PuzzleCollector

Proyecto Final de Grado

Autor: Juan Antonio Nicolás Pérez

Grado Superior de Desarrollo de Aplicaciones Multiplataforma

Curso 2021-2023

ÍNDICE

1. INTRODUCCIÓN

1.1 Resumen y justificación del proyecto

Existen varias aplicaciones en el mercado capaces de ayudar al usuario a gestionar y mantener algunos de sus hobbies al día, cómo pueden ser las aplicaciones que registran las series vistas o aquellas capaces de registrar datos relacionados con el deporte y la salud.

A pesar de que existe un gran número de aplicaciones para gestionar hobbies de usuarios, este dato no implica que aún no haya mercado para estas, ya que es muy díficil cubrir todos y cada uno de los hobbies que actualmente existen.

Lo que quiero mostrar hoy es una de estas aplicaciones de gestión de hobbies, especializada en el pasatiempo de coleccionar puzzles. Aunque no lo parezca, existen muchos usuarios que llegan a tener tal cantidad de puzles en su colección que pueden llegar a olvidarse que poseían ese artículo.

PuzzleCollector es la solución que presento para poder gestionar este hobbie de manera más eficiente. Gracias a una base de datos, la cuál se actualiza gracias a la comunidad que usa la aplicación, esta herramienta para dispositivos Android es capaz de gestionar cualquier colección de puzles que pueda tener cualquier usuario.

Cabe destacar que la aplicación no solo almacena los típicos puzles “jigsaw” que todos conocemos, sino que también se compone de otros tipos cómo pueden ser los rompecabezas 3D, cubos de rubik y “escape-boxes”.

La aplicación también posee configuraciones para adaptarse a cualquier usuario, ya sea el famoso modo oscuro que ya todo el mundo usa, o un idioma secundario(Inglés) para poder usarse por cualquier persona que no sea Hispano hablante.

A lo largo de este documento se mostrarán más a fondo la aplicación, el servidor y la base de datos que componen todo el proyecto, además de los requisitos, las herramientas usadas en su desarrollo, la estructura de los distintos módulos y las pruebas realizadas para corroborar su correcto funcionamiento.

Abstract

There are several applications on the market capable of helping the user to manage and keep some of their hobbies up to date, such as applications that record the series watched or those capable of recording data related to sports and health.

Despite the fact that there are a large number of applications to manage user hobbies, this data does not imply that there is still no market for them, since it is very difficult to cover each and every one of the hobbies that currently exist.

What I want to show you today is one of these hobby management apps, specialized in the puzzle collecting hobby. Although it may not seem like it, there are many users who come to have such a number of puzzles in their collection that they may forget that they already had that item.

PuzzleCollector is the solution that I present to be able to manage this hobby more efficiently. Thanks to a database, which is updated thanks to the community that uses the application, this tool for Android devices is capable of managing any collection of puzzles that any user may have.

It should be noted that the application not only stores the typical "jigsaw" puzzles that we all know, but also includes other types such as 3D puzzles, rubik cubes and "escape-boxes".

The application also has settings to adapt to any user, be it the famous dark mode that everyone already uses, or a secondary language (English) to be used by anyone who is not a Spanish speaker.

Throughout this document, the application, the server and the database that make up the entire project will be shown in more depth, in addition to the requirements, the tools used in its development, the structure of the different modules and the tests carried out to verify its correct operation.

2. ANÁLISIS

El objetivo principal a cumplir por este proyecto es el de lograr que cualquier usuario pueda registrar sus puzles en la app, ya estén registrados en la BD o no, y así lograr llevar un mejor control de su colección sin necesidad de mucho esfuerzo.

2.1 Objetivos Específicos

En función del tiempo disponible para el proyecto, los objetivos se presentan en 2 categorías:

Objetivos Obligatorios:

- Sistema de DB para almacenar todos los datos de la app.
- Sistema de Registro de Usuarios(creación, edición y eliminación de cuentas).
- Sistema de Perfiles de Usuario(mantener la sesión iniciada, iniciar sesión desde cualquier dispositivo con la app y poder cambiar de cuenta).
- Vista principal de la aplicación(dónde se verán todos los puzzles actualmente añadidos a la DB, algunos eventos de interés y la propia colección de puzles del usuario).
- Panel de Usuario(editar datos del usuario, acceder a la lista de puzles deseados, acceder a la colección de puzles del usuario y poder sugerir nuevos puzles a añadir en la app).
- Panel de Configuración(editar el resto de datos del usuario, cambiar a modo oscuro, cambiar el idioma de la app).

Objetivos Deseables:

- Sistema de 2FA(Two Factor Authetication) usando el email del usuario.
- GUI para poder manejar con facilidad los datos de la DB desde el servidor.

2.2 Requisitos Funcionales

RF01	
NOMBRE:	Servidor Spring para conectar con la DB
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Un servidor realizado con la API de Spring que permitirá a la aplicación acceder a la base de datos mediante la librería Retrofit.
MODIFICACIONES:	-

RF02	
NOMBRE:	Login y Registro de Usuarios
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	La aplicación permitirá al usuario loguearse o registrarse tan solo introduciendo su nombre de usuario y una contraseña.
MODIFICACIONES:	En el registro también se permitirá escoger una imagen como icono de perfil.

RF03	
NOMBRE:	Panel de Configuración – General
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Desde este panel el usuario podrá realizar varias acciones que repercutirán en la aplicación, cómo cambiar el idioma, cambiar entre modo claro y oscuro.
MODIFICACIONES:	Se ha incluido una nueva opción que permite al usuario visualizar una pantalla de información de la app(Versión, Autor, etc.)

RF04	
NOMBRE:	Cierre de Sesión
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Desde el Panel de Configuración el usuario podrá cerrar sesión en la aplicación, lo que permitirá que otro usuario inicie sesión en ese dispositivo.
MODIFICACIONES:	-

RF05	
NOMBRE:	Eliminación de Cuentas
PRIORIDAD:	**MEDIA**
DESCRIPCIÓN:	Desde el panel de Configuración el usuario podrá eliminar su cuenta, lo cuál borrará todo rastro de ese usuario en la BD. Se pedirá la contraseña actual para poder realizar esta acción.
MODIFICACIONES:	-

RF06	
NOMBRE:	Edición de Datos del Usuario
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Desde el panel de Configuración el usuario podrá cambiar su nombre y su contraseña actual. Para cambiar la contraseña se solicitará la contraseña antigua antes de proceder con el cambio
MODIFICACIONES:	-

RF06	
NOMBRE:	Pantalla de Visualización de Datos – Puzles
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Pantalla dónde se muestran todos los artículos insertados en la DB. El usuario debe ser capaz de verlos e interactuar con ellos hasta cierto punto sin necesidad de una cuenta.
MODIFICACIONES:	-

RF07	
NOMBRE:	Pantalla de Visualización de Datos – Colección
PRIORIDAD:	**ALTA**
DESCRIPCIÓN:	Pantalla dónde se muestran todos los artículos que el usuario ha decidido añadir a su cuenta. Estos artículos también son interactivos. Es necesario crear un perfil en la app para poder acceder a esta función.
MODIFICACIONES:	Se han modificado algunos parámetros recibidos en esta pantalla para que difieran mínimamente con los de otras pantallas.

RF08	
NOMBRE:	Pantalla de Visualización de Datos – Lista de Deseados
PRIORIDAD:	**BAJA**
DESCRIPCIÓN:	Pantalla dónde se muestran todos los artículos que el usuario ha decidido añadir a su cuenta pero aún no posee físicamente. Es necesario crear un perfil en la app para poder acceder a esta función.
MODIFICACIONES:	-

RF09	
NOMBRE:	Formulario de Sugerencias
PRIORIDAD:	**MEDIA**
DESCRIPCIÓN:	Botón que permite al usuario acceder a un formulario online de recopilación de datos(Google Forms) para poder sugerir nuevos artículos en la app. No es necesario poseer una cuenta para usarlo.
MODIFICACIONES:	Se ha modificado el botón para acceder a distintos formularios dependiendo del idioma de la app.

