***Figura 6.** Ventana principal de Goodreads*

Por otro lado, Goodreads cuenta con un algoritmo para poder recomendar libros a los usuarios en función de sus elecciones anteriores. Para obtener estas recomendaciones personalizadas, es necesario haber calificado al menos 20 libros. Se proporciona también la opción de adquirir ciertos productos en tiendas digitales como Amazon, y sincronizar con la página de Facebook del usuario para actualizar sus entradas más recientes.

2.1.3 My Library

2.1.3.1 Descripción

Figura 7. My Library

Aplicación móvil empleada para el almacenamiento de tu librería privada de libros, cómics o videojuegos. La aplicación además de ser gratuita, está libre de cualquier tipo de publicidad. Está traducida en muchos idiomas y está disponible solamente en las plataformas digitales de Android. (*My Library*, s. f.)

2.1.3.2 Funcionalidades principales

My Library comparte muchas funcionalidades con las anteriores aplicaciones descritas como poder añadir libros, cómics o videojuegos escaneando el código de barras, y así poder obtener el ISBN. También permite hacer búsquedas a través de dicho atributo o añadirlo manualmente en caso de no encontrarlo. Asimismo, muestra a los usuarios una serie de estadísticas de manera gráfica como el número de libros leídos o el número de páginas leídas. A diferencia de las anteriores aplicaciones, no permite almacenar los datos de sus usuarios en su sistema, no obstante, proporciona la opción de exportar los datos al propio dispositivo, a Google Drive o a Dropbox.

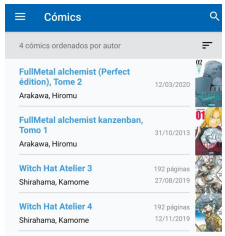

***Figura 8.** Ventana principal de My Library*

2.2 Conclusiones

Tabla 1. Priorización de requisitos con método MoSCoW

	Debe tener	**Debería tener**	**Podría tener**	**No tendrá**
Whakoom	-Listado de colecciones adquiridas -Novedades mensuales -Filtro de búsqueda por título, autor, editorial -Listado de lecturas pendientes o colecciones deseadas	-Filtro de búsqueda mediante el escaneo de códigos de barras -Perfil de usuario -Notificaciones -Estadísticas de lo leído a lo largo del año	-Añadir volúmenes manualmente	-Vender productos -Red social
Goodreads	-Listado de volúmenes leídos o adquiridos -Listado de lecturas pendientes o colecciones deseadas -Filtro de búsqueda por título o autor	-Objetivos de lectura al año -Recomenda-ciones personalizadas -Filtro de búsqueda mediante el escaneo de códigos de barras -Perfil de usuario -Notificaciones	-Añadir volúmenes manualmente -Cambio de idioma	-Red Social -Informar sobre premios o noticias del medio
My Library	-Listado de volúmenes leídos o adquiridos -Listado de lecturas pendientes o colecciones deseadas -Filtro de búsqueda por título o autor	-Filtro de búsqueda mediante el escaneo de códigos de barras -Estadísticas de lo leído a lo largo del año	-Añadir volúmenes manualmente -Cambio de idioma	-Secciones diferentes al manga -Exportar los datos a un servicio externo o al dispositivo

2.3 Propuesta

Tomando como referencia el método MoSCoW, el cual nos muestra de más a menos prioridad las características que se deberían de incorporar a este proyecto. Aunque todas las aplicaciones analizadas están enfocadas a la gestión de medios literarios, algunos requisitos no son afines a los objetivos finales del proyecto.

El objetivo principal del proyecto es ofrecer una herramienta para gestionar colecciones o volúmenes de manga, además de información sobre las novedades de cada editorial. Para hacerlo sencillo, es necesario evitar recargar la aplicación con características innecesarias por el momento. Eso elimina la posibilidad de establecer funcionalidades propias de una red social, además de mostrar noticias sobre el medio. Estas funcionalidades se podrían realizar en una versión siguiente de la aplicación.

3. Especificación

En este apartado, se describen las diversas capas que componen la arquitectura del sistema desde un punto de vista funcional.

En primer lugar, se comentará la capa de negocio donde residen los programas que se ejecutan, seguida de la capa de presentación que representa la interfaz gráfica y para finalizar la capa de persistencia donde se almacenan los datos.

3.1 Capa de negocio

En la capa de negocio se presentarán los diferentes casos de uso que describirán las diversas funcionalidades de la aplicación.

3.1.1 Iniciar sesión

Tabla 2. Caso de uso 1

Caso de Uso 1	Iniciar sesión en la aplicación.
Objetivo	Iniciar sesión en la aplicación.
Descripción	El usuario realiza el inicio de sesión a través de correo electrónico y contraseña convencionales.
Precondición	El usuario se haya registrado con anterioridad.
Secuencia Normal	1. Acceder a la ventana de inicio de sesión 2. Entrar en la aplicación
Postcondición	El usuario ha iniciado sesión en la aplicación y puede realizar diversas acciones a partir del menú principal.

3.1.2 Registrarse

Tabla 3. Caso de uso 2

Caso de Uso 2	Registrarse en la aplicación.
Objetivo	Registrarse en la aplicación.
Descripción	El usuario realiza el registro de cuenta a través de correo electrónico y contraseña convencionales.
Precondición	No haber iniciado sesión anteriormente.
Secuencia Normal	1. Elegir opción de registro para una cuenta nueva. 2. Entrar en la aplicación
Postcondición	El usuario ha sido registrado en la base de datos, y ya puede iniciar sesión en la aplicación.

3.1.3 Ver las novedades mensuales

Tabla 4. Caso de uso 3

Caso de Uso 3	Ver las novedades mensuales.
Objetivo	Ver las novedades mensuales publicadas por cada editorial.
Descripción	El usuario observa una lista ordenada por fechas de estreno, cada una de las novedades que se van a publicar en un futuro.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder a la ventana principal. 2. Pulsar el botón de «Novedades».
Postcondición	Ninguna.

3.1.4 Ver detalles de una colección

Tabla 5. Caso de uso 4

Caso de Uso 4	Ver detalles sobre una colección.
Objetivo	Ver detalles sobre una colección publicada por una editorial.
Descripción	El usuario observa detalles concretos sobre una colección.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder a la ventana principal a. Buscar la colección mediante el filtro de búsqueda. b. Seleccionar la colección. 2. Pulsar el botón «Mi Biblioteca» a. Pulsar el botón «Colecciones» b. Seleccionar una colección
Postcondición	Ninguna.

3.1.5 Ver detalles de un volumen

Tabla 6. Caso de uso 5

Caso de Uso 4	Ver detalles sobre un volumen.
Objetivo	Ver detalles sobre un volumen concreto de una colección.
Descripción	El usuario observa detalles concretos sobre un volumen perteneciente a una colección.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder a la ventana principal a. Buscar la colección mediante el filtro de búsqueda. b. Seleccionar la colección. c. Seleccionar la imagen del volumen 2. Pulsar el botón «Mi Biblioteca» a. Pulsar el botón «Tomos» b. Seleccionar un volumen
Postcondición	Ninguna.

3.1.6 Añadir un tomo adquirido o leído de una colección

Tabla 7. Caso de uso 6

Caso de Uso 5	Añadir un tomo adquirido o leído de una colección.
Objetivo	Añadir un tomo de una colección en la sección de tomos adquiridos en la biblioteca personal del usuario.
Descripción	El usuario añade un tomo de una colección a su biblioteca personal.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder a la ventana principal 2. Buscar la colección donde reside el tomo mediante el filtro de búsqueda. 3. Seleccionar la colección. a. Pulsar en la lista con todos los tomos. i. Pulsar sobre el botón «+» para añadir un tomo. b. Pulsar sobre un tomo concreto. i. Pulsar sobre el botón «+ Lo tengo!» para añadir el tomo.
Postcondición	El volumen se añadirá a la base de datos y se asociará al usuario que haya iniciado sesión.

3.1.7 Añadir un tomo pendiente de una colección

Tabla 8. Caso de uso 7

Caso de Uso 6	Añadir un tomo pendiente de una colección.
Objetivo	Añadir un tomo de una colección en la sección de volúmenes pendientes en la biblioteca personal del usuario.
Descripción	El usuario añade un tomo pendiente de adquirir o leer de una colección a su biblioteca personal.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder a la ventana principal 2. Buscar la colección donde reside el tomo mediante el filtro de búsqueda. 3. Seleccionar la colección. a. Pulsar sobre un tomo concreto. i. Pulsar sobre el botón «+ Lo quiero!» para añadir el tomo.
Postcondición	El volumen se añadirá a la base de datos y se asociará al usuario que haya iniciado sesión.

3.1.8 Ver tu lista de volúmenes deseados o novedades añadidas

Tabla 9. Caso de uso 8

Caso de Uso 7	Ver tu lista de volúmenes deseados o novedades añadidas
Objetivo	Ver tu lista de volúmenes deseados o novedades añadidas en tu biblioteca personal.
Descripción	El usuario observa la lista completa de los volúmenes que desea adquirir en un futuro que ha añadido previamente.
Precondición	Haber iniciado sesión con anterioridad y haber añadido volúmenes para adquirir en un futuro.
Secuencia Normal	1. Acceder a la ventana principal. 2. Pulsar el botón Mi biblioteca a. Consultar los volúmenes pendientes por adquirir en la sección Pendientes.
Postcondición	Ninguna.

3.1.9 Establecer un desafío de lectura anual

Tabla 10. Caso de uso 9

Caso de Uso 8	Establecer un desafío de lectura anual.
Objetivo	Establecer un desafío de lectura anual.
Descripción	El usuario introduce un número de volúmenes que debería leer a lo largo del año, a medida que se vaya leyendo los volúmenes, se verán reflejados mediante una barra de progreso en el menú de la parte izquierda de la aplicación.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder al menú de la parte izquierda de la aplicación. 2. Pulsar el botón «Reading Challenge». 3. Establecer un número de volúmenes a leer durante el año e ir viendo el progreso.
Postcondición	Ninguna.

3.1.10 Cerrar sesión

Tabla 11. Caso de uso 10

Caso de Uso 9	Cerrar sesión
Objetivo	Cerrar sesión de la aplicación.
Descripción	El usuario presiona el botón de Cerrar sesión para desloguearse de la aplicación y volver a la pantalla de inicio de sesión.
Precondición	Haber iniciado sesión con anterioridad.
Secuencia Normal	1. Acceder al menú de la parte izquierda de la aplicación. 2. Pulsar el botón «Cerrar Sesión».
Postcondición	Volver a la ventana de inicio de sesión.

3.2 Capa de presentación

En este apartado, se presentan los diferentes «mockups» que muestran visualmente los casos de uso establecidos en el punto anterior. De esta manera, se le presenta al usuario el estado previo y conceptual del proyecto para entender mejor cada una de las funcionalidades.

3.2.1 Casos de Uso 1 y 2: Iniciar sesión y Registrarse

Figura 7. Mockup Caso de Uso 1

Figura 8. Mockup Caso de Uso 2

3.2.2 Caso de Uso 3: Ver las novedades mensuales

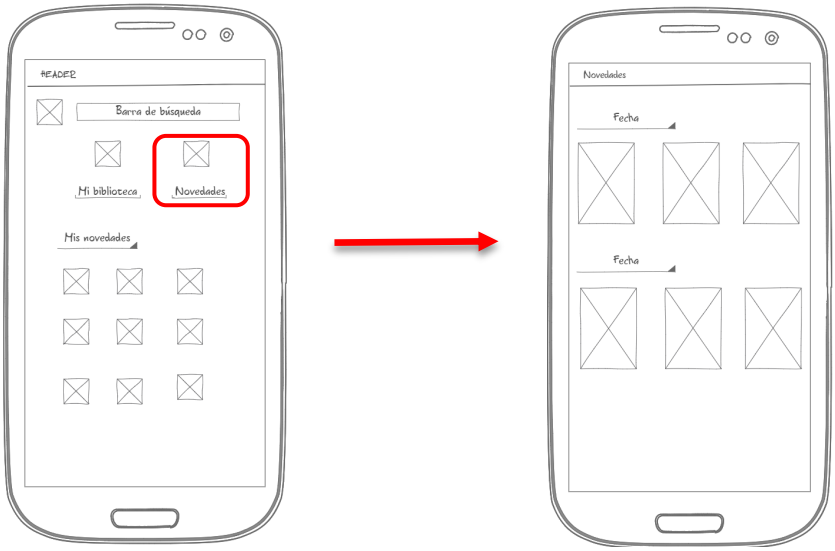

Figura 9. Mockup Caso de Uso 3

3.2.3 Caso de Uso 4: Ver detalles de una colección

En este caso, hay dos maneras diferentes de ver los detalles correspondientes a una colección, por lo tanto, se van a representar dos flujos:

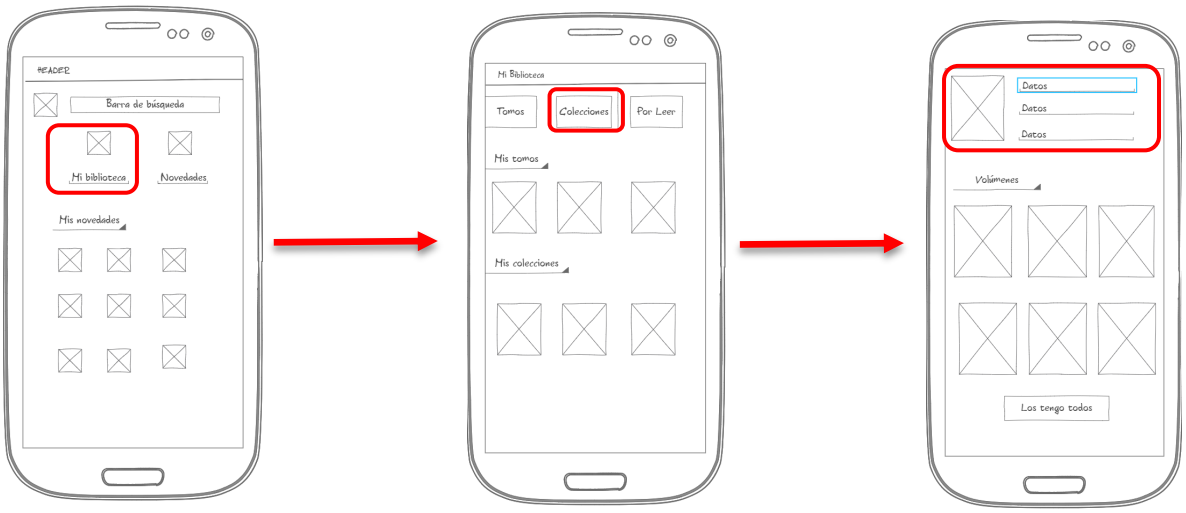

Figura 10. Flujo 1 Mockup Caso de Uso 4

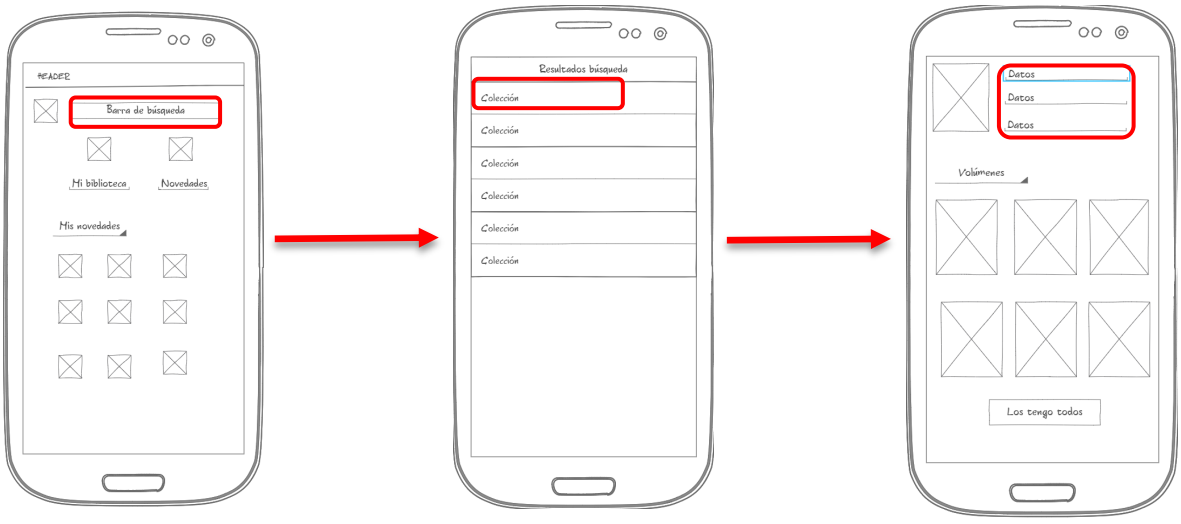

Figura 11. Flujo 2 Mockup Caso de Uso 4

3.2.4 Caso de Uso 5: Ver detalles de un volumen

En este caso, hay dos maneras diferentes de ver los detalles correspondientes a un volumen de una colección, por lo tanto, se van a representar dos flujos:

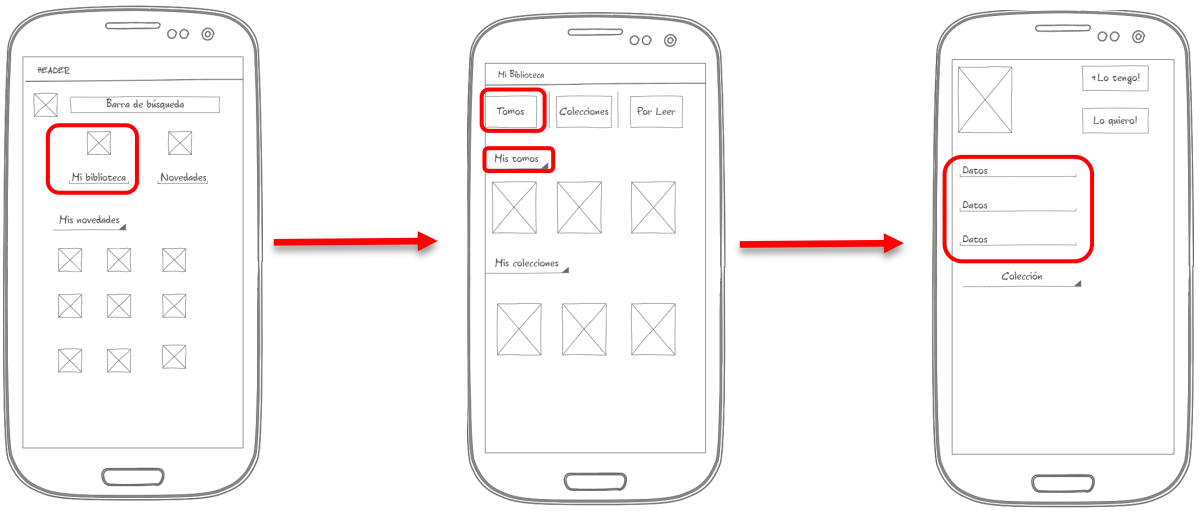

Figura 12. Flujo 1 Mockup Caso de Uso 5

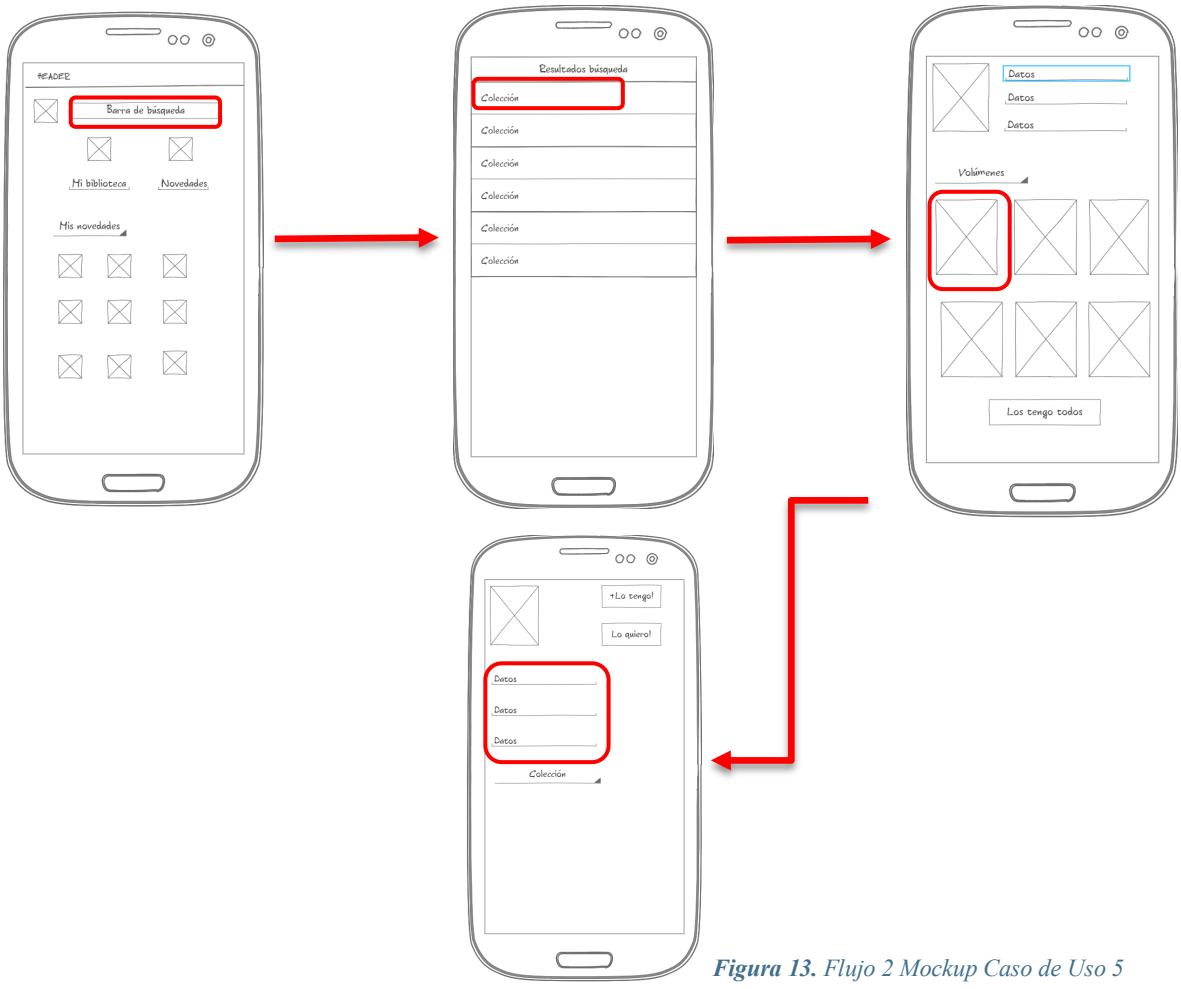

Figura 13. Flujo 2 Mockup Caso de Uso 5

3.2.5 Caso de Uso 6 y 7: Añadir un tomo adquirido o pendiente de una colección

Este caso es muy similar al caso de uso anterior en cuanto a los pasos a seguir para añadir un volumen a la biblioteca del usuario. Aunque se podrían representar dos flujos, se emplea uno solamente para evitar posibles redundancias. En la última ventana, se marca con un recuadro verde el botón para añadir un volumen pendiente a la biblioteca, y en color rojo para añadir un volumen adquirido a la biblioteca:

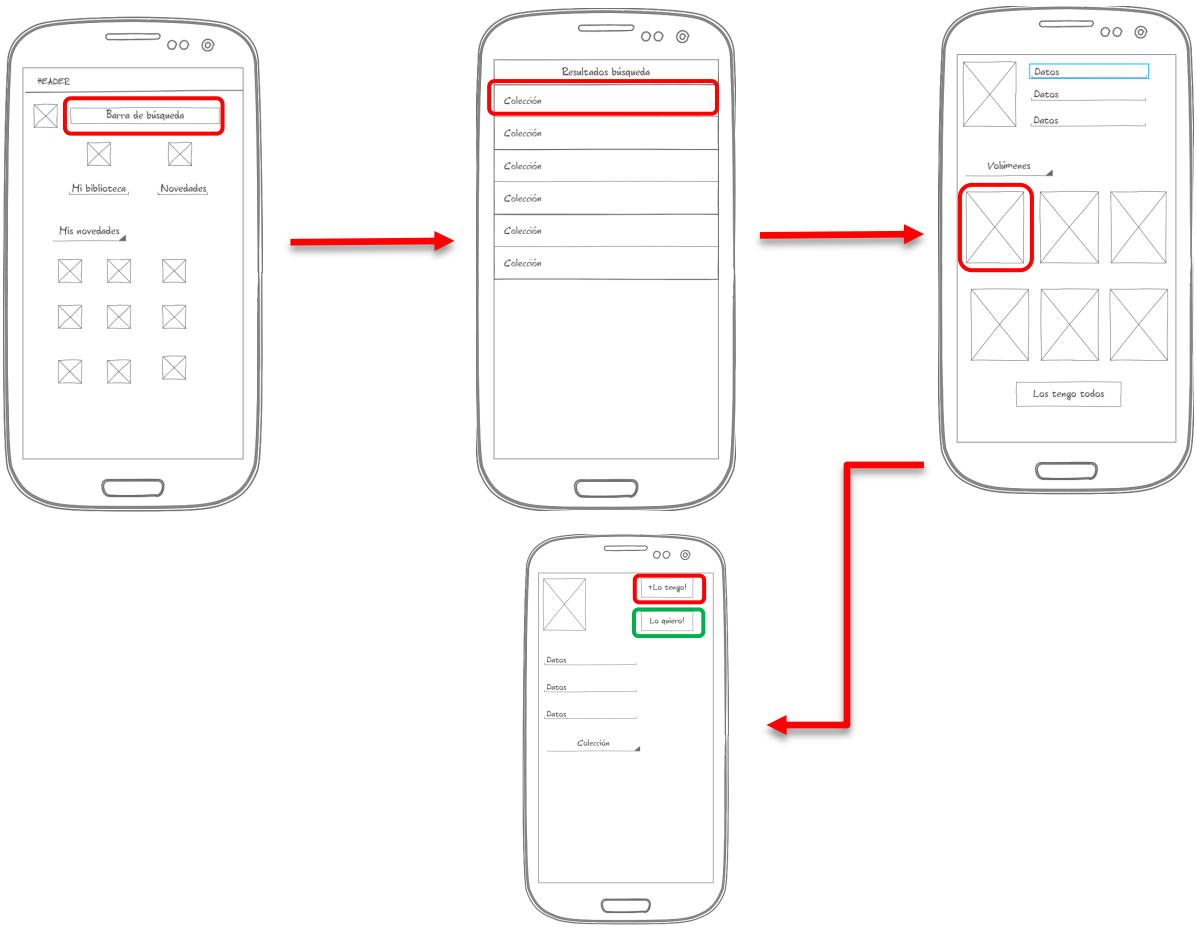

Figura 14. Mockup Casos de Uso 6 y 7

3.2.6 Caso de Uso 8: Ver tu lista de volúmenes deseados o novedades añadidas

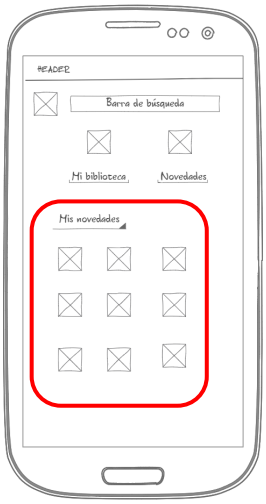

Figura 15. Mockup Caso de Uso 8

3.2.7 Caso de Uso 9: Establecer un desafío de lectura anual

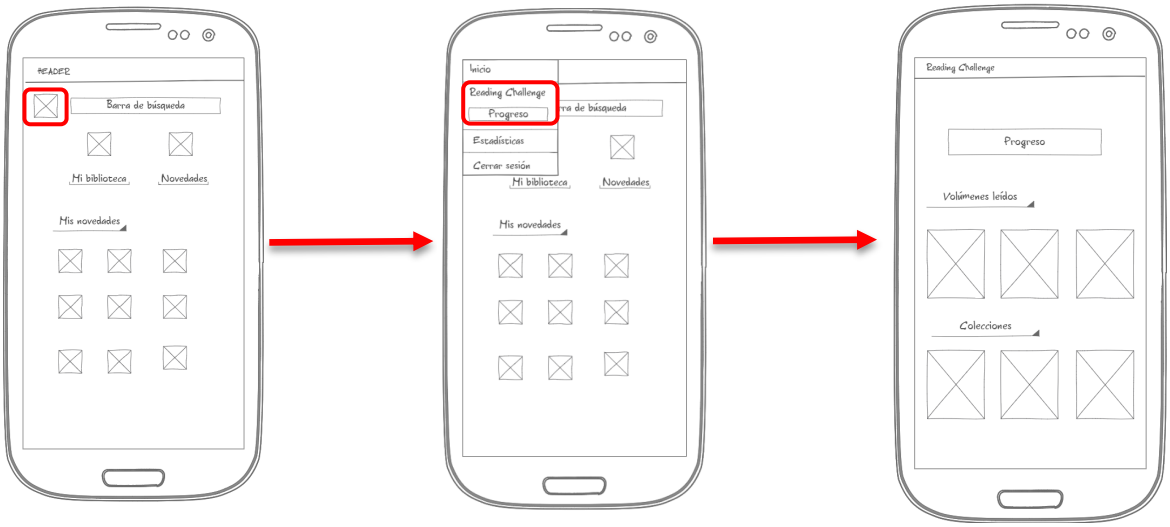

Figura 16. Mockup Caso de Uso 9

3.2.8 Caso de Uso 10: Cerrar Sesión

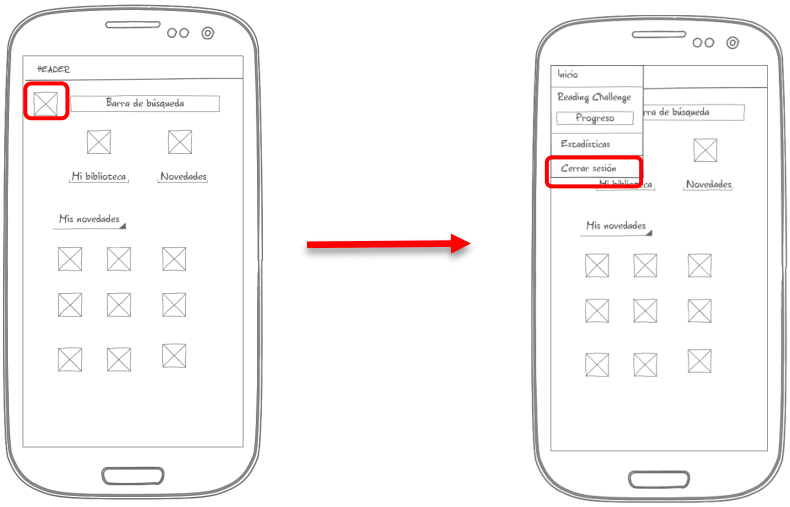

Figura 17. *Mockup Caso de Uso 10*

3.3 Capa de persistencia

En este apartado se explicará cómo se han administrado los datos de la aplicación a través de un diagrama de clases y el modelo entidad-relación. Los diagramas de clases son diagramas estructurales del lenguaje unificado de modelado o UML («Unified Modeling Language»). Este lenguaje de modelado visual es un estándar ISO para visualizar los sistemas de la programación orientada a objetos, aunque también permite visualizar procesos de negocio. Con ayuda de elementos gráficos, UML muestra los estados de los sistemas y describe las interacciones entre los elementos que lo componen. A continuación, se presenta el diagrama de clases con el objetivo de representar las diversas interacciones establecidas entre clases:

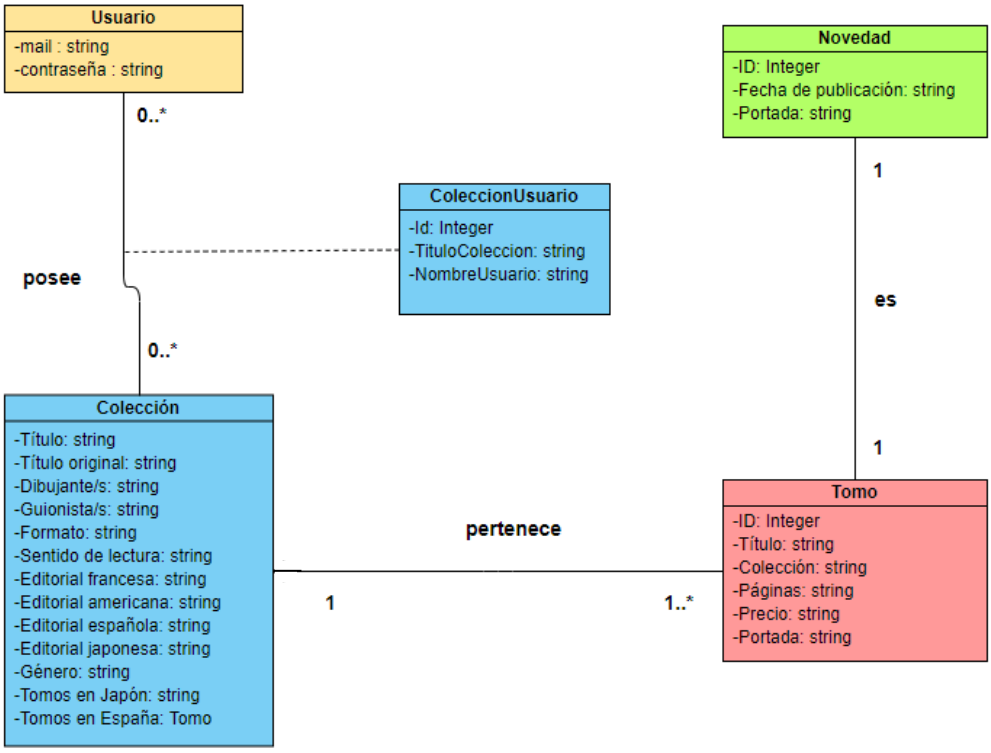

Figura 18. *Diagrama de Clases*

El modelo entidad-relación se representa mediante diagramas y se compone de los siguientes elementos (Gutiérrez, Pedro, 2013):

- **Entidad:** Representa objetos o conceptos basados en la realidad que nos rodea.
- **Atributos:** Definen características y proporcionan información acerca de las entidades.
- **Relaciones:** Vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.
- **Claves:** Atributo de una entidad que se distingue del resto de atributos, ya que se le añade una restricción que le diferencia del resto de atributos.