2.3 Requisitos No Funcionales

RNF01	
NOMBRE:	Modo Oscuro
PRIORIDAD:	**MEDIA**
DESCRIPCIÓN:	Desde el panel de Configuración se podrá habilitar, deshabilitar o establecer el modo del sistema para ver la aplicación con unos colores más apagados que no dañen la vista.
MODIFICACIONES:	-

RNF02	
NOMBRE:	Cambiar Idioma
PRIORIDAD:	**MEDIA**
DESCRIPCIÓN:	Desde el panel de Configuración se podrá cambiar el idioma de la aplicación, permitiendo que sea entendible por más usuarios.
MODIFICACIONES:	-

RNF03	
NOMBRE:	Pantalla de Visualización de Datos – Eventos Pricipales
PRIORIDAD:	**BAJA**
DESCRIPCIÓN:	Pantalla dónde se muestran algunos eventos propuestos por la administración de la app. Estos eventos solo pueden ser creados solo por usuarios con el rango de “ADMIN”.
MODIFICACIONES:	-

RNF04	
NOMBRE:	Pantalla de Visualización de Datos – Eventos Pricipales
PRIORIDAD:	**BAJA**
DESCRIPCIÓN:	Pantalla dónde se muestran algunos eventos propuestos por la comunidad. Estos eventos pueden ser creados por cualquier usuario con cuenta en la app y sólo pueden crear 1 evento de esta categoría cada 7 días[1].
MODIFICACIONES:	Esta pantalla se ha acoplado a la pantalla de Eventos Principales y ahora estas dos solo ocupan una única pantalla.

[1]: Solo pueden ser creados cada 7 días debido a que tras transcurrir ese período el evento en sí es borrado del sistema.

2.4 Fases de Realización

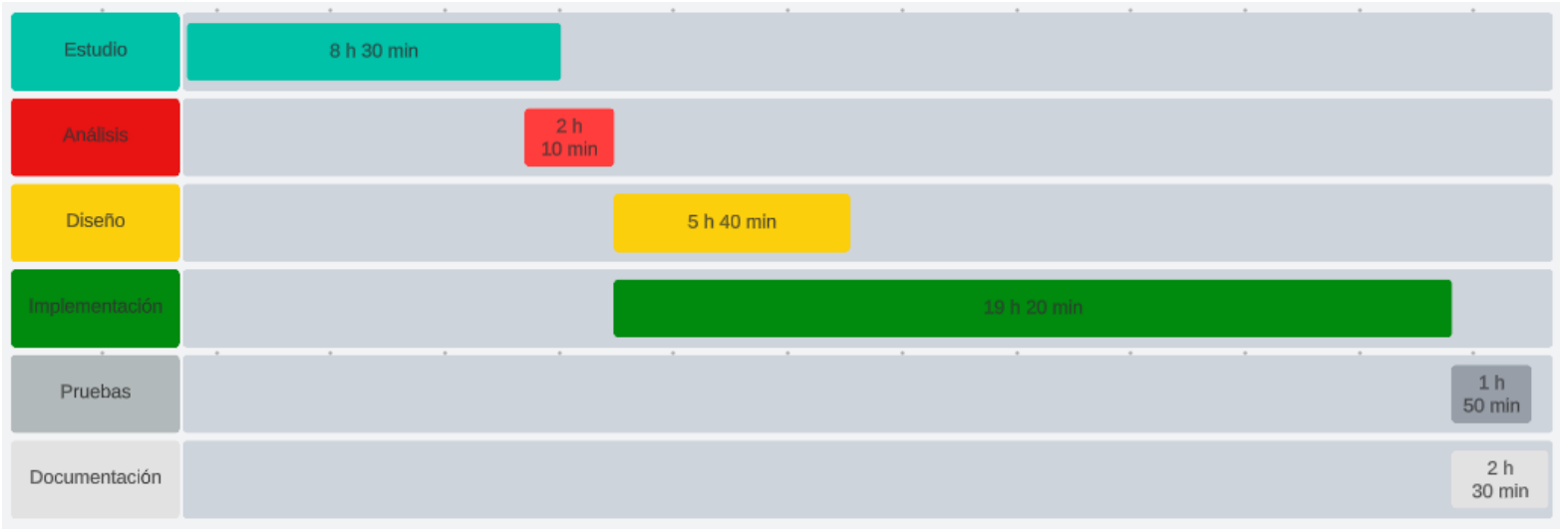
A continuación, se describen las fases seguidas a lo largo de la realización del proyecto:

- Estudio: Investigación sobre las tecnologías, lenguajes y clases a utilizar.
- Análisis: Definición de requisitos con el objetivo de definir con exactitud qué se desea construir.
- Diseño: Diseñar y aplicar la composición de la estructura de los módulos y los distintos diseños de las interfaces del proyecto. Esta fase también incluye la implementación de la DB.
- Implementación: Programación de los módulos servidor(Spring) y cliente(App Móvil).
- Pruebas: Una vez finalizadas las fases anteriores se proceden a realizar pruebas sobre todos los módulos que componen el proyecto.
- Documentación: Generación de Manuales y Documentación en el código para futuras mejoras y el correcto mantenimiento del proyecto.

2.4.1 Planificación y Duración de las Fases

Tras la definición de las fases, se han realizado una planificación y un diagrama que muestran la duración y el desglose de cada fase:

	Estudio	Análisis	Diseño	Implementación	Pruebas	Documentación
Marzo	X	X	X	X		
Abril			X	X		
Mayo				X	X	
Junio				X	X	X



2.5 Tecnologías y Herramientas Utilizadas

- Java SDK e IntelliJ Idea

IntelliJ Idea junto con el SDK de Java contiene las herramientas necesarias para desarrollar aplicaciones Java. Contiene un depurador potente y algunas herramientas IA que ayudan al desarrollo.

- Spring Boot API

Spring Boot es una API que funciona como un intermediario entre DB y aplicaciones web/móvil. Gracias a una extensa librería de Add-Ons es la API #1 para poder gestionar y usar BD en aplicaciones en mi opinión.

- MySQL Workbench

Herramienta usada para generar una base de datos MySql y poder incluir datos por defecto gracias a un archivo SQL. También se ha usado para generar un diagrama de la BD.

- Android SDK y Android Studio

Android Studio junto con Android SDK contiene las herramientas necesarias para el desarrollo de aplicaciones Android. Contiene emuladores y un depurador bastante potente.

- Librería Preferences

Líbrería de AndroidX usada para realizar el panel de Configuración.

- Material Design TextInput

Librería de Google usada para implementar campos de texto más fluidos y con un diseño más moderno.

- Retrofit

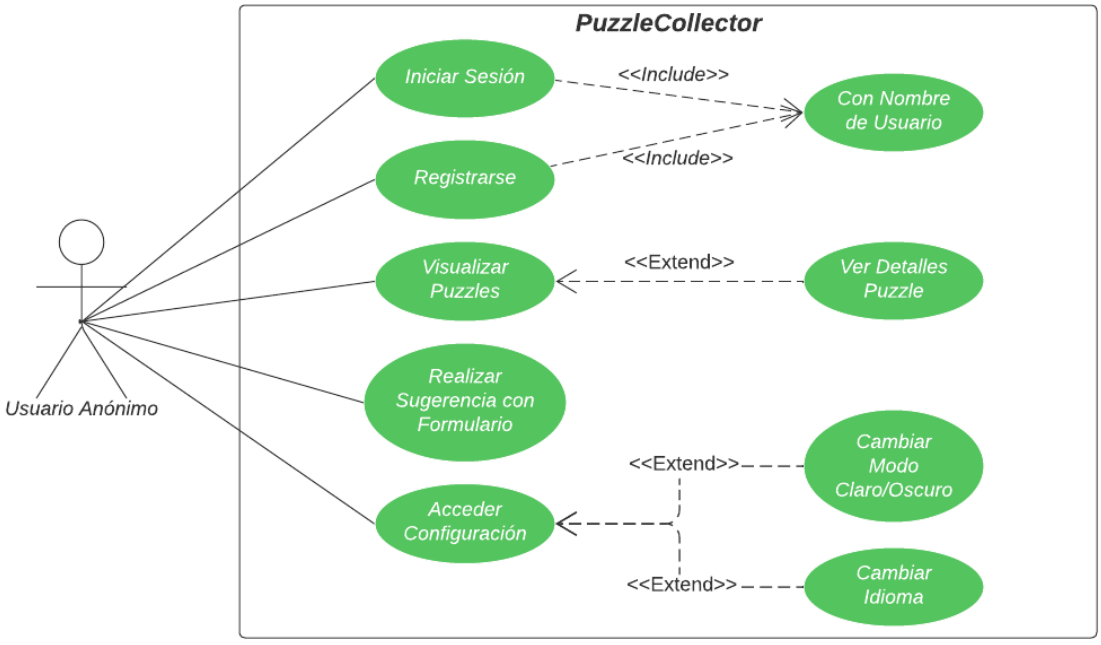
Líbrería usada en la App de Android para poder conectar con el servidor Spring Boot.