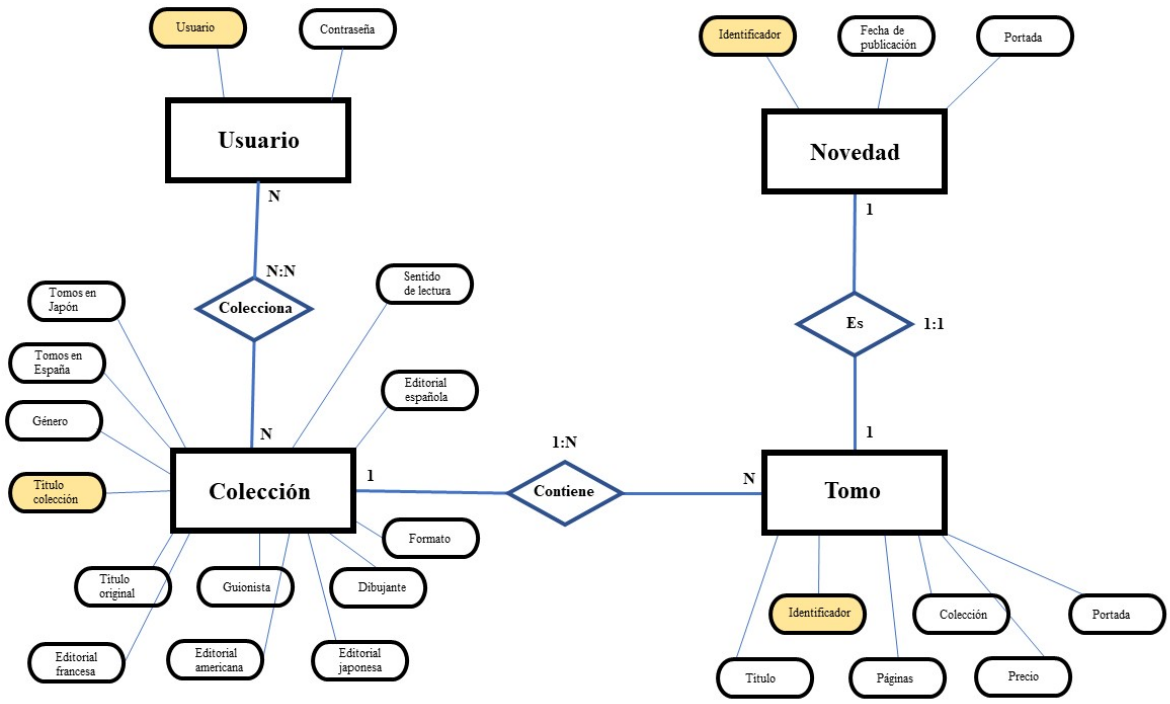

Figura 19. Modelo entidad-relación

El diagrama de entidad-relación se compone de cuatro entidades bien diferenciadas: Colección, Usuario, Novedad y Tomo. Cada entidad contiene varios atributos que definen características propias, sin embargo, hay atributos señalados con un color diferente al resto. Esto se debe a que son claves primarias, es decir, restricciones para identificar de manera única un solo atributo de la entidad, no permitiendo que se repita en ningún caso.

Entre las entidades, se establecen una serie de relaciones para definir ciertas dependencias entre sí. Cada relación está representada por un rombo, además de indicar unas cardinalidades en sus extremos. Esto complementa a las representaciones de las relaciones, especificando cuantos objetos pueden intervenir en dicha relación. Como ejemplo, tomemos la relación establecida entre las entidades Colección y Tomo: la cardinalidad representada es de 1: N, es decir, un registro de Colección puede estar relacionado con varios registros de Tomos. Una colección puede contener varios tomos, pero un tomo sólo puede estar en una colección.

4. Diseño de la solución

4.1 Arquitectura del Sistema

La arquitectura software establecida en este proyecto está esquematizada de la siguiente manera:

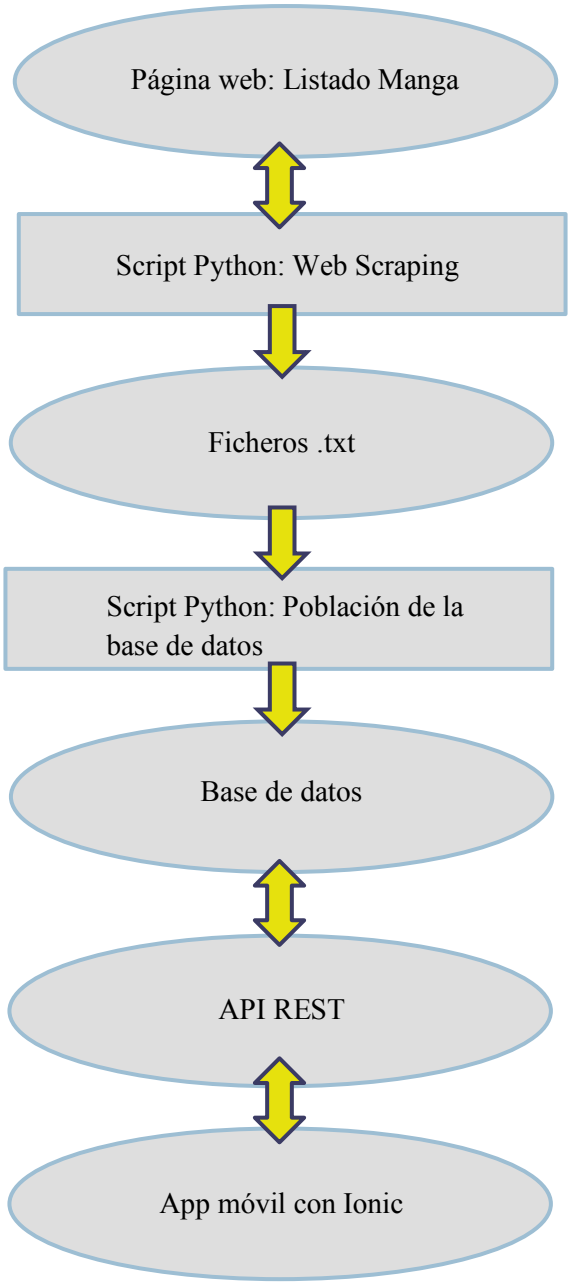

Figura 20. *Arquitectura del Sistema*

4.2 Diseño Detallado

Tras una exhaustiva búsqueda de posibles fuentes de información para el proyecto, se optó por hacer uso de la página web, «www.listadomanga.es». Se contemplaron al principio otras alternativas como utilizar la API de Amazon para acceder a la información de sus productos u otras páginas web como «www.myanimelist.net» o «www.whakoom.com». Una vez establecido nuestra fuente de información principal, se emplea la tecnología de extracción de datos, «web scraping». Realizando peticiones GET mediante el protocolo HTTP al servidor donde se aloja la información, obtenemos la información de ciertas URLs («Uniform Resource Locator»). No obstante, esta información es necesario acotarla para extraer aquello que más nos interesa y descartar el resto que no supone de gran relevancia para el proyecto. Mediante un script realizado con el lenguaje de programación Python, de cada una de las URLs se extrae información concreta de cada colección, volumen o novedad publicada o por publicar en el mercado español.

Una vez acotada la información necesaria, estos datos son almacenados en ficheros de texto para ser utilizados posteriormente por otro script que almacenará la información en la base de datos. El script leerá cada uno de los ficheros y realizará operaciones SQL, específicamente operaciones INSERT en la base de datos.

A continuación, se mostrará el esquema relacional de la base de datos implementada:

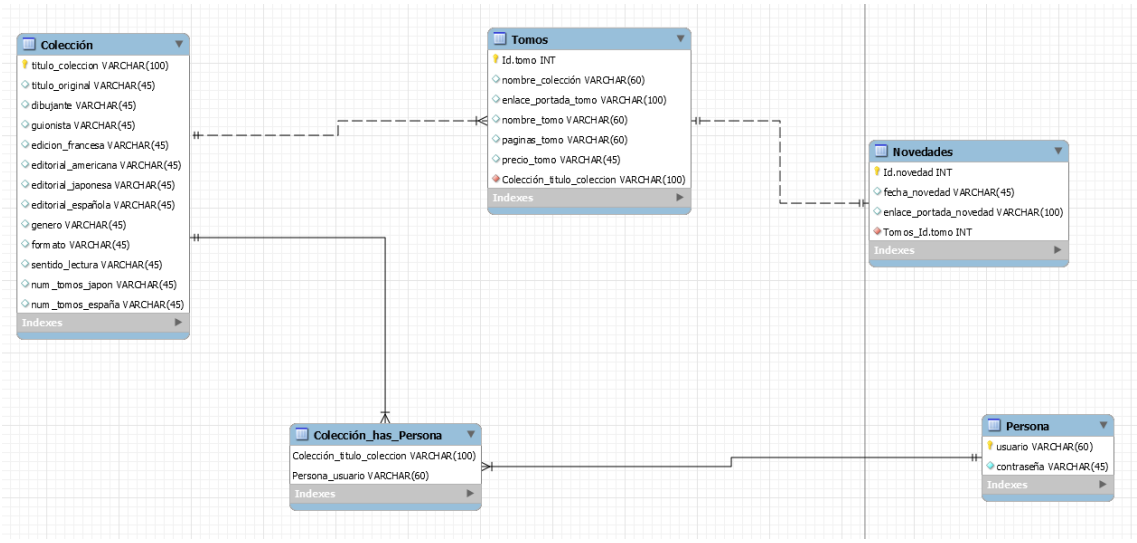

Figura 21. *Modelo físico de la base de datos relacional*

Considerando que la información recogida ha sido insertada correctamente en la base de datos, se establece una API REST pública que se comunica con el modelo relacional mediante un script. La API REST devuelve sus resultados en formato JSON a través de sus respectivas URIs («Uniform Resource Identifier»). A su vez, la aplicación móvil establece una conexión HTTP con la API REST para obtener los datos en formato JSON mediante el método GET.

5. Tecnología Utilizada

En este apartado se describen las diversas tecnologías y herramientas utilizadas a lo largo del proyecto, además de las decisiones tomadas en la fase de implementación:

5.1 Tecnologías

5.1.1 Python

Figura 22. Python

Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, con semántica dinámica. Su alto nivel de estructuras de datos incorporadas, combinadas con la mecanografía y la encuadernación dinámicas, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripts o de pegamento para conectar los componentes existentes entre sí. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y por lo tanto reduce el coste de mantenimiento del programa. Python soporta módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma de código fuente o binaria sin cargo para todas las plataformas principales, y pueden ser distribuidos libremente (*What is Python? Executive Summary*, s. f.).

5.1.2 BeautifulSoup

Beautiful Soup es una biblioteca de Python para obtener datos de HTML, XML y otros lenguajes de marcado. Ayuda a extraer un contenido particular de una página web, eliminar el marcado HTML y guardar la información. Es una herramienta de web scraping que te ayuda a limpiar y analizar los documentos que has sacado de la web (Wieringa, Jeri, 2012).

5.1.3 MariaDB

Figura 23. MariaDB

MariaDB es una base de datos relacional madura, estable y de código abierto. Desde su comienzo en 2009 como una rama o bifurcación de la base de datos MySQL, a su estado actual como la versión por defecto en la mayoría de las distribuciones de Linux, y la base de datos elegida por muchas empresas grandes y pequeñas, MariaDB ha demostrado que las comunidades de usuarios y desarrolladores, que trabajan y colaborando juntos, puede hacer más de lo que una sola empresa podría hacer. MariaDB comparte muchas características y capacidades de su base de datos principal, pero como la mayoría de los niños también ha superado a su padre en muchos aspectos (Bartholomew, Daniel, 2014, p. 1).

5.1.4 Angular

Figura 24. Angular

Angular es un proyecto de código abierto apoyado principalmente por Google. Desde su lanzamiento en 2009, Angular se ha convertido en uno de los marcos de aplicación web más populares. El objetivo de Angular es proporcionar un marco de trabajo MVVW («Model-view-viewmodel») para construir aplicaciones web complejas de una sola página. El equipo de Ionic decidió aprovechar el poder que este marco ofrece, así que se basaron en él. Por ejemplo, los componentes personalizados de la interfaz de usuario de Ionic son sólo componentes de Angular. Angular está licenciado bajo la licencia del MIT y está disponible en el sitio web de Angular.

Con el lanzamiento de Angular 2, el marco ha cambiado enormemente. Este cambio causó cierta discordia dentro de la comunidad Angular, pero muchas de las preocupaciones sobre los nuevos aspectos del marco han sido abordadas (Griffith, Chris, 2017).

5.1.5 TypeScript

Figura 25. TypeScript

TypeScript es un poco inusual como lenguaje, ya que no se ejecuta en un intérprete (como lo hacen Python y Ruby) ni compila en un lenguaje de nivel inferior (como lo hacen Java y C). En su lugar, compila a otro lenguaje de alto nivel, JavaScript. Es JavaScript el que se ejecuta, no TypeScript. Así que la relación de TypeScript con JavaScript es esencial, pero también puede ser una fuente de confusión. El sistema de tipografía de TypeScript también tiene algunos aspectos inusuales de los que debes ser consciente (Vanderkam, Dan, 2019).

5.1.6 HTML

Figura 26. HTML5

El HTML es un lenguaje informático concebido para permitir la creación de sitios web. Estos sitios web pueden ser vistos por cualquier persona conectada a Internet. Es relativamente fácil de aprender, con lo básico accesible para la mayoría de la gente en una sola sesión; y bastante poderoso en lo que permite crear. Está en constante revisión y evolución para satisfacer las demandas y requerimientos de la creciente audiencia de Internet bajo la dirección de la W3C, la organización encargada de diseñar y mantener el lenguaje.

Consiste en una serie de códigos cortos tecleados en un archivo de texto por el autor del sitio, estas son las etiquetas. El texto se guarda como un archivo .html, y se ve a través de un navegador. Este navegador lee el archivo y traduce el texto en forma visible, con la esperanza de que la página sea como el autor había previsto (Shannon, Ross, 2012).

5.1.7 Flask

Figura 27. Flask

Flask es un «micro Framework» escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC («Model-view-controller»). La palabra «micro» no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas sino que al instalar «Flask» tenemos las herramientas necesarias para crear una aplicación web funcional pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones («plugins») que se pueden instalar con Flask que le van dotando de funcionalidad (Domingo Muñoz, José, 2017).