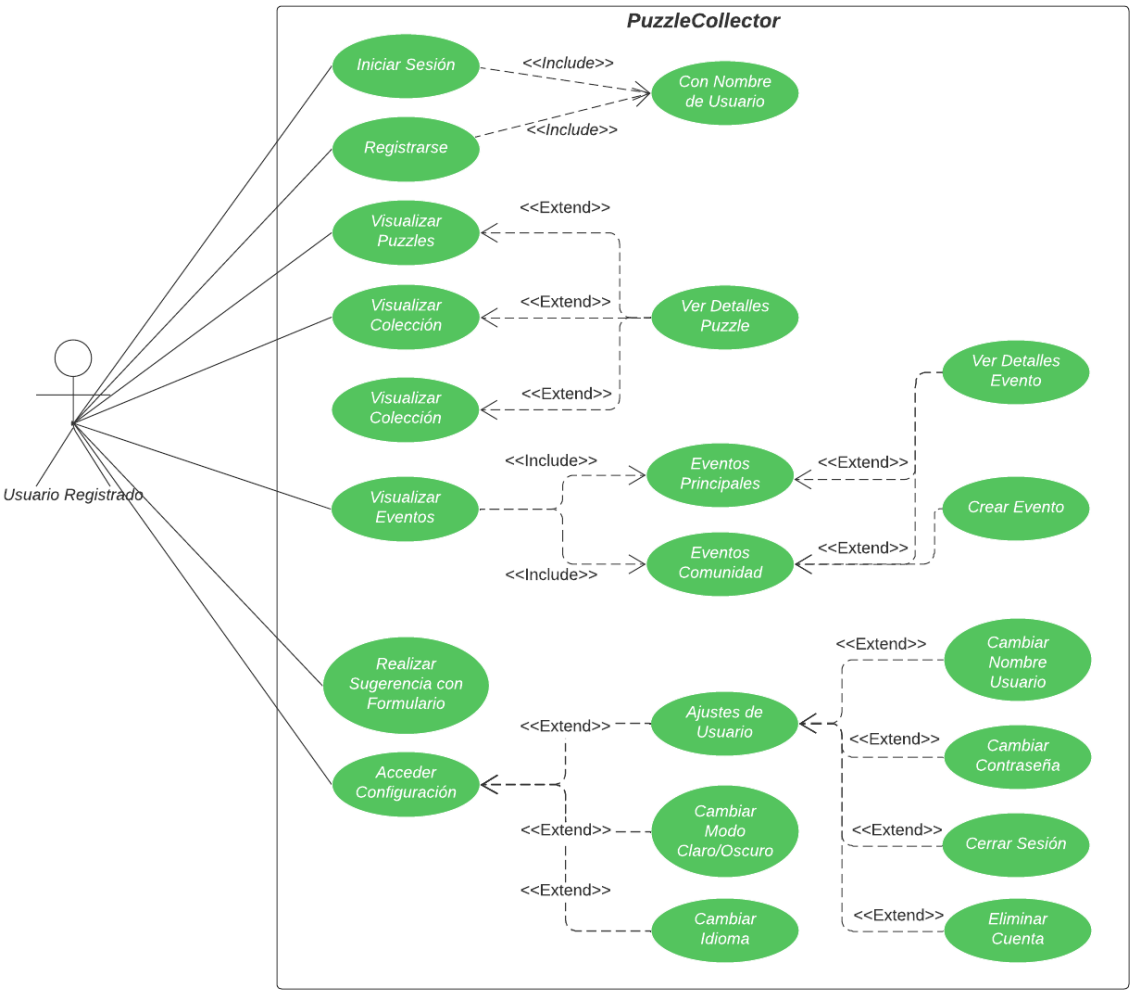
- LucidChart

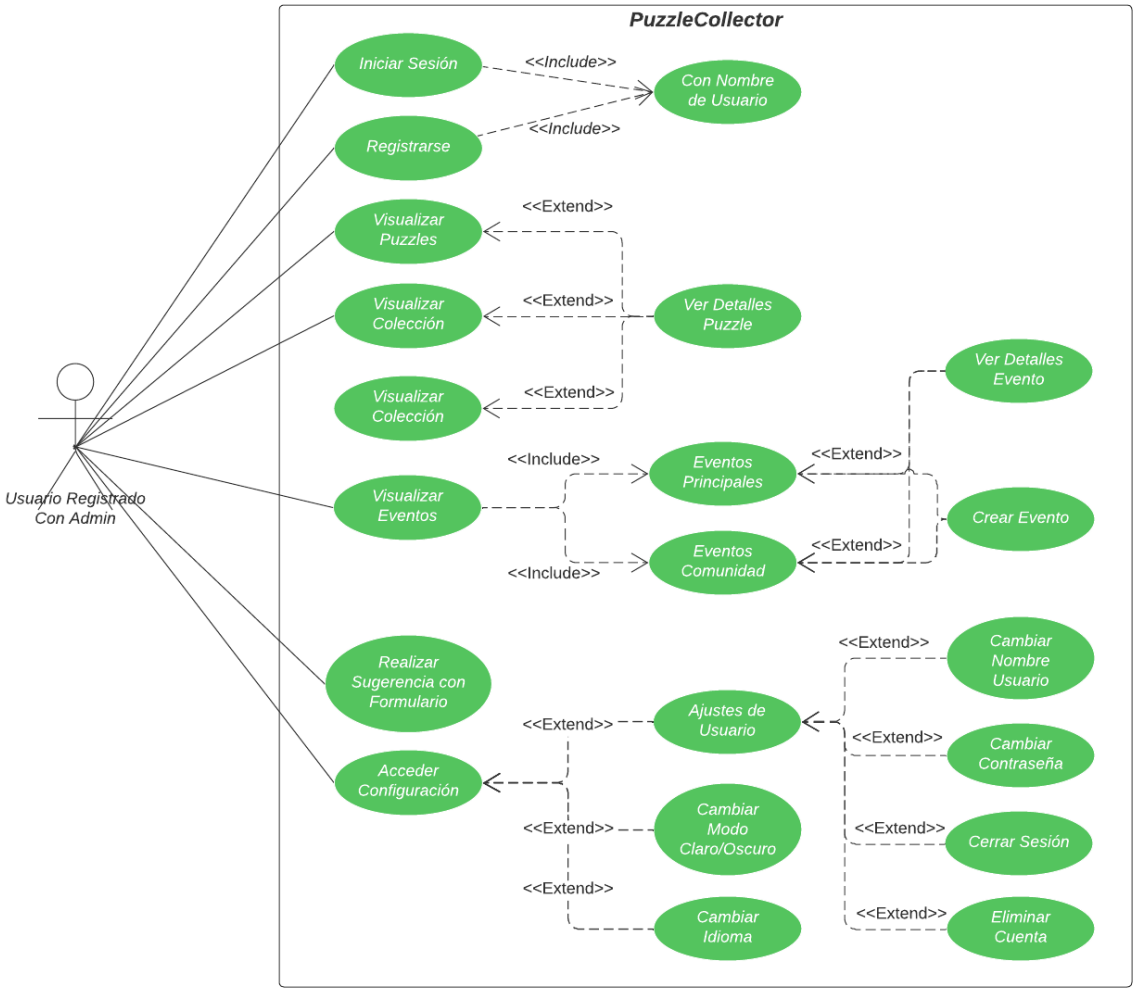
Herramienta usada para crear el Diagrama de Gantt y el diagrama UML de casos de uso de la app.

2.6 Casos de Uso

Los casos de uso permiten identificar el comportamiento de la aplicación y sus funcionalidades. Se han identificado tres actores: Usuario Anónimo, Usuario Resgistrado y Usuario Registrado Con Admin.



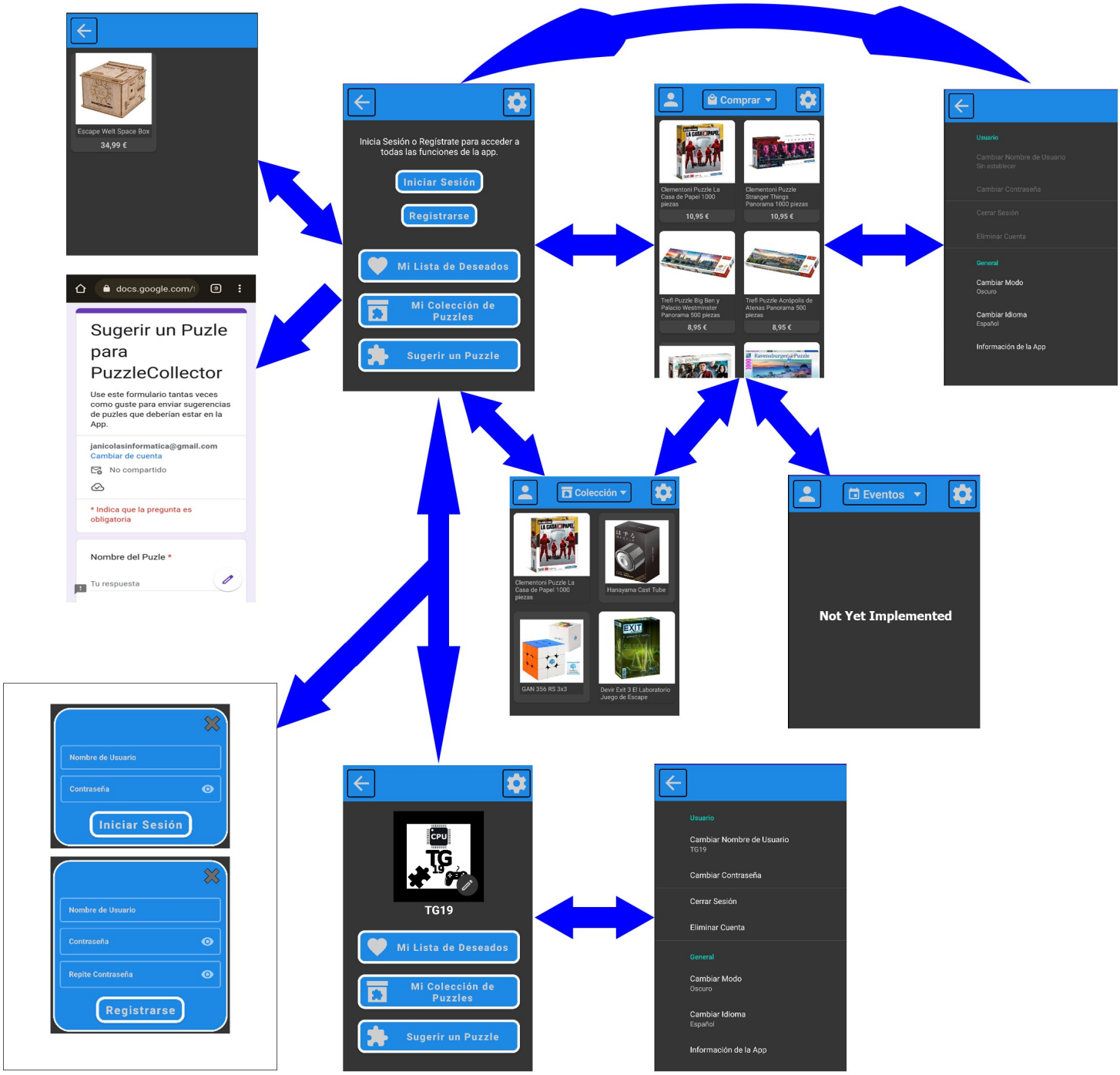

PuzzleCollector
Usuario Registrado
Iniciar Sesión
Registrarse
Visualizar Puzzles
Visualizar Colección
Visualizar Colección
Visualizar Eventos
Realizar Sugerencia con Formulario
Acceder Configuración
Con Nombre de Usuario
Ver Detalles Puzzle
Eventos Principales
Eventos Comunidad
Ajustes de Usuario
Cambiar Modo Claro/Oscuro
Cambiar Idioma
Ver Detalles Evento
Crear Evento
Cambiar Nombre Usuario
Cambiar Contraseña
Cerrar Sesión
Eliminar Cuenta
<<Include>>
<<Extend>>


PuzzleCollector
Iniciar Sesión
Registrarse
Visualizar Puzzles
Visualizar Colección
Visualizar Colección
Visualizar Eventos
Realizar Sugerencia con Formulario
Acceder Configuración
Usuario Registrado Con Admin
Con Nombre de Usuario
Ver Detalles Puzzle
Eventos Principales
Eventos Comunidad
Ajustes de Usuario
Cambiar Modo Claro/Oscuro
Cambiar Idioma
Ver Detalles Evento
Crear Evento
Cambiar Nombre Usuario
Cambiar Contraseña
Cerrar Sesión
Eliminar Cuenta
<<Include>>
<<Include>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Include>>
<<Include>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Extend>>
<<Extend>>

3. DISEÑO

3.1 Navegación entre ventanas

Este diagrama de la navegabilidad de la app se ha realizado a mano en Adobe Photoshop:

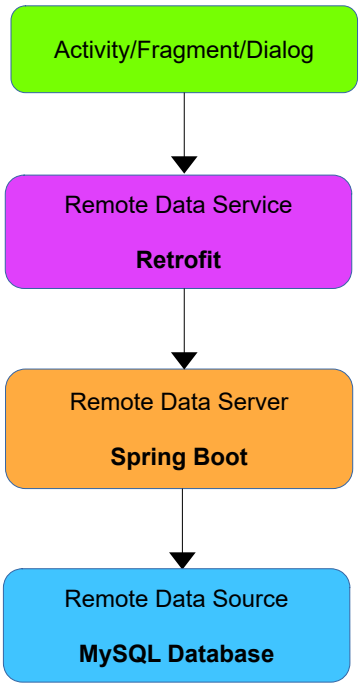


Tómese en cuenta que el diseño de las ventanas en el diagrama es con el modo Oscuro de la aplicación. También hay un modo Claro con fondo blanco y el azul es sustituido por el verde.

Se han utilizado iconos de Material Design, Vectr.com(Creados por mí) y FlatIcon. Para los diálogos de inicio de sesión, registro, etc. se han usado elementos de Google.AppCompat.

3.2 Arquitectura

El desarrollo de la aplicación se ha desarrollado siguiendo el siguiente modelo:



Cada componente mostrado en el modelo depende del componente situado en el nivel inferior al suyo. Esto significa que si, por ejemplo, fallase el componente de MySQL Database toda la aplicación con servidor incluido fallaría. Es por eso que este modelo implica que todos los componentes funcionen en sincronía y perfectamente.