5.2 Herramientas

5.2.1 Visual Studio Code

Figura 28. Visual Studio Code

Visual Studio Code es un ligero pero poderoso editor de código fuente que se ejecuta en el escritorio y está disponible para Windows, MacOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity) (*Visual Studio Code*, s. f.).

5.2.2 XAMPP

Figura 29. XAMPP

XAMPP es un software de código abierto desarrollado por los amigos de Apache. El paquete de software XAMPP contiene distribuciones Apache para el servidor Apache, MariaDB, PHP y Perl. Y es básicamente un host o un servidor locales. Este servidor local funciona en tu propio ordenador de sobremesa o portátil. El uso de XAMPP es para probar los clientes o su sitio web antes de subirlo al servidor web remoto. Este software de servidor XAMPP te da el entorno adecuado para probar proyectos MYSQL, PHP, Apache y Perl en el ordenador local.

La forma completa de XAMPP es: (X) significa Cross-platform, (A) servidor Apache, (M) MariaDB, (P) PHP y (P) Perl. La multiplataforma generalmente significa que puede funcionar en cualquier ordenador con cualquier sistema operativo (Ganesan, Prahbu, 2017).

5.2.3 Ionic

Figura 30. Ionic

Ionic es un marco de trabajo de interfaz de usuario construido con HTML, CSS y JavaScript para su uso con el desarrollo de aplicaciones móviles híbridas. Más allá de los componentes de la interfaz de usuario, el Framework Ionic se ha ampliado para incluir una robusta interfaz de línea de comandos (CLI) y un conjunto de servicios adicionales como Ionic View y Ionic Creator.

Es realmente una combinación de varias tecnologías que trabajan juntas para hacer que la construcción de aplicaciones móviles sea más rápida y fácil. La capa superior de esta pila es el propio Framework Ionic, que proporciona la capa de interfaz de usuario de la aplicación. Justo debajo de eso está Angular (formalmente conocido como AngularJS), un marco de aplicación web increíblemente poderoso (Griffith, Chris, 2017).

5.2.4 Firebase

Figura 31. Firebase

Firebase se trata de una plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida, con el fin de que se pueda aumentar la base de usuarios y ganar más dinero. La plataforma está subida en la nube y está disponible para diferentes plataformas como iOS, Android y web. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades. Se inició cuando Google la compró en 2014, y seguidamente la fue mejorando mediante la compra del equipo de Divshot (Cardona Pérez, Manuel, 2016).

5.2.5 MySQL Workbench

MySQL Workbench es una herramienta gráfica para trabajar con MySQL. Proporciona una interfaz fácil de usar para realizar las muchas tareas que implica trabajar con bases de datos. Integra el desarrollo, la administración, el diseño, la creación y el mantenimiento de bases de datos SQL en un entorno visual de desarrollo integrado (Ian, 2016).

Figura 32. *MySQL Workbench*

5.2.6 Node.js

Figura 33. *Node.js*

Node.js es un marco de trabajo para escribir aplicaciones JavaScript del lado del servidor. Está construido sobre el tiempo de ejecución de JavaScript V8 y utiliza un modelo de E/S basado en eventos y sin bloqueo que lo hace perfecto para aplicaciones intensivas en datos y en tiempo real.

Algunas de las principales empresas del mundo utilizan Node en la producción, como Netflix, Paypal, Walmart y Uber (Henderson, Michael, 2019).

6. Web Scraping

La recopilación automatizada de datos de Internet es casi tan antigua como la propia Internet. Aunque el «web scraping» no es un término nuevo, en años pasados la práctica se conocía más comúnmente como «screen scraping», minería de datos, recolección web o variaciones similares. Aunque también hace referencia a programas que específicamente atraviesan múltiples páginas como rastreadores web o se refieren a los propios programas de web scraping como «bots».

En teoría, el «web scraping» es la práctica de recopilar datos a través de cualquier medio que no sea un programa que interactúe con una API (o, obviamente, a través de un humano utilizando un navegador web). Esto se logra más comúnmente escribiendo un programa automatizado que consulta un servidor web, solicita datos (generalmente en forma de HTML y otros archivos que componen las páginas web) y luego analiza esos datos para extraer la información necesaria.

En la práctica, abarca una amplia variedad de técnicas y tecnologías de programación, como el análisis de datos, el análisis sintáctico de lenguaje natural y la seguridad de la información. Aunque los navegadores son útiles para ejecutar JavaScript, mostrar imágenes y organizar objetos en un formato más legible para los humanos (entre otras cosas), los «web scrapers» son excelentes para reunir y procesar grandes cantidades de datos rápidamente. En lugar de ver una página a la vez a través de la estrecha ventana de un monitor, se pueden ver bases de datos que abarcan miles o incluso millones de páginas a la vez (Mitchell, Ryan, 2018).

Este tipo de técnicas han facilitado en gran medida la obtención de la información necesaria para llevar a cabo este proyecto. Para este propósito, se ha hecho uso de la biblioteca de Python, BeautifulSoup4. Su gran ventaja radica principalmente en su gran sencillez y versatilidad para el aprendizaje. Es una herramienta perfecta para manejar elementos del HTML DOM, lo que permite buscar, navegar y modificar elementos incluidos en el documento. Además de poder automatizar los procesos implicados en la recogida de información.

Para poder desarrollar los scripts necesarios para la obtención de los datos, se ha utilizado la versión de Python 3.7.7 de 64 bits y la versión de Visual Studio Code 1.45.0.

En los scripts, se han establecido una serie de pasos a seguir para obtener correctamente los datos de colecciones, volúmenes y novedades. El esquema para obtener la información acerca de las colecciones y sus respectivos tomos es el siguiente:

1) Almacenar en una lista los enlaces de cada una de las colecciones publicadas
2) Acceder a cada uno de los enlaces
3) Obtener la información relativa a las colecciones y sus volúmenes
4) Almacenar los datos de cada obra y tomo en listas
5) Almacenar la información en ficheros de texto (.txt)

En otro orden de ideas, el script desarrollado para obtener las diferentes novedades anunciadas por cada editorial adopta el siguiente conjunto de pasos:

1) Obtener la fecha y el enlace de la portada de cada novedad
2) Almacenar los datos obtenidos en listas

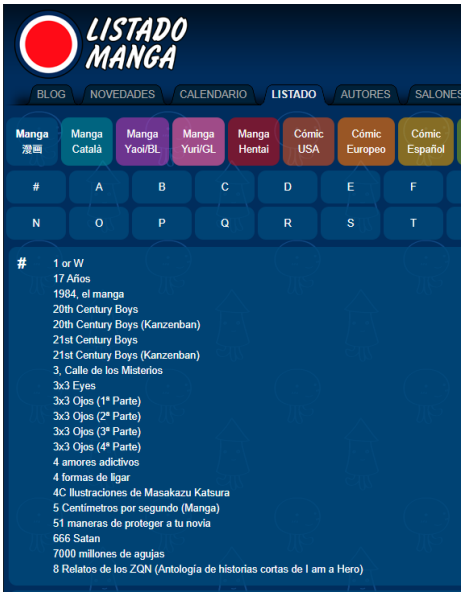

Figura 34. *Enlaces de las colecciones en Listado Manga*

6.1 Preparación del entorno de desarrollo

Primeramente, es necesario instalar el lenguaje de programación Python desde su página web «https://www.python.org». Una vez realizado este paso esencial para el desarrollo de los futuros scripts, se debe instalar la biblioteca **BeautifulSoup4** mediante el comando «pip» desde la consola de Windows:

- ***$pip install beautifulsoup4***

Posteriormente, se importan las diversas librerías a utilizar en los diversos ficheros que contendrán los scripts. En este caso, solo se ha necesitado instalar una biblioteca, debido a que el resto de las librerías utilizadas vienen predefinidas en la versión 3 de Python. En la figura 35, se muestran las librerías utilizadas para la obtención de información en un medio externo:

```
# -*- coding: utf-8 -*-
from urllib.request import urlopen
import urllib
from urllib.error import HTTPError
from urllib.error import URLError
from bs4 import BeautifulSoup
from http.client import IncompleteRead
```

Figura 35. *Importación de las librerías en los ficheros de Python*

6.2 Colecciones

Antes de todo, es necesario acceder a nuestra fuente de información, en este caso, la página web «www.listadomanga.es». La página está organizada en varias secciones, entre las cuales encontramos: Blog, Novedades, Calendario, Listado, Autores, Salones, Enlaces y Contacto. El primer objetivo a cumplir es obtener todos los enlaces relacionados con las colecciones publicadas. Para ello, el script comienza su búsqueda en la sección «Listado», por lo tanto, la URL inicial será «www.listadomanga.es/lista.php» (Figura 34). Se realiza una petición GET al servidor web mediante la biblioteca «urllib.request», en caso de producirse algún tipo de error se hace uso de la biblioteca «urllib.error» para lanzar una excepción y mostrar el error pertinente.

Una vez realizada la petición, es indispensable obtener el documento HTML y realizar una lectura para determinar los elementos necesarios y descartar el resto. Para ello, se utiliza la biblioteca «beautifulsoup4» para leer el documento y extraer los enlaces. Realizando una investigación exhaustiva de la página, se llega a la conclusión que todos los enlaces se encuentran en celdas bajo la misma clase. A continuación, se muestra la estructura propia de la página:

```
<html>

<head>…</head>
<body>
<center>
<table>
<td class=”izq”>
<a href="coleccion.php?id=11">1 or W</a>
</td>
</center>
</body>

</html>
```

***Figura 36**. Enlaces colecciones en un documento HTML*

Como se muestra en la figura 36, todos los enlaces se encuentran en celdas de la tabla indicadas con la etiqueta <td> y la clase «izq». Sabiendo esto, el script realizará la búsqueda dentro de estas etiquetas para buscar los enlaces y obtener su atributo «href», el cual nos indica la dirección web donde se sitúa la información de la colección y de sus respectivos tomos. Una vez obtenidos los enlaces, estos se almacenan en una lista.

Cabe destacar que en el script se han llevado a cabo dos métodos que obtienen de igual manera la información relativa a los enlaces de las colecciones, no obstante, no obtienen la misma información final. Mientras que uno es usado para obtener la información relacionada con las colecciones, el otro es empleado para la recopilación de los datos de cada uno de los tomos.

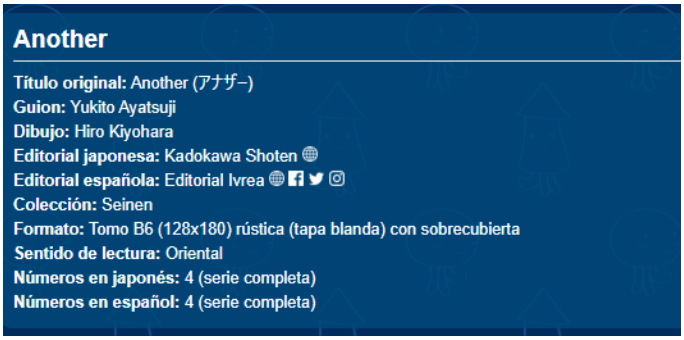

Figura 37. Ejemplo características de una obra en Listado Manga

Seguidamente se recorre la lista de enlaces para poder acceder a cada uno. Se vuelve a estudiar la estructura que presenta el documento HTML para cada colección y se llega a la conclusión que se sigue el mismo procedimiento que para obtener las direcciones web de las colecciones. Por lo tanto, los datos se encuentran en una celda de una tabla y especificada bajo la clase izq. Cada colección puede contener los siguientes atributos (Figura 37):

- Título original de la obra en japonés
- Guionista/s
- Dibujante/s
- Editorial japonesa
- Editorial americana
- Editorial francesa
- Editorial española
- Género al que pertenece la obra
- Formato de cada volumen
- Sentido de lectura (occidental u oriental)
- Número de tomos publicados en Japón y su estado, es decir, si la colección se encuentra abierta o cerrada en Japón
- Números de tomos publicados en España y su estado, es decir, si la colección se encuentra abierta o cerrada en España

Para poder obtener cada uno de los atributos de las colecciones, se incluyó en una variable de tipo «String» todo el texto contenido en la celda con la clase izq. Por medio del método «split()» se crea una lista, a raíz de separar el «String» por saltos de línea. Como resultado, se obtiene una lista donde cada atributo ocupa un índice. En cada función, se recorre dicha lista para obtener los atributos correspondientes y almacenarlos en una nueva lista. De este modo, si se almacenan los dibujantes de todas las colecciones, analizando la lista resultante después de haber aplicado el método «split()», la información relativa a los dibujantes siempre se encontrará en el índice tres. Los datos de los dibujantes se buscan mediante el método «find()», una vez encontrados se insertan en la nueva lista. Esto será utilizado más adelante para guardar la información recopilada en ficheros de texto.

```
def tit_original(children_col):
    for child_col in children_col:
        x = child_col.get_text()
        j = x.split("\n")


        for campo in j:
            ind = j.index(campo)
            if campo.find("Otras ediciones de ") == 0 or campo.find("Números editados") == 0
                j.pop(ind)
        #print(j)
        for campo in j:
            ind = j.index(campo)
            if ind==2:
                titulo_str = campo
                if titulo_str.find("Título original:") == 0:
                    titulo_org = titulo_str.split("Título original:")
```

Figura 38. *Función para obtener todos los títulos originales de las colecciones*

Una observación a tener en cuenta es que las funciones declaradas para obtener los diferentes atributos de las colecciones no devuelven la lista completa con la sentencia «return», sino que se hace uso de la sentencia «yield». Se ha tomado esta decisión debido a que las listas contienen un gran número de elementos y devolver la lista completa sería mucho más costoso en cuanto a recursos y memoria empleada. Por su lado, «yield» genera un iterador, lo que significa que se debe recorrer el resultado de la función para obtener uno a uno los elementos almacenados en la lista.

6.3 Volúmenes

Asimismo, las colecciones contienen una serie de volúmenes situados en el mismo sitio web. Sin embargo, se sitúan en diferentes tablas y los volúmenes no se sitúan en celdas bajo la clase «izq», sino en celdas con la clase «cen». Esto facilita la extracción de información entre colecciones y volúmenes. Cada volumen contiene las siguientes características (Figura 39):

- Título del volumen
- Número de páginas a color y en blanco y negro
- Precio del volumen
- Fecha de publicación

Figura 39. *Ejemplo volúmenes de una obra en Listado Manga*

Al igual que las colecciones, se utiliza una variable de tipo «String» para contener el texto correspondiente a cada volumen. Sin embargo, a diferencia del texto obtenido para las colecciones, en este no es necesario utilizar el método «split()» para separar los elementos, salvo en el caso de obtener el título de los tomos donde es necesario realizar un «splitlines()» para

eliminar los posibles saltos de línea. Aunque en este caso, no recorremos una segunda lista sino que utilizamos el método «join()» para volver a unificar los elementos y obtener un nueva variable «String» sin saltos de línea. Se vuelve a hacer uso del método «find()» para escrutar los atributos correspondientes en cada función y almacenarlos en una lista. De igual modo que en colecciones, estas listas se utilizarán para almacenar su contenido en ficheros de texto.

```
for datos_tomos in col_html:
    u = datos_tomos.get_text()
    u = "".join(u.splitlines())
    w = u.find(" página")
    n = u.find("nº")
    inicio = 0
    pesetas = u.find("Ptas.")
    pag_bn = u.find("B/N")
    pag_color = u.find("color")
    euros = u.find(" €")
    resta_ind = w - n
```

Figura 40. *Eliminar los saltos de línea de la información recogida*

6.4 Novedades

Figura 41. *Ejemplo novedades de un mes en Listado Manga*

Como se había comentado anteriormente, para obtener la información correspondiente a las diversas novedades publicadas por cada editorial se ha desarrollado un segundo script. En la página web de Listado Manga, podemos buscar las novedades tanto en el apartado de calendario como en el de novedades, en este caso se ha optado por el apartado de novedades, ya que estaba mejor estructurado y era más sencillo obtener la información. Por lo tanto, la URL inicial donde el script comenzará a buscar información será «www.listadomanga.es/novedades.php» (Figura 41). El objetivo principal es obtener tanto las fechas de publicación como los enlaces de las portadas de cada novedad. Para ello, se vuelve a hacer uso de la biblioteca «urllib.request» para realizar una petición GET al servidor web, y la biblioteca «beautifulsoup4» para obtener el DOM completo de cada una de las páginas que se van a visitar. Se realiza un análisis de la página donde

se muestran las novedades, y tanto la fecha como los enlaces de las portadas se sitúan en celdas de una tabla con la clase «cen». A continuación, se muestra un ejemplo de la estructura de la página en HTML:

```
<html>

<head>…</head>
<body>
<center>
<table>
<td class=”cen”>
<h2>Novedades de Junio 2020</h2>
</td>
<td class=”cen”>
<a href="coleccion.php?id=2478">
<img src="imagenes/colecciones/manga/1/21stcenturyboyskanzenban.jpg"></a>
</td>
</center>
</body>

</html>
```

Figura 42. *Ejemplo HTML novedades*

Como se observa en la figura 42, tanto las fechas de las novedades incluidas entre las etiquetas <h2>, y las imágenes de las portadas, las cuales comienzan con la etiqueta <img>, se encuentran en celdas de una tabla especificadas bajo la clase «cen».