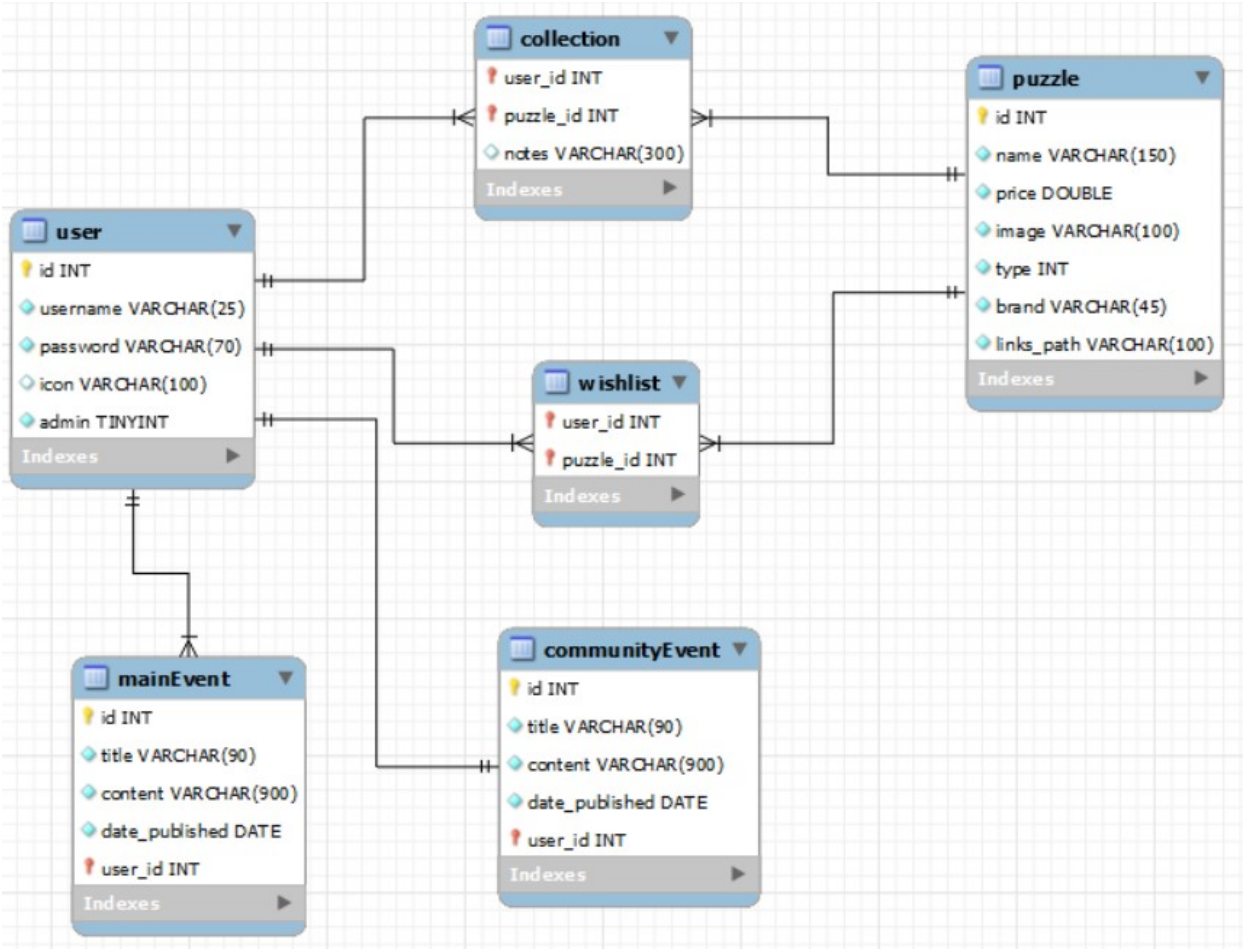
En el caso del componente superior, las activity, fragment y dialog, al fallar solo restringe el uso de la aplicación del usuario final. La administración puede seguir accediendo desde el Server al Source y así crear, modificar y eliminar datos de la app.

3.3 Modelo de Datos

MySQL Database es una base de datos MySQL, como indica su nombre, orientada al modelo entidad-relación. Esto implica que cada una de las tablas puede comunicarse directamente con otra que se considere necesaria gracias a las relaciones entre estas.

Todos los datos son generados a partir de documentos “.sql” y estos son almacenados como archivos usados por el motor de la base de datos.

El diagrama que representa las relaciones, tablas y los atributos de estas se presenta a continuación:



Tenemos 4 entidades principales:

- **User:**

Dentro de esta entidad encontramos atributos como son “username” y “password”, los cuáles permiten que un usuario inicie sesión en la app, “icon”, el cuál almacena la ruta del servidor dónde se alamcenan los iconos de usuarios y “admin”, un atributo que permite tener más funciones dentro de la app.

- **Puzzle:**

Dentro de esta entidad encontramos atributos como “name”, que definen el nombre del artículo a mostrar, “price” para el precio, “image” que almacena otra ruta para acceder a las imágenes de los puzles, “type” que define el tipo de puzle en cuestión, “brand” que especifica la marca, “description” que define la ruta del archivo que contiene la descripción del producto y “links_path”, otra

variable que almacena una ruta al archivo que contiene enlaces de compra y tutoriales.

- **MainEvent y CommunityEvent:**

Dentro de estas entidades podemos encontrar atributos que definen el título del evento “title”, el contenido de este “content” y la fecha en la que fue publicado “date_published”. La única diferencia entre ambas entidades es que la entidad MainEvent permite que un usuario pueda crear varios objetos mientras que la entidad CommunityEvent solo permite crear un objeto de esta por usuario.

Luego podemos observar que existen otras 2 entidades qué son dependientes de otras principales para poder subexistir:

- **Collection:**

Esta entidad generada por la relación M-M entre “puzzle” y “user” permite que los usuarios puedan guardar puzles en su colección de puzles. También gracias al atributo “notes” se permitirá a los usuarios almacenar algunas notas sobre el puzle en cuestión.

- **WishList:**

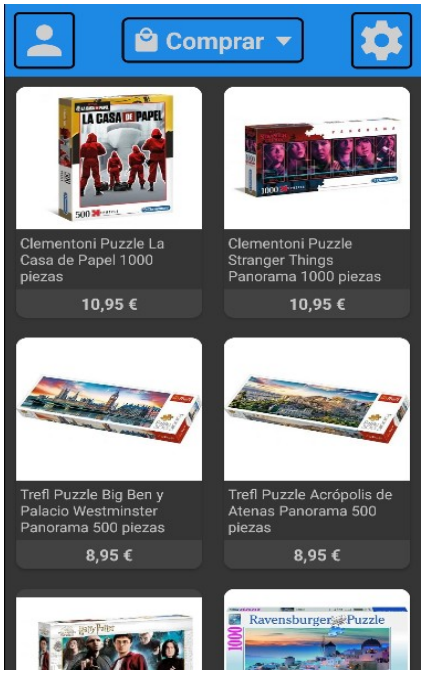
Esta entidad generada por la relación M-M entre “puzzle” y “user” permite que los usuarios puedan guardar puzles en su lista de deseados.

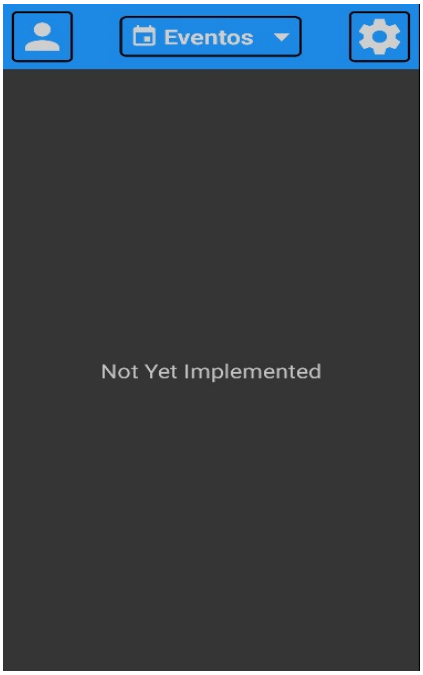
3.4 Vistas

A continuación se mostrarán algunas capturas de la aplicación en funcionamiento:

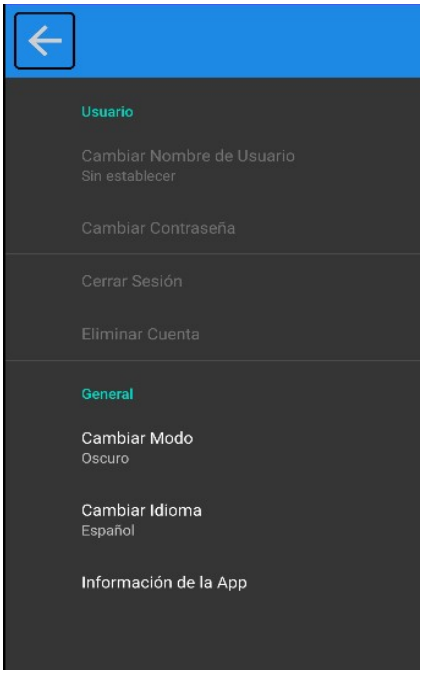
3.4.1 Panel de Usuarios

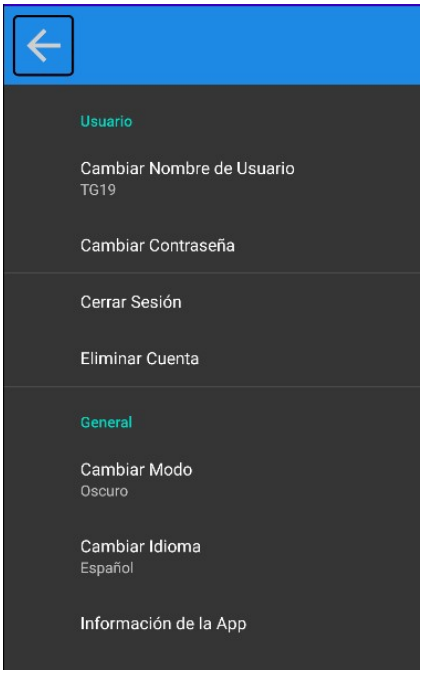
3.4.2 Paneles de Compra, Eventos y Colección





3.4.3 Panel de Configuración





4. IMPLEMENTACIÓN

4.1 Configuración del entorno de desarrollo

Para la implementación de la aplicación se ha empleado un equipo:

- Portátil AMD Ryzen 7 con 16GB de RAM.

Lo primero fue instalar Android Studio, MySQL Workbench, IntelliJ Idea y las herramientas del ambos SDK, junto a descargar el paquete inicializado de Spring Boot y Git para el control de versiones (Adobe Photoshop para el diseño de algunos iconos).

- Android Studio 17.0.6
- IntelliJ Idea Community Edition 17.0.6
- MySQL Workbench 8.0.31
- Spring Boot 3.0.7
- Git 2.33.0

4.2 Desarrollo

Una vez configurados los entornos de desarrollo se procedió a abrir Spring Boot desde IntelliJ, se generó un nuevo archivo de MySQL Workbench para generación gráfica de BD y se creó un proyecto nuevo en Android Studio sin ninguna plantilla(Actividad en Blanco)

4.2.1 Crear y Popular la Base de Datos

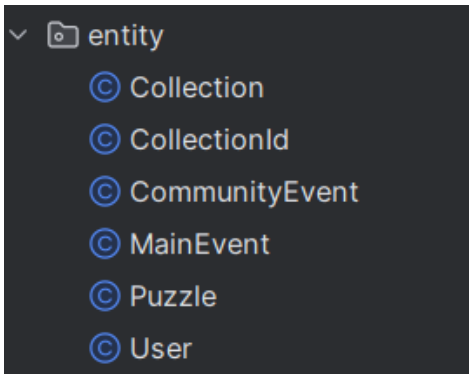
Lo primero fue crear el esquema de la BD que se ha mostrado anteriormente en este documento. Una vez generado el archivo “.sql” que permite generarla se procedió a guardar el archivo “.mwb” que permite guardar el esquema de la BD para su posterior documentación.

Por último, se generó un archivo “.sql” en blanco para poder popular(rellenar, introducir) los datos de prueba o básicos en la BD para realizar las correspondientes pruebas más adelante.

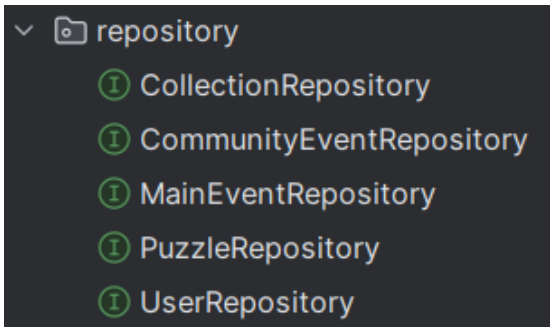
4.2.2 Abrir y Configurar Spring Boot

Una vez ya generé la BD necesaria para mi proyecto me dispuse a configurar el servidor de Spring Boot para que actúase como intermediario entre MySQL y Android. Procederé ahora a definir los distintos paquetes y clases que conforman el servidor:

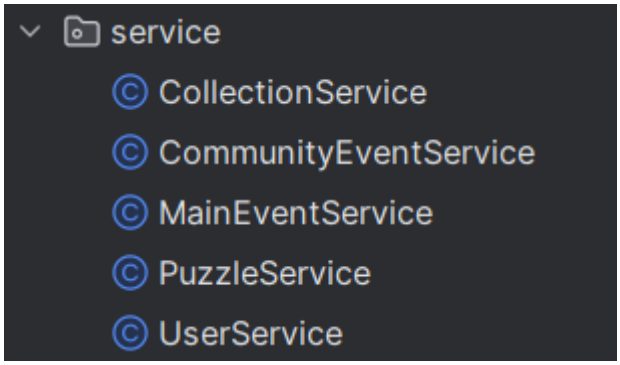
- El paquete entity, que contiene los modelos o entidades que permiten la lectura y escritura en lenguaje humano de los datos de la DB:



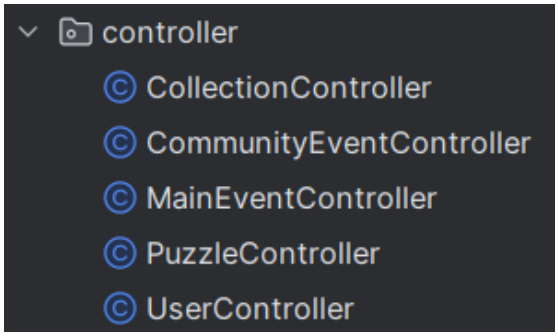
- El siguiente paquete, repository, contiene todas las interfaces que usa Spring Boot para generar las Query o Consultas necesarias para poder extraer y escribir datos en la DB:



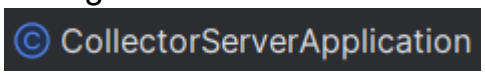
- El paquete service, que implementa las Query o Consultas específicadas en las interfaces del paquete repository:



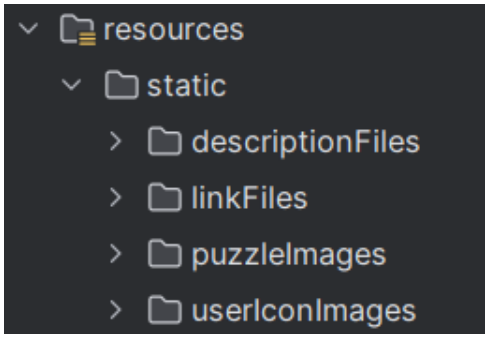
- El paquete controller, que contiene los Rest Controller necesarios para asignar las Consultas a una dirección de red para que la app pueda acceder a estas:



- La clase Application, auto generada por Spring Boot y es la encargada de arrancar el servidor:



- Y el paquete res/static/ que contiene todos los archivos e imágenes que se requieren para completar algunas entidades:



4.2.3 Configurar Retrofit

Retrofit es el paquete que usaré para poder comunicarme con Spring Boot y así poder extraer y escribir datos en la BD para que la App funcione.

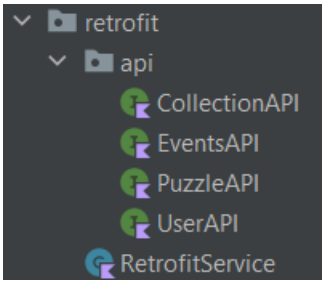
Lo primero que hice fue entrar al Manifest de mi aplicación para declarar el uso del servicio de conexión a internet, ya que sin esto, la app no podría conectarse a internet ergo, no podría comunicarse con el servidor.