Para este caso, es necesario realizar una función recursiva, ya que tiene que obtener no solo las fechas y las imágenes del mes actual, sino también de los meses posteriores. Para obtener las fechas de las novedades, se hace uso del método «find_all()» para encontrar en el HTML todas las etiquetas <h2>. El paso a seguir es buscar únicamente las que empiecen por “Novedades de”, para ello se utiliza el método «find()» y se añaden los resultados a una lista. En cambio para obtener los enlaces de las portadas de cada novedad, se utiliza el método «find_all()» para buscar etiquetas <img>.

Si observamos la figura 42, veremos que la ruta indicada en la imagen no está completa, por lo tanto, hay que completarla en el código para poder obtener la imagen en un futuro. Por lo tanto, se debe indicar en los elementos obtenidos anteriormente que solamente se va a utilizar el atributo «src», el cual indica su ubicación en el servidor. De igual modo, se añadirá al principio la ruta inicial al atributo «src» de la imagen, por lo tanto, el resultado será el siguiente: “http://www.listadomanga.es/” + img_nov[‘src’]. Cada ruta es insertada en una nueva lista, aunque debemos eliminar el logo de cada editorial, ya que no interesan para el proyecto. Esto se realiza de manera sencilla mediante el método «pop()» empleado en listas para eliminar elementos. Una vez obtenido toda la información de ese mes en concreto, es necesario volver a llamar a la función indicándole la URL del mes siguiente. Dicha URL se encuentra en una celda de la tabla y con la clase «der». Este script no deja de ejecutarse hasta que llega a un mes donde no se halla ninguna imagen de alguna novedad.

7. Creación y población de la base de datos

A partir del modelo físico diseñado mediante el programa «MySQL Workbench» (ver Figura 21), se ha empleado una de las funcionalidades propias del programa para sincronizar el modelo con la base de datos creada previamente con la herramienta «phpMyAdmin». La base de datos recibe el nombre de «mydb», desarrollada mediante «MariaDB», caracterizada por ser una base de datos relacional. En este apartado, se describirá el proceso de inserción de los datos almacenados en el apartado anterior mediante ficheros de texto y a través de la comunicación con la API REST, además de las relaciones establecidas entre las diferentes tablas de la base de datos.

7.1 Preparación del entorno de desarrollo

Es indispensable utilizar la biblioteca «mysql.connector» para poder realizar todas las sentencias relacionadas con la base de datos. Para ello, es necesario instalarla mediante el comando pip en la consola de Windows:

- ***$pip install mysql-connector-python***

A continuación, se importa la biblioteca en los ficheros que insertarán u obtendrán tuplas en la base de datos. Para ello, se hace uso de la sentencia «import» al principio de los ficheros:

- ***import mysql.connector***

7.2 Población de la base de datos

El proceso de inserción de los datos en las diversas tablas del SGBD («Sistema gestor de bases de datos») varía de manera muy poco significativa entre los scripts desarrollados. No obstante, se describirán las principales diferencias entre estos y su estructura final en las tablas.

Los pasos adoptados por parte del script encargado de almacenar información relativa a colecciones y volúmenes son los siguientes:

1) Crear una conexión y establecer una sesión con la base de datos.
2) Leer los ficheros de texto de cada atributo.
3) Insertar los datos en nuevas listas.
4) Recorrer las listas y crear tuplas para insertar correctamente los datos en la base de datos.
5) Hacer uso de la instrucción «Commit», la cual nos garantiza que todas las modificaciones realizadas en la base de datos sean permanentes.
6) Cerrar la conexión y la sesión con la base de datos.

Para crear una conexión y una sesión entre el script y la base de datos se ha hecho uso del método «connect()», el cual se conecta a la dirección IP «127.0.0.1» que corresponde al servidor local y utiliza el puerto «3306» propio de MySQL. Al mismo tiempo, para insertar las tuplas de datos se emplea el método «executemany()» para realizar varias sentencias INSERT en la base de datos. Sin embargo, para que estas sentencias resulten definitivas es necesario aplicar el método «commit()». Para finalizar el algoritmo, se emplea el método «close()» para concluir tanto la sesión iniciada como la conexión con el SGBD.

```
conn = mysql.connector.connect(host="localhost",port=3306,db="mydb",user="root",password="")
cursor = conn.cursor()
```

Figura 43. *Configuración de la base de datos en Python.*

```
    cursor.executemany(
        "INSERT INTO `novedades`(`Id.novedad`, `fecha_novedad`, `enlace_portada_novedad`,`Tomos_Id.tomo`)
         VALUES (%s,%s,%s,%s)", novedades_commit)
    conn.commit()
except mysql.connector.Error as e:
    print("Failed to insert record into MySQL table {}".format(error))
finally:
    if (conn.is_connected()):
        cursor.close()
        conn.close()
```

Figura 44. *Insertar tuplas en la base de datos, realizar un commit y cerrar la conexión.*

Por otro lado, el script empleado para obtener las novedades no sigue los mismos pasos que el anterior, esto se refleja en que no utiliza ficheros de texto para almacenar sus datos, sino que devuelve la lista con las fechas y las portadas directamente. Esto se debe a que no contiene una gran cantidad de información y por lo tanto no supone un gran coste computacional o de memoria. Esto supone que los pasos utilizados para insertar los datos en su respectiva tabla del SGBD, sea prácticamente igual.

No obstante, a lo largo del proceso de inserción de los datos nos encontramos con un problema considerable. Al insertar los datos con las claves ajenas establecidas entre tablas, se producía un error debido a que dichas referencias no encontraban valores en la columna a la que apuntaban. Por ese motivo, al insertar los datos en tablas con alguna clave ajena, se eliminaba primero la referencia, se insertaban después los datos y en último lugar se volvían a establecer las claves ajenas en sus respectivas tablas. De este modo, ya había datos disponibles y la clave ajena proporcionaba una referencia a dichos valores.

Por este motivo, en la clave ajena de novedades «Tomos_Id.tomo» se establece que la referencia debe apuntar a los tomos que sean semejantes a las novedades publicadas por cada editorial. Como se había visto en el modelo entidad-relación, se constituye una relación «uno a uno» lo que supone que una novedad es un tomo y viceversa. Esto supone un problema, ya que previamente no se había programado dicha condición. A causa de esto, en el script de novedades se realiza una función para comparar las portadas de tomos y novedades, y así obtener los tomos que cumplan dicha condición. Esto se ejecuta mediante una consulta a las dos tablas (novedades y tomos) de la base de datos. La función devuelve una lista con los valores resultantes. Para evitar que al borrar alguna de las dos tablas, las consultas no devuelvan ningún valor, es necesario guardar los valores en un fichero de texto. Esta solución permite introducir correctamente los valores en la clave ajena de la tabla novedades.

8. Implementación y configuración de la API REST

REST es el acrónimo de «Representational State Transfer». Es un estilo arquitectónico para sistemas hipermedia distribuidos y fue presentado por primera vez por Roy Fielding en el 2000 en su famosa disertación. Como cualquier otro estilo arquitectónico, REST también tiene sus propias restricciones que deben ser satisfechas si una interfaz necesita ser referida como «RESTful». Estos principios se enumeran a continuación:

1. **Cliente-servidor** - Al separar las preocupaciones de la interfaz de usuario de las preocupaciones de almacenamiento de datos, se mejora la portabilidad de la interfaz de usuario a través de múltiples plataformas y se mejora la escalabilidad simplificando los componentes del servidor.
2. **Sin estado** - Cada solicitud de cliente a servidor debe contener toda la información necesaria para entender la solicitud, y no puede aprovechar ningún contexto almacenado en el servidor. Por lo tanto, el estado de la sesión se mantiene enteramente en el cliente.
3. **«Cacheable»** - Las restricciones de la caché requieren que los datos dentro de una respuesta a una solicitud sean etiquetados implícita o explícitamente como «cacheables» o no «cacheables». Si una respuesta es almacenada en la memoria caché, entonces se le da a la memoria caché del cliente el derecho de reutilizar esos datos de respuesta para solicitudes posteriores equivalentes.
4. **Interfaz uniforme** - Aplicando el principio de generalidad de la ingeniería de software a la interfaz de componentes, se simplifica la arquitectura general del sistema y se mejora la visibilidad de las interacciones. Para obtener una interfaz uniforme, se necesitan múltiples restricciones arquitectónicas para guiar el comportamiento de los componentes. REST se define por cuatro restricciones de interfaz: identificación de recursos; manipulación de recursos a través de representaciones; mensajes autodescriptivos; e hipermedia como motor de estado de la aplicación.
5. **Sistema de capas** - El estilo de sistema de capas permite que una arquitectura se componga de capas jerárquicas restringiendo el comportamiento de los componentes de tal manera que cada componente no puede "ver" más allá de la capa inmediata con la que está interactuando.
6. **Código bajo demanda (opcional)** - REST permite que la funcionalidad del cliente se extienda descargando y ejecutando código en forma de «applets» o scripts. Esto simplifica a los clientes reduciendo el número de características que se requieren para ser pre-implementadas.

La abstracción clave de la información en REST es un recurso. Cualquier información que pueda ser nombrada puede ser un recurso: un documento o imagen, un servicio temporal, una colección de otros recursos, un objeto no virtual (por ejemplo, una persona), y así sucesivamente. REST utiliza un identificador de recursos para identificar el recurso concreto que participa en una interacción entre componentes (Fielding, Roy Thomas, 2000, Chapter 5).

Otra cosa importante asociada con REST son los métodos de recursos que se utilizarán para realizar la transición deseada. Un gran número de personas relacionan erróneamente los métodos de recursos con los métodos HTTP GET/PUT/POST/DELETE.

Para una correcta configuración, se han seguido una serie de prácticas para poder proporcionar el servicio más adecuado a los potenciales consumidores de la API. Las medidas tomadas se enumeran a continuación[1]:

1. No incluir verbos en los literales de la URI. En su lugar, mejor nombres que identifican a los recursos, y que sean las operaciones utilizadas las que definan la interacción con los datos sobre los recursos (GET, POST, PUT y DELETE).
2. No usar GET para cambios de estado de un recurso. Las modificaciones de estado de un recurso deben ir asociadas a POST, PUT o DELETE.
3. Usar singular o plural pero solo uno de ellos. Ser consistente en la nomenclatura de acceso a los recursos.
4. Codificar las relaciones como sub-recursos. Si dos recursos tienen relación entre ellos se pueden utilizar, salvo el caso N a N.
5. Usar cabeceras http para especificar el formato. Incluir en el «http-header» el formato del contenido.
6. Proporcionar capacidades de filtrado, ordenado y acceso a información de cada campo.
7. Incluir la versión de la API dentro de la URL.

Al poner en funcionamiento el script se detectó que el servidor rechazaba todas las peticiones enviadas por parte de la aplicación móvil. Esto era debido al CORS («Cross-Origin Resource Sharing»), un estándar que permite a un servidor relajar la política del mismo origen. Esto se usa para permitir explícitamente algunas peticiones de origen cruzado mientras se rechazan otras. Si la configuración del CORS no está correctamente configurada, la consola del navegador presentará un error como "Solicitud de cruce de origen bloqueada": La política de Mismo Origen no permite leer el recurso remoto en «$somesite» indicando que la solicitud fue bloqueada debido a la violación de las reglas de seguridad del CORS. Sin embargo, esto podría no ser necesariamente un error de configuración. Es posible que la solicitud esté siendo rechazada intencionadamente por la aplicación web del usuario y el servicio externo remoto. Para solucionar esto, se hizo uso de la clase CORS del módulo «Flask» para permitir las peticiones por parte de los servicios (MDN contributors, 2019).

8.1 Preparación del entorno de desarrollo

Para el desarrollo de una API REST en Python, es necesario instalar las siguientes librerías mediante el comando pip en la consola de Windows:

Flask

- ***$pip install Flask***

[1] Esta información ha sido tomada del módulo didactivo Diseño de APIs REST perteneciente a la asignatura Integración de Aplicaciones.

Flask-RESTful

- ***$pip install flask-restful***

Flask-CORS

- ***$pip install -U flask-cors***

Además, se ha hecho uso de la librería «json» para poder convertir las tuplas obtenidas de la base de datos en formato JSON. No obstante, dicha librería viene incluida en la versión 3 de Python, por lo que no es necesaria su instalación. En la figura 45, se muestran las diversas librerías importadas al inicio del script:

```
from flask import Flask,request,jsonify,abort
from flask_restful import Resource, Api
from json import dumps
from flask_cors import CORS
#import jsonify
import mysql.connector
```

Figura 45. *Importación de las librerías para la API REST*

8.2 Desarrollo de la API REST

Se han creado varias clases haciendo alusión a los diferentes recursos que se van a compartir mediante la API REST. Se ha decidido que los recursos a compartir y autorizados para su consumo sean los datos de todas las colecciones y los volúmenes existentes, además de poder buscar la información de una colección en concreto o los volúmenes pertenecientes a una colección. En cada clase, se ha llevado a cabo el mismo procedimiento para obtener los datos:

1. Establecer una conexión y una sesión con la base de datos mediante los métodos «connect()» y «cursor()», respectivamente.
2. Realizar una consulta mediante la operación SELECT y obtener los datos.
3. Guardar los datos en un diccionario para dejar los atributos y sus valores de manera estructurada.
4. Añadir el diccionario a una lista.
5. Aplicar el método «jsonify()» a la lista para devolver en formato JSON los datos obtenidos, en caso de no encontrar resultados a la petición solicitada, se muestra el error «404 – Not Found».

```python
class Tomo(Resource):
    def get(self):
        cursor.execute("SELECT * FROM `tomos`")
        query = cursor.fetchall()
        lista = []
        for x in query:
            dictionary = {}
            dictionary['id'] = x[0]
            dictionary['coleccion'] = x[1]
            dictionary['portada'] = x[2]
            dictionary['titulo'] = x[3]
            dictionary['paginas'] = x[4]
            dictionary['precio'] = x[5]
            lista.append(dictionary)
        return (jsonify(lista))
```

Figura 46. *Obtener la información de los tomos alojada en la base de datos*

Una vez realizado este procedimiento, se deben crear las URIs para identificar los recursos a publicar. Como se ha mencionado anteriormente, se han llevado a cabo una serie de prácticas para identificar los recursos de manera coherente y correcta. En primer lugar, se ha establecido una estructura interna de menos a más específico. Por ejemplo, para obtener todas las colecciones existentes se ha establecido la ruta: «/api/v1/colecciones». Como se puede apreciar, el recurso a obtener son las colecciones, indicado en plural y precedido por la versión de la API REST. Al igual que las colecciones para los tomos se hace uso de una ruta muy similar: «/api/v1/tomos». No obstante, esto varía si queremos especificar una colección o los volúmenes de una colección de manera específica. Para obtener los volúmenes de una colección, se utiliza la siguiente estructura en la URI: «/api/v1/tomos/<string:titulo_coleccion>». A todos los recursos se puede acceder mediante el método GET y el puerto 5002. El servicio es proporcionado a través de un servidor web funcionando en local. A continuación, se muestra el resultado de la búsqueda para conocer los volúmenes de la colección «La Chica a la Orilla del Mar» como ejemplo ilustrado:

127.0.0.1:5002/api/v1/tomos/La%20Chica%20a%20la%20Orilla%20del%20Mar


```
// 20200526214619
// http://127.0.0.1:5002/api/v1/tomos/La%20Chica%20a%20la%20Orilla%20del%20Mar

[
  {
    "coleccion": "La Chica a la Orilla del Mar",
    "id": 10153,
    "paginas": "194 páginas en B/N",
    "portada": "http://www.listadomanga.es/imagenes/colecciones/manga/l/lachicaalaorilladelmar01.jpg",
    "precio": "8,00 €",
    "titulo": "La Chica a la Orilladel Mar nº1"
  },
  {
    "coleccion": "La Chica a la Orilla del Mar",
    "id": 10154,
    "paginas": "216 páginas en B/N",
    "portada": "http://www.listadomanga.es/imagenes/colecciones/manga/l/lachicaalaorilladelmar02.jpg",
    "precio": "8,00 €",
    "titulo": "La Chica a la Orilladel Mar nº2"
  }
]
```


Figura 47. *Resultado en JSON de los volúmenes de una colección*

9. Diseño y desarrollo de la aplicación móvil.

9.1 Preparación del entorno de desarrollo

9.1.1 Instalación de Node y npm

Antes de comenzar a desarrollar una aplicación móvil en Ionic, es indispensable instalar previamente la biblioteca «Node.js» y el gestor de paquetes «npm». El instalador se puede obtener de manera sencilla a través de su página web «https://nodejs.org/es/». Esta instalación incluye conjuntamente npm.

9.1.2 Instalación de Ionic

Antes de poder comenzar la instalación de Ionic CLI («Command-Line Interface»), era necesario disponer de la biblioteca «Node.js» y el gestor de paquetes «npm». Para una instalación global de «Ionic CLI», se ha hecho uso de la consola «npm» propia de Visual Studio Code. El comando utilizado es el siguiente:

- ***$npm install -g @ionic/cli***

También se nos permite conocer la versión instalada, mediante el siguiente comando:

- ***$ionic –version***

En este proyecto, se ha instalado la versión 5.4.16 de Ionic.

9.1.3 Creación de un proyecto en Ionic

Primeramente, se crea una carpeta con el nombre de la aplicación para alojar el proyecto. Se ha decidido que el nombre de la aplicación sea finalmente: Ikari. Al comenzar un nuevo proyecto, Ionic proporciona a los desarrolladores una serie de plantillas para facilitar el comienzo del proyecto. Sin embargo, uno de los objetivos es desarrollar desde cero una aplicación, por lo tanto en este caso, no se hace uso de las plantillas. Para crear un proyecto de cero, se utiliza el siguiente comando:

- ***$ionic start Ikari blank***

En este paso, se nos pregunta qué tipo de «framework» vamos a usar a lo largo del proyecto: Angular o React. Debido a su simplicidad y a su compatibilidad tan amplia con Ionic, se ha optado por usar Angular para este proyecto. Angular ya viene por defecto con Ionic, por lo que no hay necesidad de instalar ningún paquete relacionado con Angular.

Para poner en marcha el servidor que alojará nuestra aplicación y poder visualizarla en el navegador a través del puerto 8100, se hace uso del siguiente comando:

- ***$ionic serve***

En el fichero config.xml, se puede cambiar el nombre por defecto de la aplicación en la etiqueta «name» e incluir una descripción en la etiqueta «description».

9.1.4 Configuración de Firebase

Para este proyecto se ha decidido hacer uso de la plataforma digital de Google, Firebase. Dicha plataforma va a ser realmente útil en la integración dinámica de los usuarios de la aplicación mediante Firebase Authentication, además de poder almacenar en formato JSON los datos de cada usuario mediante su base de datos NoSQL, Cloud Firestore. Para poder utilizar estas herramientas, es necesario generar un proyecto en Firebase con el mismo nombre que tendrá nuestra aplicación. Al crear un nuevo proyecto, se nos proporciona una configuración destinada al desarrollo de la aplicación. En el fichero «environment.ts» de nuestro proyecto, se crea una variable denominada «firebaseConfig» con los parámetros de la configuración proporcionada por Firebase. Dicha variable es importada en el fichero «app.module.ts» para permitir la utilización de los servicios de Firebase en todos los módulos de la aplicación.

9.2 Inicio de sesión y registro

Como se ha mencionado anteriormente, para crear un servicio funcional y rápido de autenticación se va a hacer uso de las herramientas: Firebase Authetication y Cloud Firestore. En nuestro proyecto, será necesario generar un servicio que realice el rol de proveedor de datos entre Firebase y la aplicación. Este servicio será consumido por los diversos componentes de la aplicación, en los cuales se delegará la responsabilidad de acceder a la información. Para generar el servicio «AuthService», es necesario ejecutar el siguiente comando:

- $ionic generate service AuthService

Una vez creado el servicio, es necesario importar los módulos a utilizar en el fichero «auth.service.ts» con las siguientes directrices:

```
import { Injectable } from '@angular/core';
import {AngularFireAuth} from "@angular/fire/auth";

import {Router} from "@angular/router";
import {AngularFirestore} from "@angular/fire/firestore";
```

Figura 48. *Módulos del servicio AuthService*

Es importante que el servicio sea exportable a otros componentes para poder utilizar sus métodos más adelante. Se define un método «login()» para verificar las credenciales de los diversos usuarios de la aplicación. Este método recibe como parámetros un usuario y una contraseña de tipo String, respectivamente. El proceso de verificación es muy simple, ya que se emplea el método «signInWithEmailAndPassword()». En caso de devolver true, se accede a las colecciones y los documentos del usuario en la base de datos de Firebase. Si el usuario no ha iniciado sesión con éxito, se captura la excepción y se muestra el error por consola.

```
login(email:string, password: string){

  return new Promise((resolve,rejected) =>{
    this.AFauth.signInWithEmailAndPassword(email,password).then((user) =>{
      const userUid = user.user.uid;
      const datos = {
        colecciones: [],
        volumenes: [],
        pendientes: []
      }
      this.db.collection("manga").doc(userUid).set(datos);
      resolve(user);

    }).catch(err => rejected(err));
  })
```

Figura 49. Login de usuarios en AuthService

Por otro lado, se ha desarrollado un método para registrar usuarios a través del método «register()». Este método recibe tres parámetros de tipo String: un email, una contraseña y el nombre del usuario. Se emplea el método «createUserWithEmailAndPassword()» para el registro de usuarios en Firebase. Al utilizar este método, se crea una colección denominada «users» y un documento con un identificador único de cada usuario mediante el método «set()». Este documento contiene los siguientes atributos: name, uid, email y password. A continuación, se muestra un documento generado en el proceso de registro:

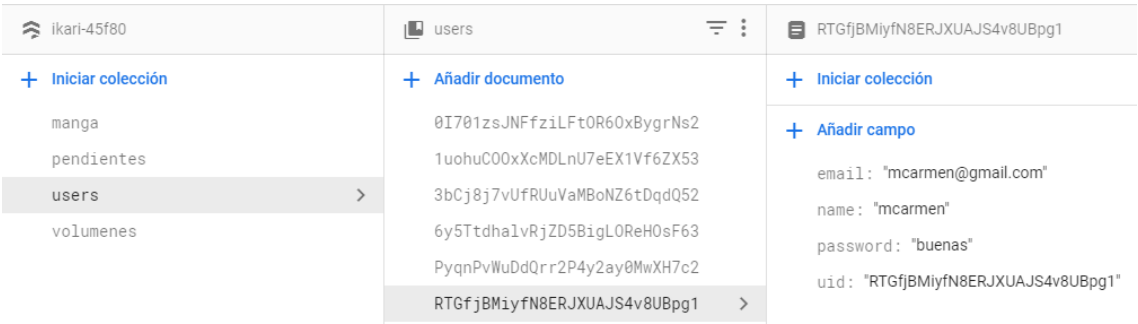

Figura 50. Documento generado durante el registro de un usuario

En adición, para desloguear a un usuario se ha implementado el método «logout()». Hace uso principalmente de la función «signOut()» para obtener al usuario logueado y llevarlo de vuelta a la ventana de inicio de sesión.

Seguidamente, se deben generar los componentes que consumirán los datos proporcionados por el servicio «AuthService». Las órdenes usadas en la consola de Visual Studio Code son las siguientes:

- $ionic generate page login
- $ionic generate page registro

Una vez creados los componentes, se emplea el fichero con extensión Typescript de cada componente para desarrollar la parte funcional. Para poder consumir los datos proporcionados por el servicio «AuthService», es necesario importar el servicio junto al módulo Router para desplazarse entre componentes. En el componente login, se crea un método «OnSubmitLogin()» para acceder al método «login()» del servicio, en caso de devolver true, se accede a la ventana principal de la aplicación. En caso contrario, se muestra un error en la aplicación.

De manera muy similar se comporta el componente register, este contiene la función «OnSubmitRegister()» que emplea el método «register()» del servicio. Al igual que en login, si la promesa devuelve true entonces se accede a la ventana principal de la aplicación, sino se devuelve un error.

Para la parte visual, se han creado un fichero HTML y un fichero CSS en cada componente. Los campos destinados al inicio de sesión se representan mediante «ion-inputs» , también se ha creado un botón que ejecuta las funciones correspondientes.

No obstante, se decidió establecer un nivel mayor de seguridad en cuanto al inicio de sesión y al registro de usuarios. El objetivo principal es evitar que usuarios sin los permisos adecuados accedan a la aplicación. Por este motivo, se ha hecho uso de los «guards». Este elemento permite identificar a un usuario y permitirle el acceso a la aplicación bajo unas condiciones. Para este proyecto, se han generado dos «guards» con el propósito de controlar los usuarios que acceden a la ventana de login y a la ventana principal de la aplicación. Para generar estos elementos se han usado los siguientes comandos:

- $ionic generate guard auth
- $ionic generate guard nologin

En el fichero «app-routing.module.ts» donde se sitúan todas las rutas de los componentes, importamos ambos «guards». En la ruta del componente home, especificamos que se va a utilizar uno de tipo «AuthGuard» mediante la función «canActivate: [AuthGuard]», y en los paths de login y register de tipo «NologinGuard» con la función «canActivate: [NologinGuard]». La función «canActivate()» se ha declarado previamente en cada «guard» para restringir el acceso a los diversos componentes.

9.3 Ventana principal y filtro de búsqueda

Una vez revisadas las credenciales del usuario y verificar su autenticidad, se le da paso a la ventana principal de la aplicación. Esta se caracteriza principalmente por establecer una barra de búsqueda para el filtrado de colecciones, un botón para visualizar la biblioteca personal del usuario y otro botón para conocer las novedades mensuales publicadas por cada editorial. Asimismo, en el lateral izquierdo se ubica un menú desplegable, el cual ofrece distintas funcionalidades a tener en cuenta. Por un lado, cuenta con un botón Inicio para volver a la ventana principal, un segundo botón para desplazarse a la ventana donde los usuarios pueden establecer su propio objetivo anual de volúmenes leídos y para finalizar, un botón para desloguear al usuario conectado.

Durante el proceso de desarrollo de la ventana principal y del menú de la aplicación, se pretendía incluir toda una funcionalidad conjunta en un mismo componente. Sin embargo, por problemas hallados al incluir el menú en el componente «Home», se decidió separar la funcionalidad propia de la ventana principal en el módulo «Home», y el menú situarlo en el componente raíz de la aplicación. Para desarrollar el menú de la aplicación, se desarrolló una función denominada «sideMenu()» para establecer diversos atributos con los nombres, las rutas y los iconos de sus elementos. En el HTML, se recorre cada elemento mediante la instrucción

«*ngFor» y se asocia cada grupo de atributos a un «ion-item». Por otro lado, como deseamos que el botón de deslogueo se sitúe en la parte inferior del menú, se crea un pie de página por medio de la etiqueta «ion-footer». En su interior, se coloca un «ion-button» para ejecutar la función «Onlogout()» creada previamente en el componente raíz.

Para motivar al usuario a seguir usando la aplicación y continuar adquiriendo manga, se ha establecido un desafío de lectura anual. En este desafío, los usuarios establecen un número de volúmenes a alcanzar al final del año y se va llevando la cuenta de los volúmenes leídos en todo momento. Esto se ha llevado a cabo en el componente «Challenge», donde se ha declarado una función asincrónica para mostrar un formulario. Este formulario permite al usuario introducir el número de volúmenes a leer o adquirir. En otro orden de ideas, es necesario obtener el número de volúmenes adquiridos hasta el momento, por este motivo se hace uso de un servicio denominado «ProveedorAPI». Este servicio es usado como intermediario en la aplicación para intercambiar información entre componentes. Al principio del fichero «challenge.page.ts» se importa el servicio «ProveedorAPIService» para usar el método «getObjetivoVolumenes()» declarado en el servicio. En adición, se incluye en el HTML una barra de progreso para mostrar el porcentaje de objetivo cumplido mediante la etiqueta «ion-progress-bar».

Como se ha mencionado previamente, la ventana principal se sitúa en el componente «Home». Este componente sirve de viaducto para llegar a otras ventanas de la aplicación, no obstante, también muestra en la parte central un listado de los volúmenes que el usuario ha marcado como pendientes de adquirir o leer en un futuro. En este listado, solamente se muestran las portadas de los volúmenes. Para obtener esta lista de tomos, es necesario importar el servicio «ProveedorAPI» y hacer uso de su método «getPendientes()». Este devuelve toda la información relativa a los volúmenes marcados como pendientes, aunque en este caso solo se hace uso del atributo portada.

Anteriormente, se ha mencionado que en la ventana principal se encuentra una barra de búsqueda. Al pulsar sobre ella, la aplicación muestra una nueva ventana con todas las colecciones incluidas previamente en la API REST. Para obtener esta información, se emplea el método «obtenerDatosColecciones()» del servicio «ProveedorAPI». Esta función realiza una petición GET a la siguiente ruta de la API REST: «http://127.0.0.1:5002/api/v1/colecciones». Esta ruta proporciona en formato JSON toda la información relativa de todas las colecciones publicadas a día de hoy en España. Para realizar esta petición, se importa el módulo «HttpClient» de Angular y se inicializa en el constructor. Para obtener los datos del servicio, se crea un nuevo componente denominado «Search». Al igual que en el resto de componentes, es necesario importar el servicio «ProveedorAPI». Se declara una función denominada «inicializaJSONData()» y se hace uso del método «subscribe()» para obtener todas las colecciones. Estos datos son almacenados en una variable global denominada «colecciones». Una vez obtenidos los datos, es indispensable establecer un filtro para facilitar la búsqueda de información al usuario. Se declara la función «filterItems()», a la cual se le pasa como parámetro las palabras introducidas por el usuario en el filtro de búsqueda. Esta información devolverá las colecciones que coincidan con los valores introducidos por el usuario.

```
inicializaJSONData(){
  this.proveedor.obtenerDatosColecciones().subscribe((data)=>{this.colecciones = data;
  },
  (error)=>{console.log(error);})
}

filterItems(searchTerm){
  return this.colecciones.filter(item =>{
    return item.titulo.toLowerCase().indexOf(searchTerm.toLowerCase()) > -1;
  });
}
```

Figura 51. Obtención y filtrado de las colecciones

Por otro lado, en el fichero HTML se hace uso de la etiqueta «ion-searchbar» para mostrar una barra de búsqueda. También, en el momento de buscar colecciones se ha querido mostrar una animación de carga. Esto se consigue gracias a la etiqueta «ion-spinner». Las colecciones son mostradas mediante un «ion-list», y en cada uno de sus «ion-items» únicamente aparece el título de estas. Al hacer click sobre una colección, se activará el evento «click» y se ejecutará la función «setData()» del servicio «ProveedorAPI». A través de esta función, le facilitamos al servicio la información relativa a dicha colección para ser reutilizada en otra ventana posteriormente.