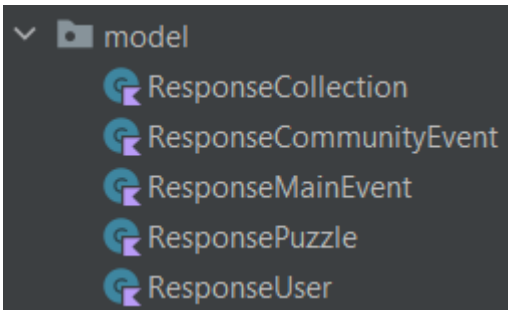
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Lo siguiente que hice fue implementar, desde el archivo “build.gradle”, el paquete de Retrofit y el paquete GSON, una utilidad necesaria para poder “iterpretar” la información recibida por el servidor, la cuál se encuentra en formato JSON.

```
dependencies {
    implementation 'androidx.core:core-ktx:1.10.1'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.preference:preference:1.2.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
}
```

Por último procedí a crear las clases necesarias para que Retrofit funcione. Creando la clase RetrofitService, unas cuántas interfaces API para realizar Query(Consultas) y unos modelos que servirán como molde para inyectar los datos recibidos por el servidor acabé de configurar esta utilidad.

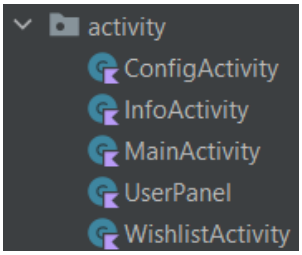


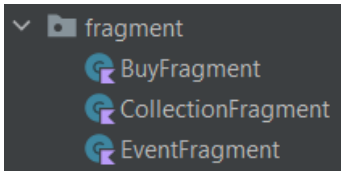


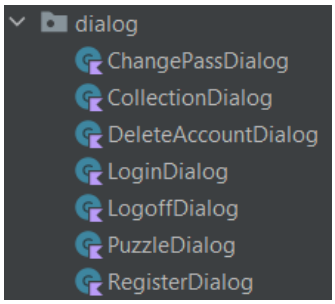
4.2.4 Activities, Fragments y Dialogs

Una vez terminé con Retrofit, me dispuse a configurar las pantallas de mi aplicación. Debido a que me disgusta usar las ActionBar por defecto de Android quise hacer la mía propia y con mi propia navegación entre pantallas.

Las activity se encargan de dotar a la app de funcionalidad y de manejar la navegación entre las distintas pantallas, los fragment se encargan de esperar información del servidor para poder arrancar y los dialogs solo aparecen cuándo el usuario los solicita, ya sea a través del panel de usuario o del panel de configuración.







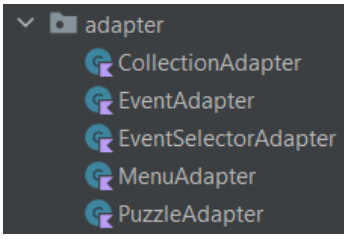
4.2.5 Adaptadores

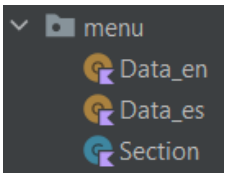
Después de terminar con todas las pantallas me puse a crear Adaptadores, unas clases especiales que permiten iniciar algunos componentes de la aplicación.

Con el MenuAdapter y el paquete menu genere ese spinner tan Customizado(Personalizado) que se aprecia en la pantalla principal de la aplicación.

Con los adaptadores PuzzleAdapter y CollectionAdapter inicialicé los RecyclerView necesarios para mostrar los puzzles en las pantallas de Compra, Colección y Lista de Deseados.

Por último, con EventApdater inicialicé el RecyclerView que muestra los eventos y con EventSelectorAdapter inicialicé el Switch que me permite visualizar los Eventos Principales o los Eventos de la Comunidad.[1]

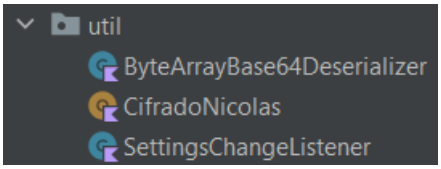




[1]: Actualmente las clases relacionadas con los Eventos no realizan ninguna funciñon debido a que, a fecha de realizar esta memoria, no me dio tiempo a implementar los Eventos. En la presentación de la app puede que si que lleguen a estar implementados.

4.2.6 Utilidades

Por último explicaré las clases que cree para usarlas como Utilidades. Estas clases la única función que tienen es de realizar alguna acción especial en momentos concretos cuándo se utiliza la app. Las utilidades que implementé son las siguientes:



- ByteArrayBase64Deserializer: Utilidad buscada en Internet para que a la hora de intercambiar imágenes con el servidor estas se puedan visualizar, ya que los datos de imagen son enviados en formato Base64 String.
- CifradoNicolas: Utilidad creada totalmente por mí para cifrar y descrifrar las contraseñas de los usuarios con un cifrado aleatorio de César para garantizar seguridad sobre los datos sensibles en mi aplicación.
- SettingsChangeListener: Utilidad investigada en Internet pero implentada por mí la cuál se encarga de inyectar un objeto Listener en el panel de Configuración para aplicar los cambios realizados desde ahí inmediatamente sobre la app.

5. PRUEBAS

5.1 Pruebas en dispositivos

Se han realizado multitud de pruebas en varios dispositivos virtuales y reales con distintas versiones de Android. Los dispositivos usados fueron:

- Pixel 3A (API 33 & API 31) - Virtuales
- Pixel 6 (API 28) - Virtual
- Realme 7 5G (API 31) - Real
- Samsung Galaxy J4+ (API 28) – Real

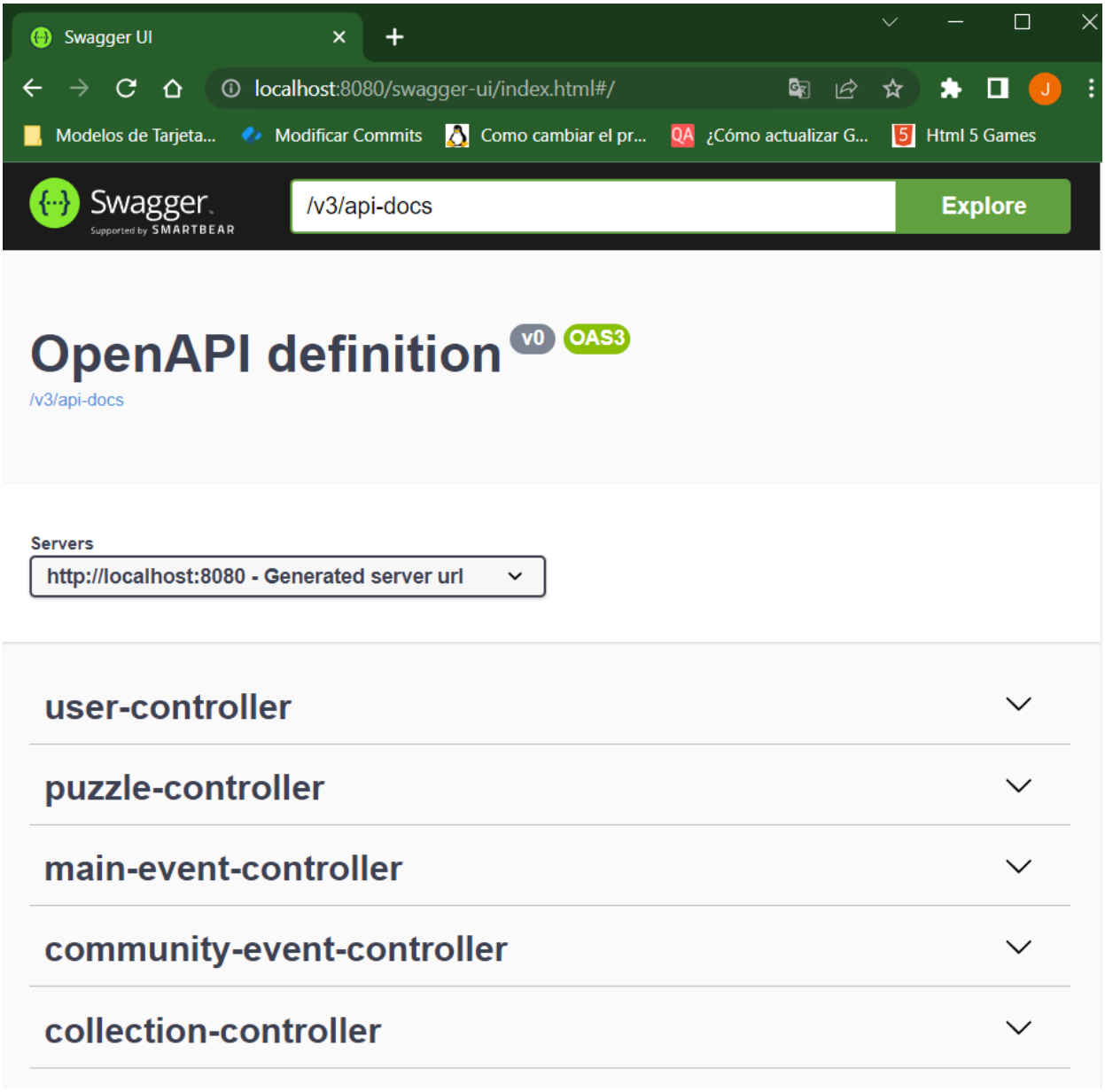
Pruebas con Cuentas de Usuario	RESULTADO
No se puede registrar un usuario sin los datos solicitados	OK
No se puede iniciar sesión sin las credenciales adecuadas	OK
No se puede acceder a la configuración de usuario sin haber iniciado sesión	OK
No se puede acceder a la pestaña de Lista de Deseados sin haber iniciado sesión	OK
Se puede usar la aplicación de forma limitada sin una cuenta	OK
No se muestra contenido alguno en la pestaña de Colección	OK
No se puede acceder al apartado de Colección desde el Panel de Usuario	OK
Se puede acceder al formulario de sugerencia de puzles sin necesidad de tener una cuenta	OK
Se puede manipular la Configuración general sin una cuenta	OK