9.4 Colecciones y volúmenes

Al seleccionar una colección en el filtro de búsqueda, se muestra una nueva ventana con la información más relevante de dicha colección, además de ofrecer los diversos volúmenes que la componen. Para presentar esta información al usuario, se crea un nuevo componente denominado «Colección-Card». Como en anteriores componentes, es necesaria la importación del servicio «ProveedorAPI». Mediante el método «getData()» se obtiene toda la información relativa a la colección seleccionada previamente. No obstante, esta información no desvela los volúmenes que contiene la colección. Por este motivo, es necesario hacer uso del módulo «HttpClient» y de su método «get()» para realizar una petición a la API REST. La petición realizada se dirige a la siguiente ruta: «http://127.0.0.1:5002/api/v1/tomos/<nombre_colección>». Por lo tanto, antes de realizar la petición, es indispensable guardar el título de la colección en una variable para concatenarla con el enlace. El resultado de esta petición es devuelto mediante la función «obtenerDatosTomosImgs()». Finalmente, los datos son obtenidos empleando el método «subscribe()».

```
public obtenerDatosTomosImgs(){
  this.datos_colec_card=this.proveedor.getData();
  this.titulo = this.datos_colec_card.titulo;
  this.titulo = this.titulo.replace(/\s/g,"%20");
  this.enlace = "http://127.0.0.1:5002/api/v1/tomos/";
  this.enlace2 = this.titulo;
  this.res = this.enlace.concat(this.enlace2);

  console.log(this.res);
  return this.http.get(String(this.res));
}
```

Figura 52. Obtener los datos de todos los tomos de una colección

En el HTML, se ha establecido que la información sea recopilada y estructurada mediante un «ion-card». Estos elementos son usados principalmente para ofrecer información más detallada como punto de entrada. Respecto a la información facilitada por la API REST de cada colección, no se ha hecho uso de todos sus atributos, sino que se han seleccionado los más relevantes. La información mostrada en la aplicación para cada colección manga es la siguiente:

- Título de la colección en España
- Título original de la colección
- Dibujante/s
- Guionista/s
- Editorial japonesa
- Editorial española
- Volúmenes publicados en Japón
- Volúmenes publicados en España
- Formato de la colección

Por otro lado, para representar los volúmenes de cada colección se ha usado un «ion-grid» para formar una matriz. Esta matriz se caracteriza principalmente por organizar las portadas de los volúmenes en cuatro columnas. Pulsando encima de una portada, se activa el evento «click» y se envían los datos de dicho volumen al servicio «ProveedorAPI» mediante el método «setData()». En consecuencia, se genera una nueva ventana con toda la información del volumen seleccionado. Se genera un nuevo componente bajo el nombre de «Tomo-Card» y se hace uso del método «getData()» para obtener los datos del volumen seleccionado con anterioridad. Asimismo, en el JSON obtenido se añaden dos nuevos atributos: «añadidoChecked» y «pendienteChecked». Estos atributos son utilizados en el HTML, ya que en el «ion-card» empleado para mostrar toda la información, se incluyen dos «ion-toggles». Estos elementos se emplean para conocer si un volumen debe ser añadido en la sección de volúmenes adquiridos o volúmenes pendientes. Para obtener su valor de entrada, se emplea el atributo «[(ngModel)]». Dependiendo del «ion-toggle» seleccionado por el usuario, el evento «ionChange» ejecutará la función «setVolumen()» para indicar que el volumen ha sido adquirido/leído o «setVolumen_Pendiente()» para indicar que el usuario desea obtener en un futuro el volumen.

```
<div>
  <ion-toggle class="toggle" slot="start" [(ngModel)]="tomo_info.anadidoChecked" (ionChange)="this.proveedor.setVolumen(tomo_info)">
  <ion-label  class="label">Lo tengo!</ion-label>
</div>
```

Figura 53. *Ion-toggle para añadir volúmenes adquiridos a la biblioteca personal*

También es posible acceder a un listado completo de los volúmenes de la colección, pulsando sobre el «ion-item» con nombre Volúmenes. Mediante este listado, se le permite al usuario seleccionar por medio de un «check-box» un gran número de volúmenes que han sido adquiridos o leídos. Estos volúmenes se añadirán posteriormente a la sección de volúmenes adquiridos de la biblioteca personal del usuario.

9.5 Biblioteca personal

En este apartado, se expondrá el desarrollo realizado con el objetivo de mantener un control de todos los volúmenes leídos o por leer de los usuarios y de las colecciones comenzadas o completadas. Para llevar esto a cabo, se ha partido de un nuevo componente bajo el nombre de «Intermediaria». En su fichero «intermediaria.page.ts», se importan los siguientes módulos:

```
import { Component, OnInit } from '@angular/core';
import {Router} from "@angular/router";
import { ProveedorAPIService} from '../proveedor-api.service';
import {AngularFirestore} from "@angular/fire/firestore";
import * as firebase from 'firebase';
```

Figura 54. *Módulos importados en el fichero intermediaria.page.ts*

Lo primero a obtener son los volúmenes seleccionados como adquiridos o pendientes de adquirir. Es necesario verificar que se haya seleccionado algo para no incluir elementos vacíos en la base de datos de Firebase. Dependiendo del estado que incluya el volumen seleccionado, se ejecutarán los métodos: «sendVolumen()» o «sendPendiente()» para incluir los tomos adquiridos o pendientes en Firestore. Ambos métodos siguen la misma dinámica algorítmica, creando una nueva colección en Firestore e incluyendo un documento internamente con el UID del usuario conectado y los volúmenes seleccionados. En la figura 55, se muestra un ejemplo del esquema establecido por Firebase al añadir tomos:

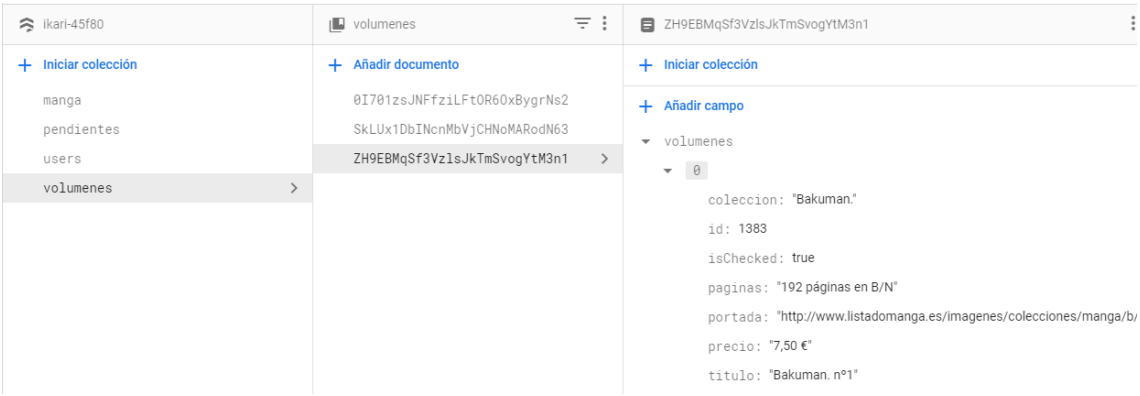

Figura 55. *Documento generado al insertar volúmenes adquiridos*

Posteriormente, se acceden a dichas colecciones mediante el servicio «ProveedorAPI» y se devuelve el resultado mediante los métodos «getVolumenes()» o «getPendientes()». Se obtiene también la longitud de cada uno de los arrays resultantes para mostrar al usuario el número de volúmenes adquiridos o de su lista de deseados. Por otro lado, la estructura seguida en el fichero HTML es establecer un «ion-item» para mostrar el volumen de tomos comprados o leídos y de tomos pendientes. En la parte central, se muestran las portadas de los volúmenes organizadas mediante un «ion-grid». Cada sección permite al usuario acceder a un listado completo de los volúmenes. Además, es posible acceder a la información propia de cada tomo, en la cual se muestra la portada, el número de páginas, el precio y la colección de cada volumen.

Por otro lado, con el objetivo de mostrar las diversas colecciones comenzadas o finalizadas, en el módulo «Intermediaria» se ha hecho uso del método «obtenerColeccion()» para obtener de cada volumen su atributo «colección». Mediante esta información, se puede realizar

una petición de tipo GET a la API REST para conseguir todos los datos relativos a dicha colección. Al contrario que la información mostrada en una colección al realizar una consulta por el filtro de búsqueda, en «Mi Biblioteca» se muestran también los volúmenes adquiridos hasta el momento de la colección seleccionada y el porcentaje de volúmenes adquiridos respecto al número de volúmenes publicados actualmente en España.

9.6 Novedades mensuales

Anteriormente, se ha mencionado que en la ventana principal de la aplicación se encontraba un botón con el nombre de Novedades. Al pulsar este botón, se le muestra al usuario todas las novedades por publicar por las diferentes editoriales. Esto supone crear un nuevo componente en el proyecto denominado Novedades. Para obtener en formato JSON todas las novedades, es necesario emplear el método «obtenerDatosNovedades()» del servicio «ProveedorAPI». Este realiza una petición GET a la siguiente URL: «http://127.0.0.1:5002/api/v1/novedades». Una vez llamada la función en el componente, se hace uso del método «subscribe()» para guardar el array resultante en la variable «datos_novedades». Para concluir, en el fichero HTML se recorre la variable mediante la directriz «*ngFor» y se accede a los atributos: fecha y portada. Esta información es estructurada por medio de un «ion-grid» y estableciendo tres columnas para representar todas las novedades.

10. Resultados

En este apartado, se mostrarán una serie de capturas tomadas a través del navegador web Google Chrome y con un dispositivo virtual, el smartphone iPhone 8 Plus, haciendo uso de la aplicación móvil Ikari. Además de realizar un test de funcionalidad para ilustrar si se han cumplido todos los casos de uso propuestos al inicio del proyecto.

10.1 Capturas de pantalla

10.1.1 Inicio de sesión

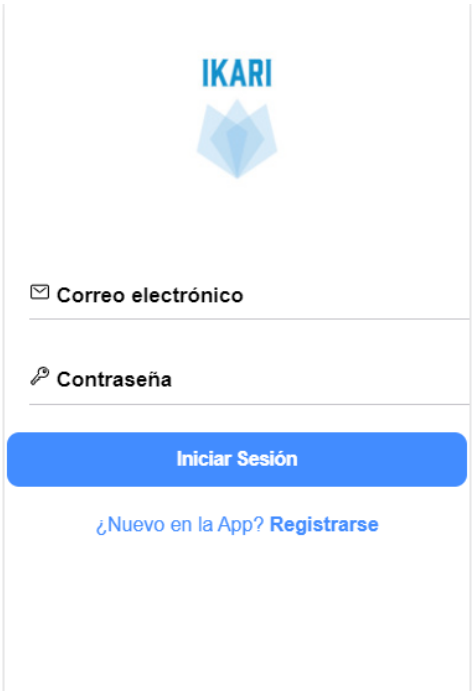

Figura 56. Pantalla de inicio de sesión

10.1.2 Registrarse

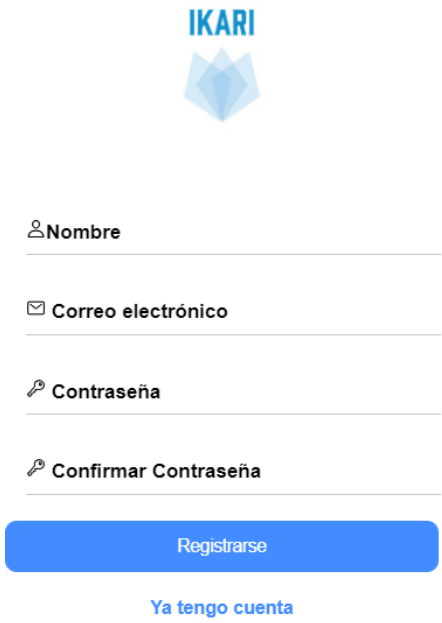

Figura 57. Ventana de registro

10.1.3 Ventana principal

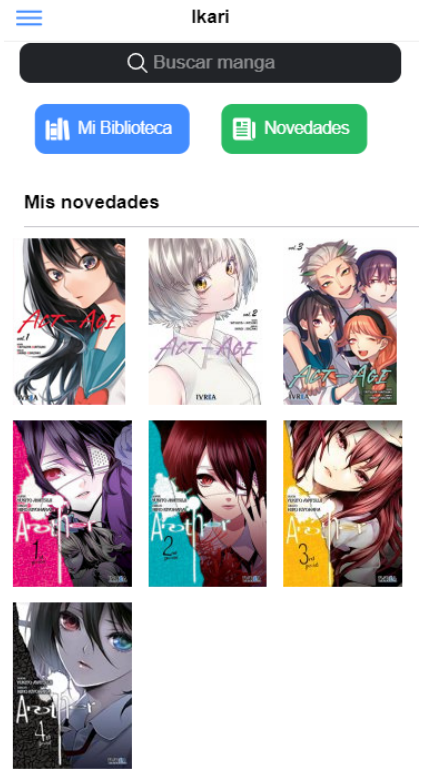

Figura 58. Pantalla principal de Ikari

10.1.4 Filtro de búsqueda

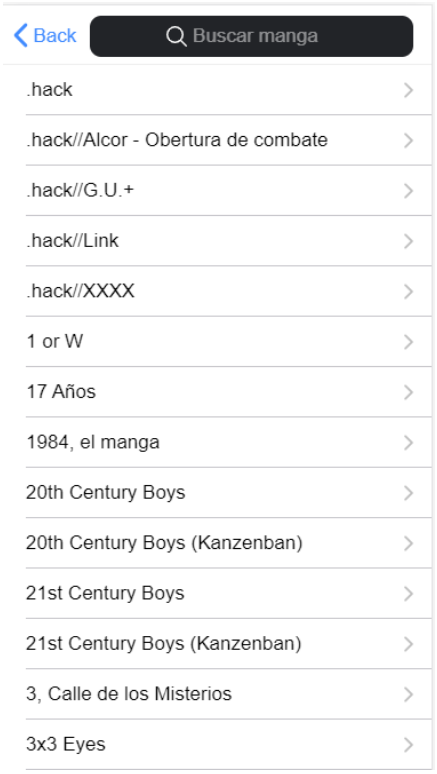

Figura 59. Filtro de búsqueda

10.1.5 Novedades mensuales

Figura 60. Novedades publicadas por las editoriales mensualmente

10.1.6 Ficha de una colección

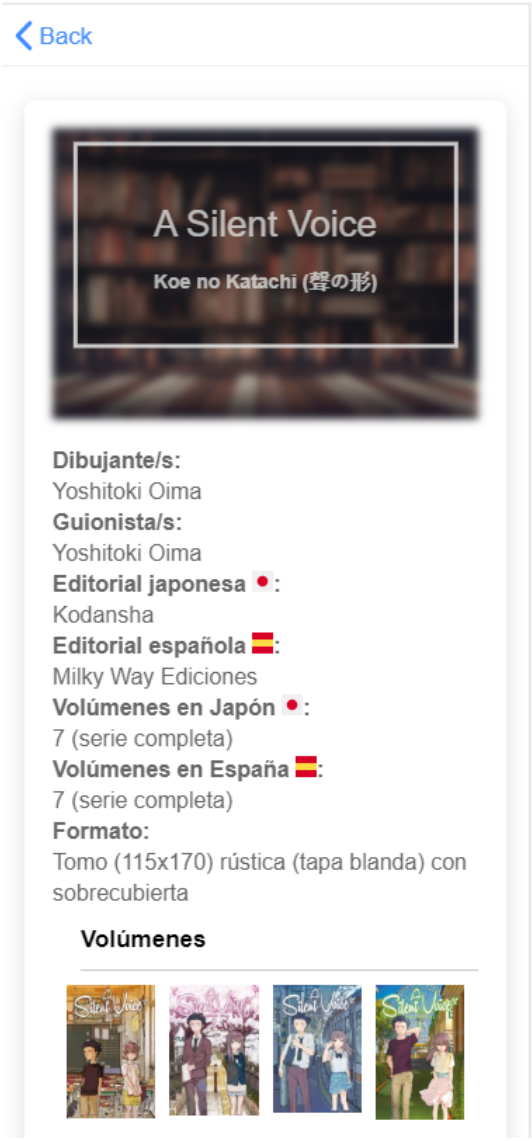

Figura 61. *Información sobre una colección*

10.1.7 Ficha volumen

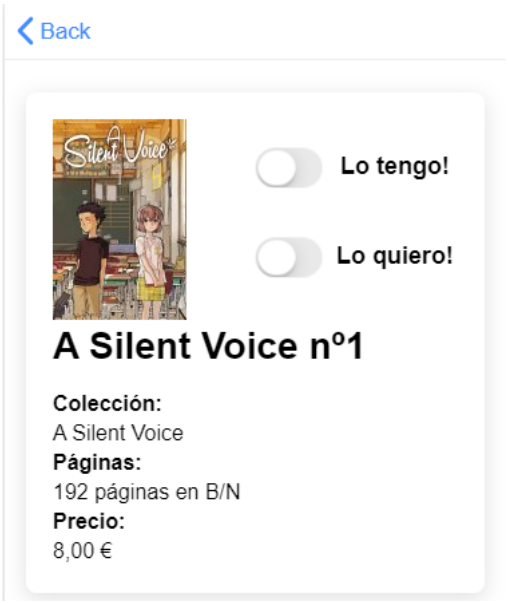

Figura 62. Información sobre un volumen

10.1.8 Lista volúmenes de una colección

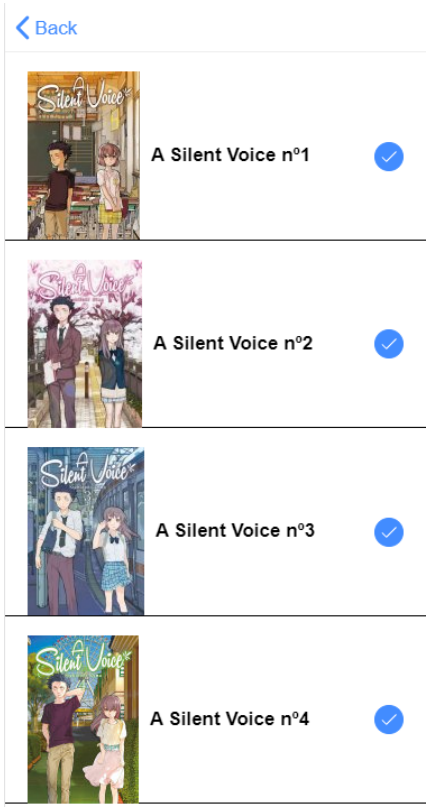

Figura 63. Volúmenes de una colección

10.1.9 Mi Biblioteca

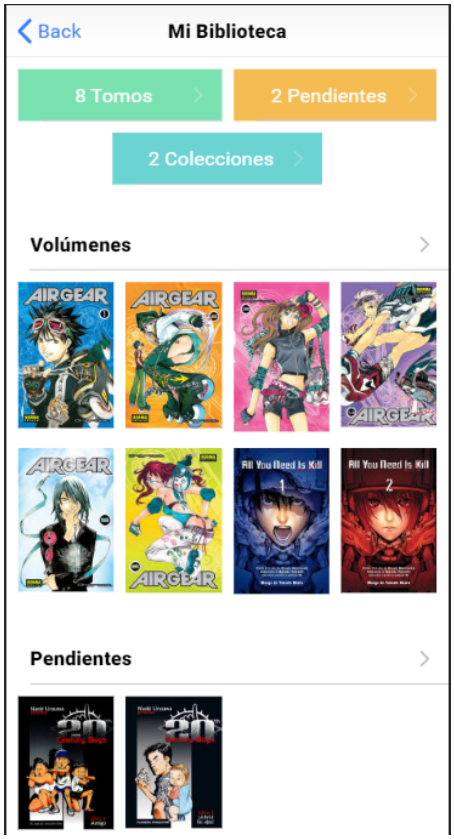

Figura 64. Ventana Mi Biblioteca

10.1.10 Reading Challenge

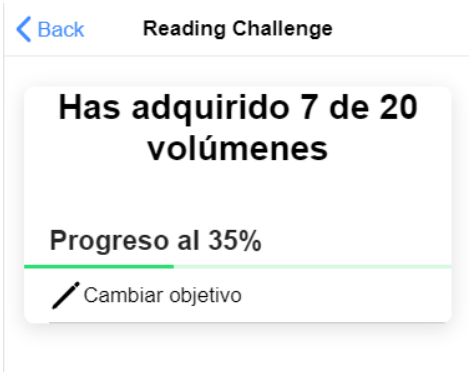

Figura 65. Reading Challenge

10.2 Test de funcionalidad

Se ha llevado a cabo un test de funcionalidad para comprobar si se han cumplido correctamente los objetivos establecidos previamente en los casos de uso. Los resultados son ilustrados a continuación por medio de una tabla:

Tabla 12. Test de funcionalidad de los casos de uso

Casos de Uso	Resultados
Caso de Uso 1	La aplicación comprueba correctamente las credenciales del usuario y le permite entrar a la aplicación. En caso de no ser correctas, se muestra un error en pantalla.
Caso de Uso 2	A través de la pantalla de inicio de sesión se accede sin problemas a la pantalla de registro. Se comprueba que al confirmar la contraseña coincida con el campo anterior. Una vez introducidos todos los campos, se permite al usuario entrar a la aplicación.
Caso de Uso 3	Al pulsar sobre el botón «Novedades» de la pantalla principal, se muestran todas las novedades publicadas por las editoriales mostrando las portadas y las fechas de las publicaciones.
Caso de Uso 4	Una vez seleccionada la colección en el filtro de búsqueda, se muestra de manera estructurada la información detallada de la colección. Además de todos los volúmenes que componen la colección. También se muestra correctamente toda la información referente a una colección, en el apartado Mi Biblioteca. Además de mostrar los volúmenes adquiridos y el porcentaje de volúmenes adquiridos respecto a los volúmenes publicados hasta el momento en España.
Caso de Uso 5	Al buscar una colección mediante el filtro de búsqueda, aparecen un conjunto de volúmenes asociados. Al pulsar sobre la imagen de un volumen concreto, aparece una nueva ventana con toda la información del tomo. Por otro lado, en la ventana de Mi Biblioteca, al pulsar sobre un volumen adquirido o pendiente se muestra también sus datos.
Caso de Uso 6	Al mostrar el listado completo de volúmenes de una colección y seleccionar uno o varios volúmenes, se añaden correctamente a la biblioteca del usuario. De manera similar ocurre cuando el usuario entra en un volumen concreto y selecciona «Lo tengo!».
Caso de Uso 7	Al pulsar sobre un volumen concreto y seleccionar la opción «Lo quiero!», se añade correctamente el volumen a la biblioteca personal del usuario.
Caso de Uso 8	Para poder visualizar correctamente la lista de volúmenes deseados en la pantalla principal, es necesario acceder previamente a la ventana de «Mi Biblioteca» para actualizar el contenido. En la

	ventana «Mi Biblioteca» se muestran los volúmenes sin problema.
Caso de Uso 9	Se muestra correctamente el porcentaje y la barra de progreso del objetivo establecido por el usuario.
Caso de Uso 10	Al pulsar en el botón de «Cerrar Sesión» situado en el menú de la aplicación, se devuelve al usuario a la pantalla de inicio sesión sin ningún inconveniente aparente.

11. Conclusiones

En este proyecto se ha desarrollado una aplicación móvil con el objetivo de mostrar todas las colecciones publicadas en España y las novedades anunciadas por las distintas editoriales. Para llevar esto a cabo, se han afrontado varios retos durante el desarrollo a fin de explorar nuevas tecnologías e incrementar los conocimientos adquiridos durante el grado.

En primer lugar, hacer uso de técnicas de Web scraping implica programar en un lenguaje de programación desconocido como es Python. Sin embargo, gracias a las bases adquiridas en diferentes asignaturas de la carrera, el tiempo de aprendizaje ha resultado ser más reducido respecto a las estimaciones iniciales. En contraste, la comunicación establecida entre la base de datos y los scripts de Python resultó costosa al principio debido a las claves ajenas entre tablas. En segundo lugar, tomar los conocimientos teóricos acerca del desarrollo de una API REST y desarrollarla desde cero mediante el «framework» de Python, Flask. Finalmente, el objetivo principal del proyecto se ha alcanzado llevando a cabo una aplicación móvil implementada en Angular y con almacenamiento de datos en Firebase.

El resultado obtenido son varios componentes comunicados entre sí para dar lugar a un servicio transparente y unificado. Asimismo, el aprendizaje de nuevas tecnologías como Python o Angular permite acceder a nuevas oportunidades laborales.

12. Relación con los estudios cursados

Los estudios cursados en el grado de ingeniería informática han influenciado al proyecto de la siguiente manera:

En primer lugar, la especificación de los diversos casos de uso y del diagrama de clases pone a prueba lo aprendiendo en la asignatura Ingeniería del Software. Al ser una asignatura dada hace un par de años, hubo que repasar algunos conceptos acerca de las relaciones establecidas en el diagrama de clases.

Por otro lado, el desarrollo y población de la base de datos se ha llevado a cabo tomando como referencia conocimientos adquiridos en las asignaturas Bases de datos y sistemas de información, y Tecnología de bases de datos. Esta última pertenece a la especialidad cursada, Tecnologías software de la información. Al ser una base de datos sin demasiada complejidad estructural o relacional, los conocimientos aprendidos fueron suficientes para resolver cualquier duda.

También se han puesto en práctica los conocimientos teóricos aprendidos en Integración de aplicaciones para un correcto desarrollo de una API RESTful. La dificultad a tener en cuenta era utilizar un «framework» como Flask para su puesta en marcha, aunque siguiendo en todo momento cada una de las pautas expuestas en la asignatura.

Para el desarrollo de la parte visual y funcional de la aplicación se ha hecho uso de los conocimientos adquiridos en la asignatura Desarrollo Web. En concreto, HTML, CSS y JavaScript.

Por último, las asignaturas de Desarrollo centrado en el usuario e Interfaces persona computador sentaron las bases para establecer una serie de principios de diseño en la interfaz de usuario de la aplicación móvil. También se aplicaron en el diseño final algunas leyes de Gestalt como la ley de la similitud o la ley de la proximidad.

13. Trabajos futuros

Esta primera versión de Ikari cumple con los objetivos planteados en un principio, sin embargo este precedente permite realizar mejoras en ciertos aspectos tales como introducir nuevas funcionalidades o implementar nuevas tecnologías en beneficio de los usuarios y las editoriales españolas.

En primer lugar, el tiempo de carga de todas las colecciones en el filtro de búsqueda debería ser más reducido, de manera que una búsqueda pudiera hacerse de manera casi instantánea. Por otra parte, para mostrar la lista de tomos deseados en la pantalla principal es necesario primero acceder al apartado de Mi biblioteca para cargar los volúmenes.

De igual manera, se podrían incluir una serie de nuevas funcionalidades en la aplicación con la intención de proporcionar el mayor número de facilidades y de información a los usuarios. Mediante este pretexto, se podrían añadir funcionalidades propias de una red social para puntuar colecciones o volúmenes. Además de poder añadir comentarios o llevar un seguimiento de las colecciones adquiridas por otros usuarios. Otra funcionalidad importante que comparten otras aplicaciones del mercado sería incorporar un escáner de código de barras con el objetivo de obtener el código ISBN de los volúmenes. Para finalizar, sería conveniente introducir estadísticas mensuales acerca de las diversas colecciones adquiridas.

Algo a realizar una vez implementadas todas las nuevas funcionalidades y resuelto los posibles cuellos de botella, debería ser implementar un servicio basado en Big Data por el cuál tanto usuarios como editoriales se verían beneficiados. Mediante este servicio se pretende reducir las posibles pérdidas económicas de las editoriales y predecir mejor la evolución del mercado en España. En muchos casos, se importan obras que en Japón pueden estar generando un volumen considerable de ganancias, pero que en España no funcionan de igual manera y no venden lo suficiente para hacer frente a las pérdidas. Por medio de las fuentes de información adecuadas, se podría analizar la evolución del mercado y su situación actual para obtener una serie de conclusiones acerca de todas las colecciones publicadas y los beneficios generados. Esto indirectamente también repercute en las demografías que se hacen uso en la industria del manga. Por otro lado, las redes sociales juegan hoy en día un papel fundamental para conocer la opinión de los lectores. Mediante el Big Data se podría recoger información acerca de las colecciones más solicitadas por la comunidad o recoger opiniones sobre colecciones ya publicadas.

Referencias

[1] Bartholomew, Daniel. (2014). *MariaDB Cookbook*. Packt Publishing. *Preface*. Recuperado el 18 de abril de 2020 de http://1.droppdf.com/files/hHBD6/packt-publishing-mariadb-cookbook-2014.pdf

[2] Bernabé, Marc. (2019, diciembre 30). *Estadísticas manga 2019*. Mangaland. Recuperado el 13 de febrero de 2020 de http://www.mangaland.es/2019/12/estadisticas-manga-2019/

[3] Cardona Pérez, Manuel. (2016, octubre 14). *Firebase, qué es y para qué sirve la plataforma de Google*. Recuperado el 19 de abril de 2020 de https://www.iebschool.com/blog/firebase-que-es-para-que-sirve-la-plataforma-desarroladores-google-seo-sem/

[4] Domingo Muñoz, José. (2017, noviembre 17). *Qué es Flask*. Recuperado el 18 de abril de 2020 de https://openwebinars.net/blog/que-es-flask/

[5] Fielding, Roy Thomas. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Chapter 5). Recuperado el 20 de abril de 2020 de https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[6] Ganesan, Prahbu. (2017, noviembre 16). *What is XAMPP?* Recuperado el 18 de abril de 2020 de https://www.wpblogx.com/what-is-xampp/

[7] *Goodreads*. (s. f.). Google Play. Recuperado el 4 de marzo de 2020 de https://play.google.com/store/apps/details?id=com.goodreads&hl=es

[8] Griffith, Chris. (2017). *Mobile App Development with Ionic, Revised Edition. Chapter 1: Hybrid Mobile Apps.* Recuperado el 19 de abril de 2020 de https://learning.oreilly.com/library/view/mobile-app-development/9781491998113/ch01.html#hybrid_mobile_apps

[9] Gutiérrez, Pedro. (2013). Fundamente de las bases de datos: Modelo entidad-relación. *November 5, 2013*.

Recuperado el 9 de marzo de 2020 de https://www.genbeta.com/desarrollo/fundamento-de-las-bases-de-datos-modelo-entidad-relacion

[10] Henderson, Michael. (2019, mayo 13). *What is NodeJS and Why You need to learn it*.

Recuperado el 18 de abril de 2020 de https://medium.com/@michaelhenderson/what-is-nodejs-and-why-you-need-to-learn-it-f0760ba9a76a

[11] IAN. (2016, mayo 29). *What is MySQL Workbench?*

Recuperado el 19 de abril de 2020 de https://database.guide/what-is-mysql-workbench/

[12] MDN contributors. (2019, marzo 18). *CORS errors*.

Recuperado el 25 de abril de 2020 de https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors

[13] Mitchell, Ryan. (2018). *Web Scraping with Python, 2nd Edition*. O'Reilly Media, Inc. *Preface.*

Recuperado el 22 de abril de 2020 de https://learning.oreilly.com/library/view/web-scraping-with/9781491985564/

[14] Shannon, Ross. (2012, agosto 21). *What is HTML?*

Recuperado el 18 de abril de 2020 de https://www.yourhtmlsource.com/starthere/whatishtml.html

[15] Vanderkam, Dan. (2019). *Effective TypeScript*. *Chapter 1. Getting to Know TypeScript.*

Recuperado el 19 de abril de 2020 de https://learning.oreilly.com/library/view/effective-typescript/9781492053736/ch01.html#ch-intro

[16] Visual Studio Code. (s. f.).

Recuperado el 19 de abril de 2020 de https://code.visualstudio.com

[17] Whakoom. (s. f.). Google Play.

Recuperado el 4 de marzo de 2020 de https://play.google.com/store/apps/details?id=com.whakoom.app&hl=es

[18] What is Python? Executive Summary. (s. f.).

Recuperado el 18 de abril de 2020 de https://www.python.org/doc/essays/blurb/

[19] Wieringa, Jeri. (2012, diciembre 30). *Intro to BeautifulSoup*.

Recuperado el 24 de abril de 2020 de https://programminghistorian.org/en/lessons/intro-to-beautiful-soup

Apéndice A
Glosario

Siglas

API: *Application Program Interface*

REST: *Representational State Transfer*

ISBN: *International Standard Book Number*

UML: *Unified Modeling Language*

ISO: *International Organization for Standardization*

HTTP: *Hypertext Transfer Protocol*

URL: *Uniform Resource Locator*

SQL: *Structured Query Language*

URI: *Uniform Resource Identifier*

XML: *Extensible Markup Language*

MVW: *Model-view-viewmodel*

MVC: *Model-view-controller*

DOM: *Document Object Model*

SGBD: *Sistema gestor de bases de datos*

CORS: *Cross-Origin Resource Sharing*

JSON: *JavaScript Object Notation*

CLI: *Command-Line Interface*

UID: *Unique Identifier*