Pruebas Funcionales de la App	RESULTADO
Las distintas pantallas que dependen del servidor cargan los elementos a mostrar de forma correcta	OK
Al aplicar una configuración desde el Panel de Configuración se aplica inmediatamente sobre la app	OK
El Spinner Customizado funciona perfectamente	OK(98%)[1]
Los botones que invocan diálogos funcionan como deberían	OK
Los diálogos interpretan y delegan correctamente la información introducida por el usuario	OK
El formulario cambia de idioma al cambiar el idioma desde el Panel de Configuración	OK
Los diálogos de información que muestran información detallada sobre Puzles y Eventos la muestran de forma correcta	OK

[1]: El error del 2% indicado en esta prueba se refiere a un ligero error de diseño que a veces provoca que el spinner se extienda demasiado y se perciba una línea divisora de categoría debajo de la última categoría. No supone un error grave ni funcional.

5.2 Pruebas en Swagger UI sobre servidor Spring Boot

Swagger UI es una aplicación web integrada en Spring que permite al usuario probar las distintas Querys establecidas en los Rest Controller del servidor y revisar los datos enviados, en formato JSON, procedentes de la DB.



6. CONCLUSIONES

El resultado de realizar este proyecto ha sido satisfactorio y educativo, ya que me he visto obligado a investigar e implementar muchas tecnologías nuevas para satifacer los requisitos expuestos en mi propuesta inicial.

Paquetes como Retofit han sido un reto ya que no tuve la oportunidad de manejarlo mucho durante el temario en el que se explicó por encima para realizar un proyecto optativo. Realizar un Spinner Customizado y una Pantalla de Configuración de la aplicación sí que fueron conceptos totalmente ajenos a mi persona y me han resultado muy interesantes, ya que implicaron muchas pruebas para poder dejarlos a punto(y aún así con el Spinner siguen habiendo algunos fallos menores…).

La planificación del proyecto . He comprendido que realizar una aplicación siguiendo pautas de calidad impuestas por empresas de renombre de forma solitaria es muy duro, y eso me ha hecho darme cuenta de la importancia de todos los distintos departamentos en una empresa de software que intervienen para poder llevar un proyecto como este a buen puerto.

Como se ha comentado en el punto *2.1 Objetivos Específicos*, los objetivos deseables son mejoras que tuve intención de implementar en la app, pero por falta de tiempo y de conllevar una investigación muy exhaustiva no pude llegar a implementar. Este es el único punto que me desagradó a la hora de realizar el proyecto, pues esperaba poder realizar la aplicación como la había planteado en los bocetos, pero comprendí que no todo sale como uno suele desear.

Deseo cerrar mis conclusiones acerca del proyecto agradeciendo a Patricia Marti Gran por dedicar su tiempo y esfuerzo a enseñarme cómo usar de forma correcta una API tan compleja y completa como es Spring Boot. También dar gracias a Laura Ysabel Sarabia Mora por enseñarme a programar aplicaciones Android con el lenguaje Kotlin a pesar de no ser su departamento.

7. BIBLIOGRAFÍA

[1] - StackOverflow

[2] - YouTube

[3] - Android Developer

[4] - Spring Initializr

[5] - Baeldung

[6] - Maven Repository

[7] - Material Design Icons

[8] - FlatIcon

[9] - Vectr

[10] - Base64 Converter

[11] - LucidChart

[12] - GitHub

8. ANEXOS

8.1 Manual de Usuario

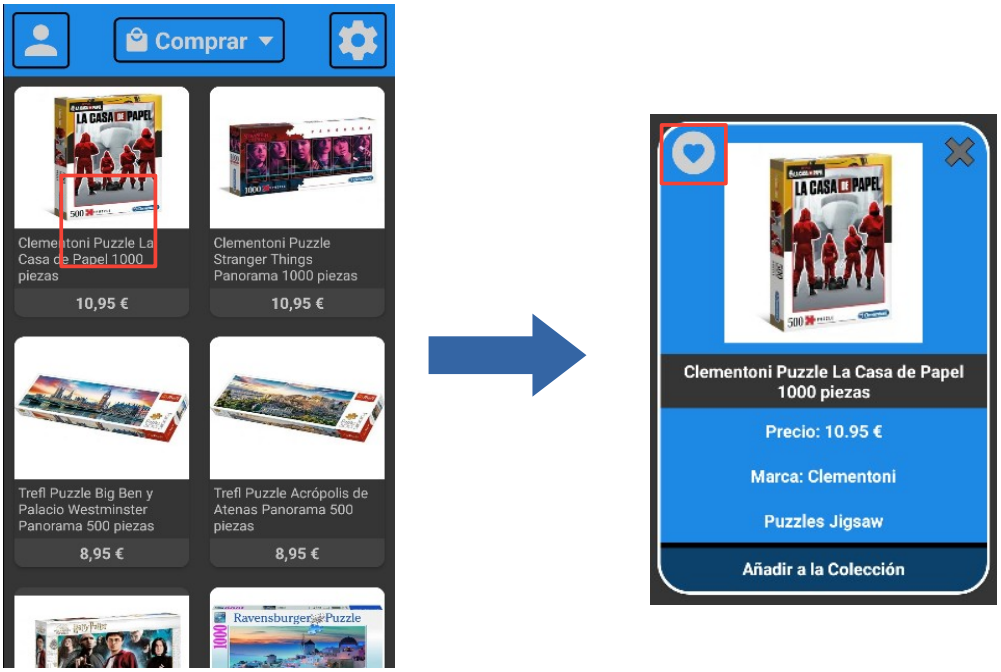
8.1.1 Iniciar Sesión / Registrarse

Para Iniciar Sesión o Registrarse es necesario acceder al Panel de Usuario ubicado en la esquina superior Izquierda y presionar sobre el botón con la acción deseada:

8.1.2 Añadir Puzle en Lista de Deseados / Eliminar Puzle de la Lista de Deseados

Para añadirr un puzle a la lista de deseados basta con abrir la información detallada de un puzle presionando sobre este desde cualquier pantalla, a excepción de la Lista de Deseados y presionar sobre el símbolo ♥:

Para eliminar un puzle de la lista de deseados es realizar la misma acción pero sobre un puzle que ya este añadido y esta acción si puede hacerse desde cualquier pantalla:



8.1.3 Guardar Puzle en Colección / Eliminar Puzle de la Colección

Para guardar un puzle en tu colección basta con abrir la información detallada de un puzle presionando sobre este desde cualquier pantalla, a excepción de la Lista de Deseados y presionar sobre el botón “Añadir puzzle a colección” ubicado en el borde inferior del dialogo:

Para eliminar un puzzle de la colección basta con realizar lo mismo que antes pero sobre la pantalla Colección. Puedes acceder a esta a través de la pantalla principal o desde el panel de usuario: