INSITUTO DE EDUCACIÓN SECUNDARIA SERPIS

BloodStats

Herramienta para jugadores nuevos del World Of Warcraft

Proyecto de Desarrollo de Aplicaciones Multiplataforma

Ciclo Formativo Desarrollo de Aplicaciones Multiplataforma

Departamento de Informática

Autor: Marrahy Arenas, Sergi

Tutor: Zomeño Perona, Jose

Curso: 2023/2024

Modalidad: Presencial

Dedicatoria

Este proyecto se lo agradezco a todos los profesores que me han formado tanto en el ciclo de SMR como en el ciclo de DAM. Se agradece especialmente a los profesores Lionel Tarazón, Sergio Badal, Àngels Chover, Álex Torres y a mi tutor Jose Zomeño, por la paciencia y las enseñanzas en programación, bases de datos, tecnologías actuales y por los consejos dados para la vida laboral en este sector.

También se lo agradezco a mi madre, a mi hermana y mi padre por todo el apoyo que me han dado durante la etapa de desarrollo de este proyecto, y a mis amigos Albert Lozano, Duncan Rúa y Nacho Pérez por compartir todos estos años de formación ya que, sin duda, lo han hecho mucho más ameno.

Por último, agradecer a la todos los jugadores de World of Warcraft que dedican su tiempo a mantener la información faltante en la API de Blizzard completa y actualizada en varias páginas web las cuales podemos visitar para recopilar información.

Resumen: Español

La aplicación móvil BloodStats se ha diseñado con el objetivo principal de dar apoyo a los jugadores nuevos de World of Warcraft que se sienten sobrepasados por la gran cantidad de contenido disponible en el juego. Con la abundancia de las nuevas expansiones y actualizaciones, la curva de aprendizaje puede resultar intimidante, existen varios desafíos desde bien temprano que es dónde uno puede sentirse confundido, como, por ejemplo: se da a elegir entre 14 razas, con sus pasivas y estilos de juego diferentes, 9 clases y dentro de cada clase de 2 a 4 especializaciones, después de haber creado el personaje, se necesita saber qué tipo de equipo es útil para esa clase de personaje ya que tiene varios atributos, que según el estilo de juego seleccionado, puede llegar a ser lioso. Es aquí donde BloodStats entra en juego como una herramienta que brinda una gran ayuda proporcionando orientación y claridad. Además de su enfoque en ayudar a los jugadores a comprender las clases y especializaciones del juego, BloodStats ofrece una variedad de sistemas integrados diseñados para mejorar la experiencia del usuario. El resultado que se quiere obtener es reducir el índice de abandono entre los jugadores nuevos que sienten que no van a poder abordar todo el contenido que se les ofrece. En conclusión, BloodStats no solo es una herramienta de referencia, sino una aplicación útil para los jugadores que buscan crecer y mejorar su habilidad. Esta aplicación Android está programada en Kotlin y se ha probado en varios emuladores para su correcto funcionamiento, además se consume la API de Blizzard para ofrecer todos datos necesarios.

Palabras clave:

- World of Warcraft, Android, Kotlin, Emuladores, Programación, API, Blizzard.

Resum: Valencià

L'aplicació mòbil BloodStats s'ha dissenyat amb l'objectiu principal de donar suport als jugadors nous de World of Warcraft que se senten aclaparats per la gran quantitat de contingut disponible en el joc. Amb l'abundància de noves expansions i actualitzacions, la corba d'aprenentatge per als nous jugadors pot resultar intimidant, existeixen diversos reptes des de ben aviat on un pot sentir-se confós, com, per exemple: es dona a triar entre 14 races, amb les seues passives i estils de joc diferentes, 9 classes i dins de cada classe de 2 a 4 especialitzacions, després d'haver creat el personatge, ja que té diversos atributs, que segons l'estil de joc seleccionat, pot arribar a ser embolicat. És ací on BloodStats entra en joc com una eina que brinda una gran ajuda proporcionant orientació i claredat. A més del seu enfocament en ajudar els jugadors a comprendre les classes i especialitzacions del joc, BloodStats ofereix una varietat de sistemes integrats dissenyats per millorar l'experiència de l'usuari. El resultat que es vol obtenir és reduir l'índex d'abandonament entre els jugadors nous que senten que no podran abordar tot el contingut que se'ls ofereix. En conclusió, BloodStats no solament és una eina de referència, sinó una aplicació útil per als jugadors que busquen créixer i millorar la seua habilitat. Aquesta aplicació Android està programada en Kotlin i s'ha provat en diversos emuladors per al seu correcte funcionament, a més consumeix la API de Blizzard per a oferir totes les dades necessàries.

Paraules clau:

- World of Warcraft, Android, Kotlin, Emuladors, Programació, API, Blizzard.

Abstract: English

The mobile application BloodStats has been designed with the primary goal of supporting new players of World of Warcraft who feel overwhelmed by the vast amount of content available in the game. With the abundance of new expansions and updates, the learning curve can be intimidating, and there are several early challenges where one can feel confused, such as, for example: there is a choice of 14 races, each with their own passives and different play styles, 9 classes and within each class, 2 to 4 specializations; after creating the character, it is necessary to know what type of equipment is useful for that character class as it has various attributes, which depending on the selected play style, can become confusing. This is where BloodStats comes into play as a tool that provides great assistance by offering guidance and clarity. In addition to its focus on helping players understand the classes and specializations of the game, BloodStats offers a variety of integrated systems designed to enhance the user experience. The desired outcome is to reduce the dropout rate among new players who feel they will not be able to tackle all the content offered to them. In conclusion, BloodStats is not only a reference tool but also a useful application for players looking to grow and improve their skills. This Android application is programmed in Kotlin and has been tested on various emulators for its proper functionality. Additionally, it consumes the Blizzard API to provide all necessary data.

Keywords:

- World of Warcraft, Android, Kotlin, Emulators, Programming, API, Blizzard.

Tabla de contenido

Tabla de ilustraciones

1. Justificación

Este proyecto debe ser tratado como una herramienta de guía para jugadores novatos que quieren mejorar su personaje y habilidad como jugador. El propósito es disminuir la tasa de abandono y se espera lograr que jugadores nuevos puedan disfrutar mucho más del juego.

Month	Average Monthly Players	Monthly Gain / Loss	Monthly Gain / Loss %	Average Players Daily
Last 30 Days	7,769,828	-417,556	-5.10%	2,097,853.61
April 2024	8,187,385	-261,928	-3.10%	2,210,593.90
March 2024	8,449,314	-849,406	-9.13%	2,281,314.65
February 2024	9,298,720	-111,881	-1.19%	2,510,654.44
January 2024	9,410,602	460,057	5.14%	2,164,438.45

Figura 1 - Tabla sobre los jugadores activos mensuales en la actualidad.

Según la página activeplayer.io, la cantidad de gente suscrita al juego va disminuyendo por varias razones, pero un gran porcentaje es por la poca ayuda que ofrece el juego, lo que hace que los jugadores se sientan perdidos y frustrados por no saber cómo mejorar. Existen muchas dificultades en las que los nuevos jugadores se enfrentan, comenzando desde la pantalla de creación de un personaje hasta que consiguen acceder a un contenido más complejo del juego. Actualmente no existen aplicaciones para dispositivos móviles Android en el mercado dedicadas a guiar a estos jugadores.

Problema:

1. **Falta de información:** Los jugadores que quieran mejorar en contenido PVE avanzado suelen tener dificultades para encontrar guías detallas y actualizadas sobre el equipo, atributos y talentos utilizados por los mejores jugadores.

2. **Exceso de contenido:** El gran volumen de información disponible en el WoW puede abrumar a los jugadores nuevos, especialmente a aquellos que quieren aventurarse a contenido más desafiante y avanzad.

2. Gestión del proyecto

En este apartado se presenta un diagrama de Gantt que define como se va a organizar el proyecto en base al tiempo dado para su desarrollo junto a la memoria.

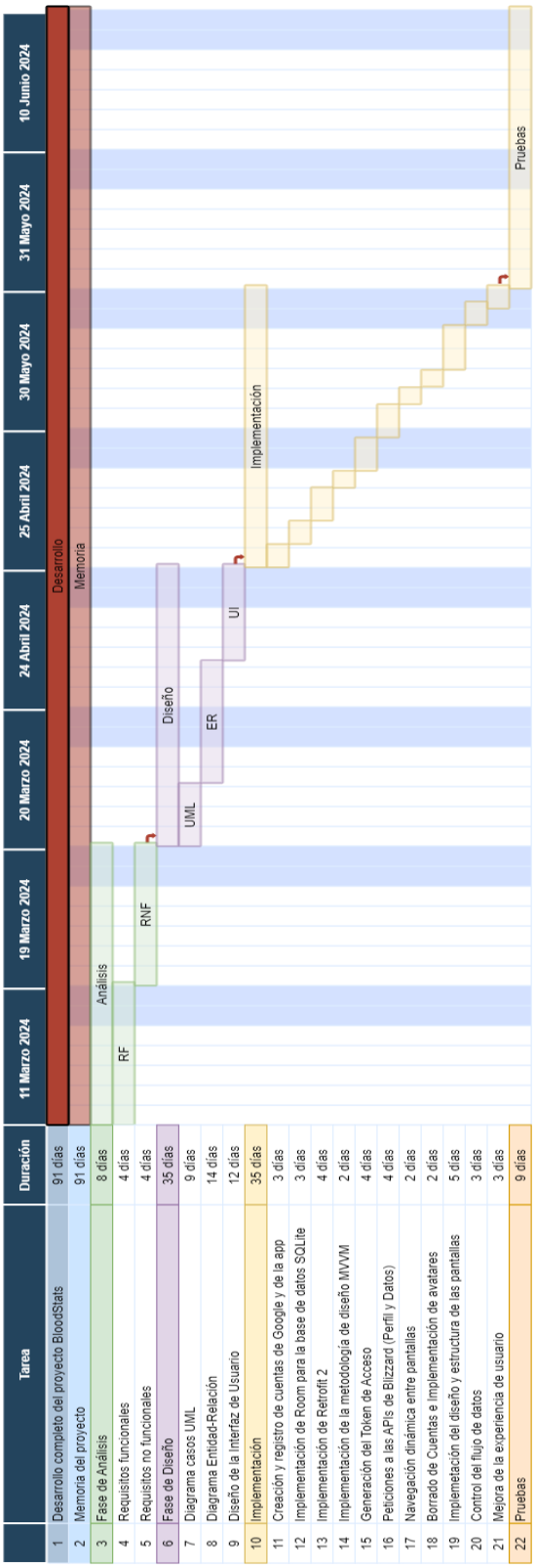


Figura 2 - Diagrama de Gantt

3. Tecnologías, Lenguajes y Herramientas utilizadas

A continuación, se van a nombrar las herramientas, tecnologías y lenguaje utilizado en el desarrollo de la aplicación:

3.1. Entorno de Desarrollo – Herramienta

- **Android Studio**

Android Studio es un entorno de desarrollo completo para la creación de aplicaciones móviles Android. Ofrece un conjunto de herramientas intuitivas y avanzadas como un editor inteligente, un sistema de construcción flexible y un emulador de dispositivos integrado, Android Studio simplifica el proceso de desarrollo y optimización de aplicaciones para Android. Además, su integración con servicios como Firebase y Google Play facilita la implementación de tecnologías utilizadas en este proyecto.

Figura 3 - Logo de Android Studio

3.2. Framework – Herramienta

- **Jetpack Compose**

Jetpack Compose es un kit de herramientas desarrollado por Google, diseñado para crear interfaces de usuario de una manera simple y rápida, acelerando así el desarrollo de la interfaz en aplicaciones Android. Este framework utiliza Kotlin como lenguaje oficial aún que también se puede desarrollar aplicaciones con Java y se enfoca en la declaración de Composables para la creación de UIs. Jetpack Compose mantiene una compatibilidad con Material Design para generar paletas de colores, fuentes de letra, etc. más atractivas para el usuario. La UI se actualiza automáticamente como respuesta a los cambios que el usuario pueda accionar, de esta manera se elimina la necesidad de controlar manualmente el estado de la UI.

Figura 4 - Logo de Jetpack Compose

3.3. Lenguaje de programación – Lenguaje

- **Kotlin**

Kotlin es un lenguaje de programación que se utiliza para el desarrollo de esta aplicación ya que es una elección sólida por la interoperabilidad con Java, su concisión y claridad de código, la seguridad contra nulos, librerías como Retrofit 2 y Room, compatibilidad con las Coroutines y la gestión de hilos.

Figura 5 - Logo de Kotlin.

3.4. Control de versiones – Herramienta

- **Github**

GitHub ofrece una solución para el control de versiones. En este proyecto ha proporcionado una plataforma segura para almacenar el proyecto y la memoria, lo que ha permitido acceder a ellos desde cualquier lugar, compartirlos con el tutor y otros colaboradores de manera sencilla y segura.

Figura 6 - Logo de GitHub

3.5. Base de datos – Herramienta

- **SQLite**

SQLite es una biblioteca de gestión de bases de datos relacionales que se ha utilizado en el desarrollo de esta aplicación móvil. Es una solución ligera, eficiente y autónoma que permite crear y gestionar bases de datos locales en el dispositivo del usuario. Con soporte completo para transacciones ACID;

> **Atomicidad**: Posibilidad de revertir los cambios o algún fallo en la base de datos con un rollback.
>
> **Consistencia**: Asegura que todas las restricciones de integridad, como las claves primarias y las claves foráneas, se mantengan en todo momento.
>
> **Aislamiento**: Las transacciones se ejecutan de forma aislada y no se van a ver afectadas por las transacciones concurrentes que se están ejecutando simultáneamente en la base de datos.

Durabilidad: Asegura que una vez que una transacción se haya confirmado (committed), los cambios realizados por esa transacción permanezcan en la base de datos incluso en caso de falla del sistema, como un corte de energía.

Figura 7 - Logo de SQLite

3.6. Editor de textos - Herramienta

- **Microsoft Office Word**

Se utiliza Microsoft Word para redactar la memoria del Proyecto de Fin de Curso (PFC) por la cantidad de herramientas y funcionalidades específicas para realizar la memoria de este proyecto. Word tiene una interfaz intuitiva que facilita la creación y edición de contenido, así como la aplicación de formatos y estilos de texto que cumplen con las normas de presentación de los informes proporcionados. Además, Word proporciona herramientas para la revisión de texto.

Figura 8 - Logo de Microsoft Office Word

3.7. Herramienta de diagramación - Herramienta

- **Draw.io**

Herramienta utilizada para crear diagramas UML (Unified Modeling Language) de casos de uso y diagramas de Entidad-Relación (ER) para la representación visual de la estructura y relaciones de datos del sistema.

Figura 9 - Logo de draw.io

3.8. Creación del diseño IU – Herramienta

- **Excalidraw**

El propósito de Excalidraw es el mismo que el de Draw.io, pero en este caso se ha utilizado para desarrollar la interfaz de usuario (UI) que ha permitido crear los bocetos de la aplicación y los flujos de trabajo.

Figura 10 - Logo de Excalidraw

3.9. Creación de la paleta de colores – Herramienta

- **Material Theme Builder**

Se ha utilizado esta herramienta para generar una paleta de colores adecuada para esta aplicación.

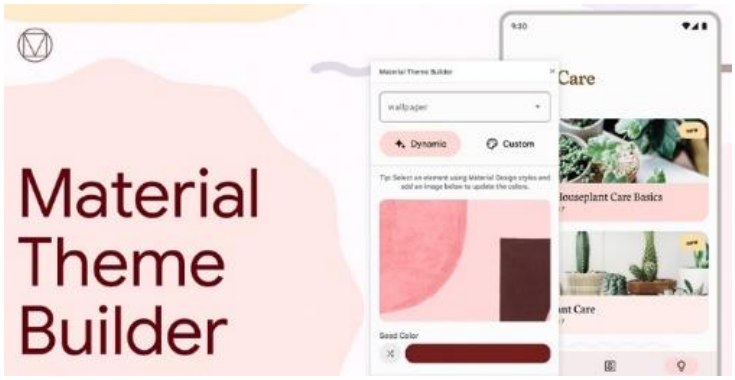


Figura 11 - Logo Material Theme Builder

3.10. Generación de solicitudes HTTP – Tecnología

- **Retrofit**

La librería Retrofit ha sido utilizada para facilitar las solicitudes a la API de Blizzard en la aplicación. Simplifica el proceso de comunicación al proporcionar una forma intuitiva de definir y realizar peticiones HTTP. Utilizando anotaciones Java, Retrofit permite estructurar las solicitudes de una manera muy sencilla. Esto simplificará significativamente el consumo de la API de Blizzard, permitiendo así un desarrollo más conciso en la lógica de la aplicación.

Figura 12 - Logo de Retrofit

3.11. Servicio de inicio de sesión – Tecnología

- **Firebase authentication**

La función Sign-In One Tap de Google, integrada en el servicio de Firebase Authentication, simplifica el inicio de sesión al permitir a los usuarios acceder con un solo toque, evitando la introducción manual de credenciales. Esta característica utiliza la información de la cuenta de Google para una autenticación segura y conveniente. Su implementación y su capacidad para reducir la fricción en el proceso de inicio de sesión mejora significativamente la UX y la tasa de conversión de una aplicación.

La tasa de conversión se refiere a la cantidad de usuarios que realizan una acción en la aplicación en comparación con el número total de usuarios que interactúan con ella.

Figura 13 - Logo de Firebase

4. Descripción del proyecto

4.1. Análisis

En este apartado describiremos los *requisitos funcionales* y los *requisitos no funcionales*, parte fundamental del proyecto ya que tienen un impacto significativo en la calidad de la aplicación.

Con este paso se permite establecer las bases de desarrollo del proyecto y una visión de la aplicación final.

- **Introducción:**

Este análisis se centra en una aplicación la cual está dirigida a jugadores nuevos de World of Warcraft (WoW) que buscan realizar contenido PVE avanzado, como raids o Míticas+. BloodStats pretende solucionar la falta de información de la cual el juego carece sobre los jugadores de élite que lideran este tipo de contenido. Esta aplicación tiene como público objetivo jugadores que nunca han jugado al WoW.

  - PVE: Contenido basado en el jugador contra la máquina.
  - Raids: Grupos de 15 o 25 jugadores preparados para realizar el contenido más avanzado del juego. Este modo de juego se basa en batallar contra los jefes más fuertes y difíciles de hacer. Generalmente se requiere aprender varias estrategias.
  - Míticas+: Grupos de 5 jugadores que hacen una versión más desafiante de las mazmorras normales. Para poder hacerlas más difíciles agregan sufijos; modificadores que se aplican a niveles más altos (+10 … +20)
  - Jugador de élite: Jugador muy experimentado que pertenece a una hermandad que compite por ser los mejores del mundo.

4.1.1. Tabla Requisitos Funcionales

Número	Requerimiento	Descripción	Prioridad
RF1	Iniciar sesión	El usuario deberá introducir las credenciales existentes en los campos correspondientes para que el sistema compruebe esos datos verificarlos y navegar hacia la pantalla de búsqueda.	Alta
RF2	Registrar una cuenta	Para hacer uso de la aplicación el usuario debe crearse una cuenta introduciendo los datos que la aplicación necesita para crear la cuenta. Al crear la cuenta	Alta

		exitosamente el sistema navega hacia la pantalla de inicio de sesión.	
RF3	Iniciar sesión con Google	Este tipo de inicio de sesión requiere de una cuenta existente y registrada en el dispositivo móvil para el correcto funcionamiento de la función. Si se da el caso, creará una cuenta en la base de datos si no existe ninguna relacionada a la seleccionada de Google.	Media
RF4	Buscar personaje	Esta función le permite al usuario buscar un personaje de cualquier reino situado en los servidores de Europa Central.	Alta
RF5	Buscar piezas de equipo	Esta función le permite al usuario buscar piezas de equipo o artículos para recibir la descripción e información de este en base el personaje buscado.	Media
RF7	Navegar por las diferentes pantallas de la aplicación	Se ha implementado una interfaz de usuario amigable e intuitiva para navegar por las diferentes pantallas que ofrece la aplicación.	Alta
RF8	Agregar personajes a la lista de favoritos	El usuario puede agregar varios personajes a una lista de favoritos para un acceso cómodo y rápido.	Media

4.1.2. Requisitos no funcionales

Número	Requerimiento	Descripción	Prioridad
RNF1	Exigencias software	Se requiere un sistema operativo Android con una versión 7.0 (Nougat) o superior para hacer uso de la aplicación. La aplicación está disponible en varios idiomas.	Alta
RNF2	Exigencias hardware	Como requisitos mínimos el dispositivo móvil debe tener al menos 4Gb de RAM y un procesador de doble núcleo, además se requiere como mínimo 500Mb de almacenamiento para descargar la aplicación.	Alta
RNF3	Animaciones	La aplicación está dotada de animaciones para la mejora visual de la aplicación.	Baja

RNF4	Validación de datos	Mensajes claros y comprensibles en caso de datos inválidos.	Alta
RNF5	Seguridad	El sistema asegura los datos del usuario tanto con la librería UUID de Kotlin, la encriptación de datos en la base de datos SQLite implementada con Room y con la gestión automática de las credenciales de Google con Firebase.	Alta
RNF6	Compatibilidad	Asegura la compatibilidad con muchas versiones de Android y garantiza el correcto funcionamiento en diferentes pantallas y resoluciones.	Media
RNF7	Personalización	Se le permite al usuario cambiar su nombre y contraseña, además podrá cambiar el tema entre claro y oscuro.	Baja

Tablas de Casos de Uso:

Nombre	Iniciar Sesión
Descripción: Permite acceder a las funciones de la aplicación	
Actores: Usuario	
Precondiciones: El usuario debe estar previamente registrado	
Flujo Normal: 1.- El autor inserta sus credenciales en los campos correspondientes. 2.- El actor pulsa el botón de iniciar sesión. 3.- El sistema comprueba si las credenciales existen en la base de datos. 4.- El sistema navega hacia la pantalla de búsqueda de personaje.	
Flujo Alternativo: 3.A.- El sistema comprueba si las credenciales son correctas, si no lo son, se avisa al actor con un mensaje de error.	
Postcondiciones El usuario puede hacer uso de las funciones de la aplicación.	

Nombre	Iniciar Sesión con Google
Descripción: Permite acceder a las funciones de la aplicación	
Actores: Usuario	
Precondiciones: El usuario debe tener registrada una cuenta de Google en el dispositivo	
Flujo Normal: 1.- El autor pulsa el botón de Iniciar Sesión con Google. 2.- El sistema comprueba si hay una cuenta existente de Google registrada en el dispositivo. 3.- El sistema navega hacia la pantalla de búsqueda de personaje.	
Flujo Alternativo: 2.A.- El sistema comprueba si hay una cuenta existente de Google registrada en el dispositivo, si no la hay, se le avisa al actor con un mensaje de error.	
Postcondiciones El usuario puede hacer uso de las funciones de la aplicación.	

Nombre	Crear una cuenta nueva
Descripción: Permite al usuario crear una cuenta	
Actores: Usuario	
Precondiciones: El usuario debe introducir unas credenciales que no se estén ya registradas en la base de datos.	
Flujo Normal: 1.- El autor introduce credenciales no existentes en la base de datos. 2.- El autor pulsa el botón de registrarse. 3.- El sistema comprueba que el nombre de usuario no existe en la base de datos. 4.- El sistema navega hacia la pantalla de búsqueda de personaje.	
Flujo Alternativo: 3.- El sistema comprueba que el nombre de usuario existe en la base de datos, avisa al actor con un mensaje de error.	
Postcondiciones El usuario puede hacer uso de las funciones de la aplicación.	

Nombre	Buscar un personaje
Descripción: Permite a un usuario registrado la búsqueda de personajes.	
Actores: Usuario	
Precondiciones: El usuario debe introducir el nombre exacto de un personaje existente en el reino/servidor donde esté alojado.	
Flujo Normal: 1.- El sistema carga todos los datos de este personaje y los monta de una forma estructurada y amigable.	
Flujo Alternativo: 1.A.- Si el nombre o el reino/servidor donde está alojado el personaje son incorrectos, el sistema volverá a la pantalla de búsqueda de personaje mostrando un mensaje de error.	
Postcondiciones El usuario puede consumir toda la información del personaje buscado.	

Nombre	Pantalla de Atributos
Descripción: Muestra todos los atributos del personaje buscado y permite añadir el personaje a la lista de favoritos.	
Actores: Usuario	
Precondiciones: El usuario puede acceder a esta pantalla pulsando el botón de Atributos, también puede agregar el personaje con el botón en forma de estrella.	
Flujo Normal: 1.- El sistema muestra todos los atributos principales del personaje. 2.- El sistema guarda en la base de datos la relación entre el ID del personaje favorito y el ID del usuario. Además, se guarda el nombre, reino/servidor y clasificación en las míticas para mostrarlo en el perfil y poder buscarlo con un clic. Con el mismo botón de favoritos se puede eliminar el personaje de la lista.	
Flujo Alternativo: Ninguno.	
Postcondiciones El usuario puede hacer uso de esta información para compararlo con su personaje.	

Nombre	Pantalla de Equipamiento
Descripción: Muestra todas las piezas de equipo que el personaje buscado tiene equipadas.	
Actores: Usuario	
Precondiciones: El usuario debe haber introducido bien el nombre del personaje y el reino donde se encuentra alojado.	
Flujo Normal: 1.- El sistema estructura toda la información que se quiere mostrar en la pantalla junto a una lista de piezas de equipo equipadas con la función *clickable* para que el usuario pueda ver una descripción más detallada del equipo.	
Flujo Alternativo: 1.A.- El nombre del personaje o el servidor donde se aloja son incorrectos.	
Postcondiciones El usuario puede hacer uso de esta información para compararlo con su personaje.	

Nombre	Pantalla de Especialización
Descripción: Muestra todos los talentos de clase y de especialización del personaje buscado.	
Actores: Usuario	
Precondiciones: El usuario debe haber introducido bien el nombre del personaje y el reino donde se encuentra alojado.	
Flujo Normal: 1.- El sistema muestra dos botones, Talentos de Clase y Talentos de Especialización. Cada uno muestra talentos diferentes para que el usuario pueda copiarse esta rama de talentos o leer lo que hacen.	
Flujo Alternativo: 1.A.- El personaje buscado no contiene ninguna especialización activa y no tiene asignado ningún punto de habilidad en los talentos de clase.	
Postcondiciones El usuario puede hacer uso de esta información para compararlo con su personaje.	

Nombre	Pantalla de Mazmorras
Descripción: Muestra todas las mazmorras activas de la nueva expansión junto al tiempo, puntuaje y clasificación del personaje.	
Actores: Usuario	
Precondiciones: El usuario debe haber introducido bien el nombre del personaje y el reino donde se encuentra alojado.	
Flujo Normal: 1.- El sistema muestra las imágenes de las mazmorras, el tiempo invertido en hacerla y el puntuaje obtenido por el personaje al pasársela. También muestra la clasificación del personaje que se basa en el total de la suma de todos los puntuajes.	
Flujo Alternativo: 1.A.- El personaje no ha realizado ninguna mazmorra, en consecuencia, solo se muestran las fotos pero sin datos.	
Postcondiciones El usuario puede hacer uso de esta información para compararlo con su personaje.	

Nombre	Pantalla de Hermandad
Descripción: Muestra el nombre, el escudo de la facción y una lista de miembros.	
Actores: Usuario	
Precondiciones: El usuario debe haber introducido bien el nombre del personaje y el reino donde se encuentra alojado.	
Flujo Normal: 1.- El sistema muestra la información del clan al que pertenece el personaje buscado. Esta pantalla contiene los nombres de todos los miembros pertenecientes a la hermandad con una función *clickable* para una búsqueda mucho más rápida del personaje.	
Flujo Alternativo: 1.A.- El personaje no pertenece a ninguna hermandad, en ese caso se muestra un icono avisando del error.	
Postcondiciones El usuario puede buscar personajes desde la pantalla de la hermandad de una manera mucho más rápida.	

Nombre	Pantalla de Perfil
Descripción: Muestra el avatar escogido en la pantalla de inicio de sesión o la foto de perfil que contenga la cuenta de Google.	
Actores: Usuario	
Precondiciones: El usuario debe registrarse o haber iniciado sesión.	
Flujo Normal: 1.- El usuario puede cerrar sesión pulsando el botón cerrar sesión. 2.- El usuario puede eliminar su cuenta pulsando el botón eliminar cuenta. 3.- El sistema muestra un mensaje de advertencia preguntando si de verdad quiere eliminar su cuenta permanentemente. 4.- Si cancela el proceso, vuelve a la pantalla de perfil. 5.- Si acepta el proceso, se elimina la cuenta permanentemente sin posibilidad de recuperación y volviendo a la pantalla de inicio de sesión. 6.- El sistema muestra una lista de todos los personajes favoritos que el usuario haya querido guardar.	
Flujo Alternativo: 6.A.- Si el usuario no ha agregado ningún personaje favorito aún, se muestra un mensaje advirtiendo de que no se han añadido personajes a la lista de favoritos.	
Postcondiciones El usuario puede eliminar la cuenta o buscar personajes de una manera mucha más rápida.	

Nombre	Pantalla de Opciones
Descripción: Muestra todas las opciones que el usuario puede ajustar a su gusto.	
Actores: Usuario	
Precondiciones: El usuario debe registrarse o haber iniciado sesión.	
Flujo Normal: 1.- Cambio de tema de claro a oscuro y viceversa. El sistema cambia los colores e la aplicación.	
Flujo Alternativo: Ninguno.	
Postcondiciones El usuario puede personalizar la aplicación al gusto.	

4.2. Diseño

Solución propuesta:

BloodStats ofrece toda la información necesaria sobre los mejores jugadores del WoW y varios sistemas útiles:

a) **Talentos de clase y especialización:** La aplicación se centra en proporcionar la información necesaria sobre los talentos de las clases y especializaciones, permitiendo a los jugadores encontrar estrategias y técnicas personalizadas.
b) **Mejora del personaje:** La aplicación ayuda a los jugadores a identificar mejoras del equipo que pueden adaptar en sus personajes para aumentar el rendimiento del personaje fijándose en otros jugadores.

Ventajas:

a) **Información precisa y actualizada:** Al utilizar la API de Blizzard oficial la información es 100% verídica y se mantiene siempre actualizada.
b) **Aprendizaje eficiente:** La información se presenta de forma muy amigable y fácil de entender para que el usuario pueda entender la estrategia que utiliza ese personaje.
c) **Motivación y guía:** La aplicación puede motivar a los jugadores a alcanzar un alto nivel de rendimiento en el modo de juego PVE con todo el contenido que conlleva, ya que, al tener toda la información de un jugador experimentado en la aplicación, simplemente podría copiar el equipo y los talentos para tener una base consistente y llegar a niveles más altos.

Funcionalidades:

a) **Búsqueda de personajes:** Para poder encontrar un jugador la aplicación tiene como requisito 2 parámetros: Nombre y Servidor. El servidor es el nombre del reino donde ese personaje está alojado.
b) **Diseño intuitivo:** La aplicación ofrece una interfaz intuitiva y fácil de usar para que los jugadores puedan encontrar información con mucha rapidez.
c) **Guardado de personajes:** La aplicación permite al usuario marcar como favorito a todos los personajes que desee, esta lista de personajes se encontrará en la pantalla del perfil del usuario que haya iniciado sesión para la búsqueda del personaje.
d) **Muestra de toda la información necesaria:** Se muestra toda la información de un personaje en varias pantallas con un sistema de navegación dinámico.

Conclusión:

BloodStats tiene el potencial de ser una herramienta valiosa para jugadores del WoW que buscan mejorar su personaje o como jugador en contenido PVE avanzado.

4.2.1. Diagrama Entidad Relación

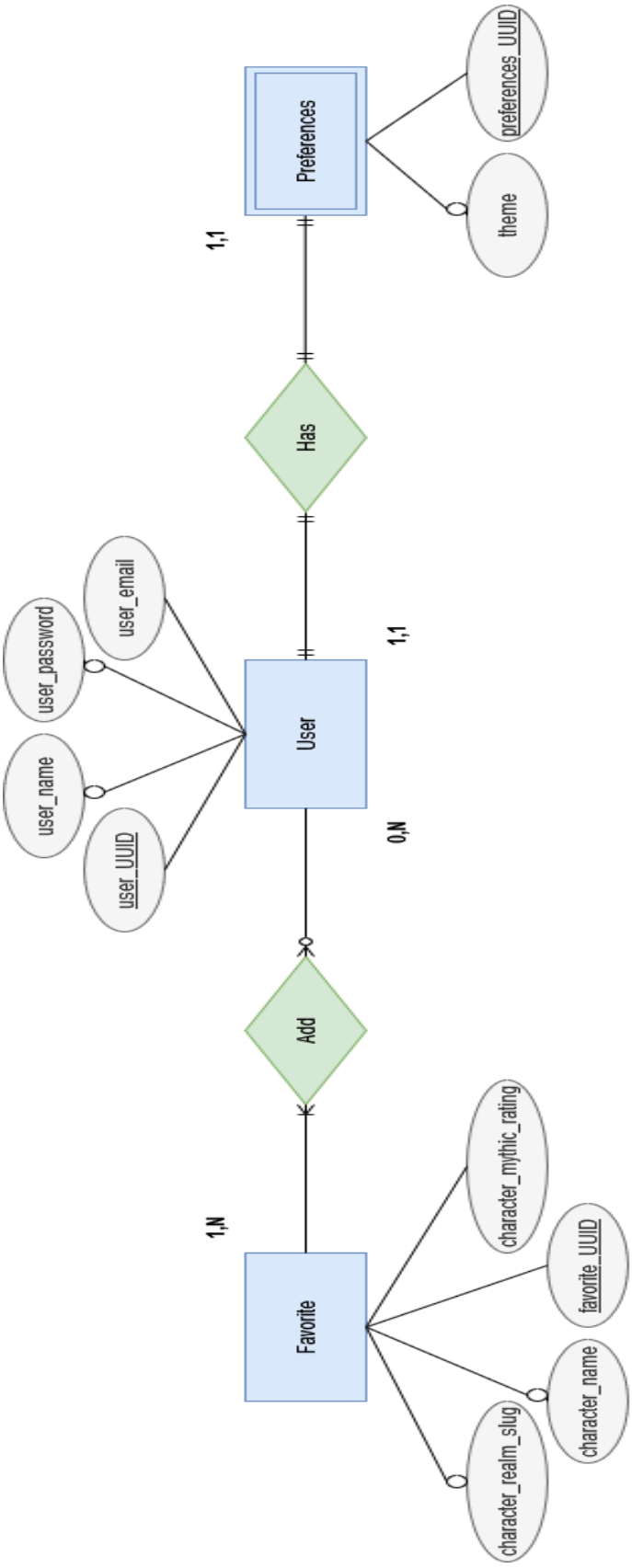


Figura 14 - Diagrama Entidad - Relación

4.2.2. Diagrama UML Casos de Uso

En este apartado se presenta el diagrama de Casos de Uso para tener una visión general de las funciones que ofrece este proyecto.

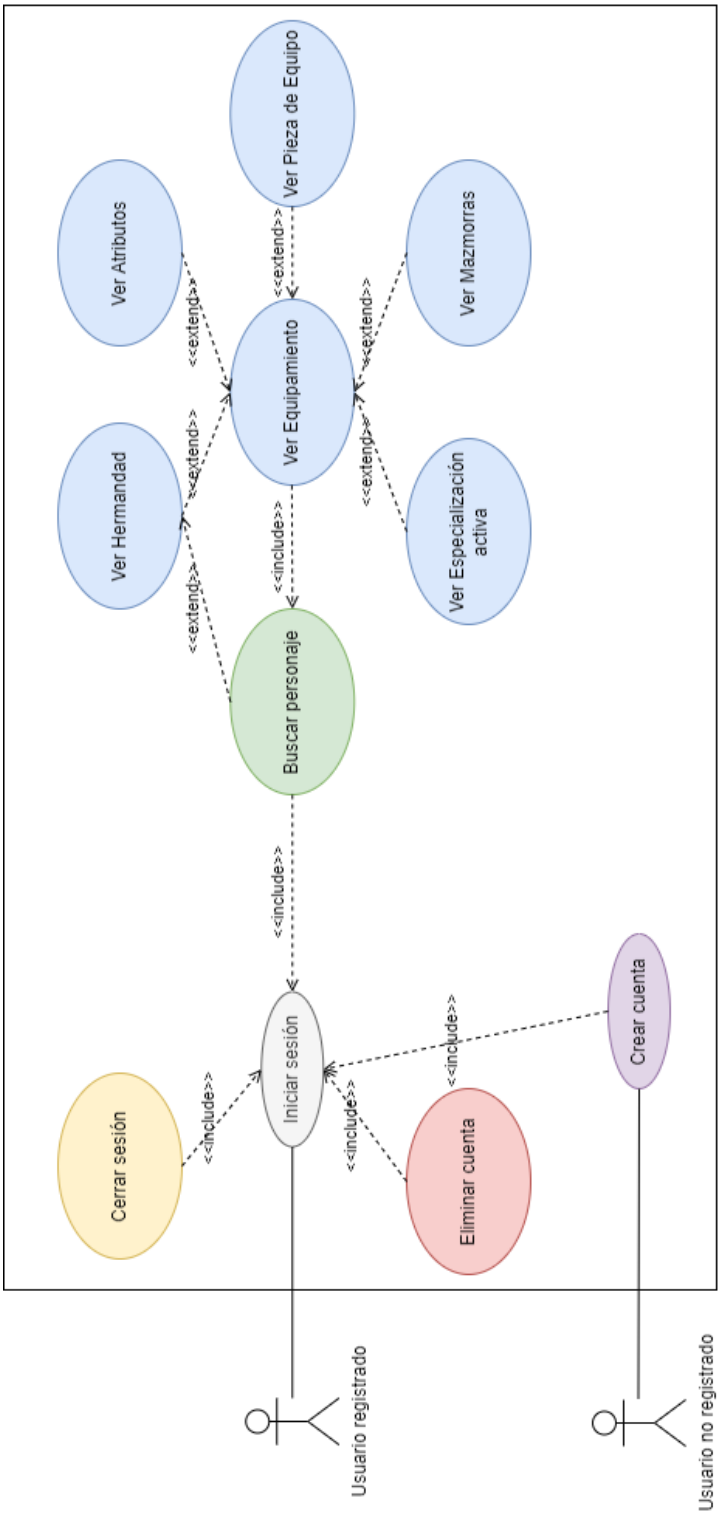


Figura 15 - Diagrama UML de los Casos de Uso

4.2.3. Bocetos de la interfaz de usuario

- **Pantalla Splash:** Pantalla que introduce la aplicación con el Logo de BloodStats

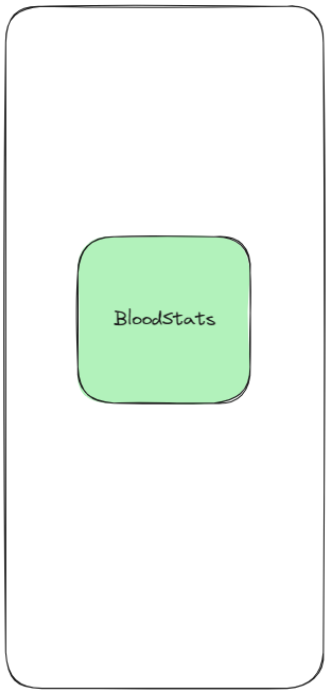


Figura 16 - Pantalla splash donde se muestra el logo de la aplicación

- **Pantalla Inicio de Sesión/Registro:** En esta pantalla el usuario podrá iniciar sesión de dos maneras, por Google o con una cuenta existente que el sistema almacena cuando se registra.

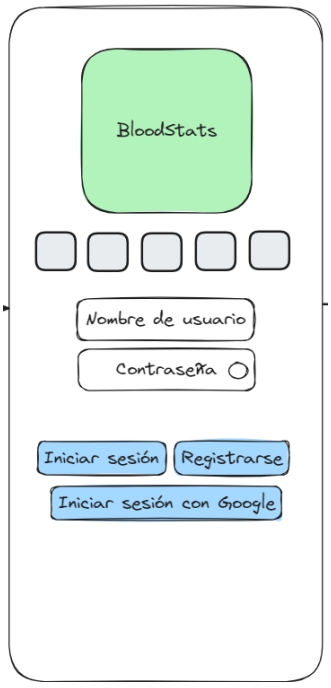


Figura 17 - Pantalla de registro o inicio de sesión

- **Pantalla Búsqueda de Personaje:** En esta pantalla el usuario podrá buscar el personaje, sin embargo, es necesario saber a qué reino/servidor pertenece este personaje para una respuesta exitosa.

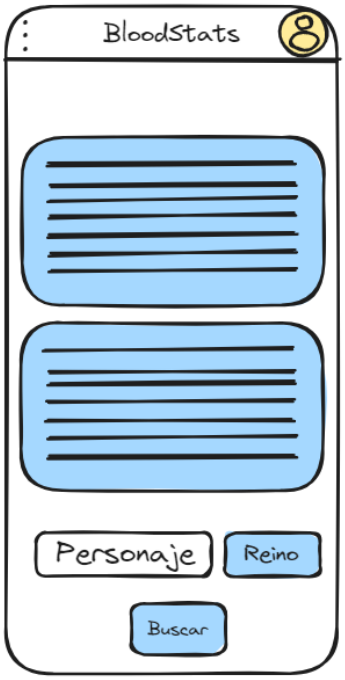


Figura 18 - Pantalla para buscar personajes

- **Pantalla de Equipamiento:** En esta pantalla se muestra el nombre, especialización, nivel de equipo y equipamiento. El usuario podrá hacer clic en el nombre de las piezas de equipo para obtener una descripción más detallada.

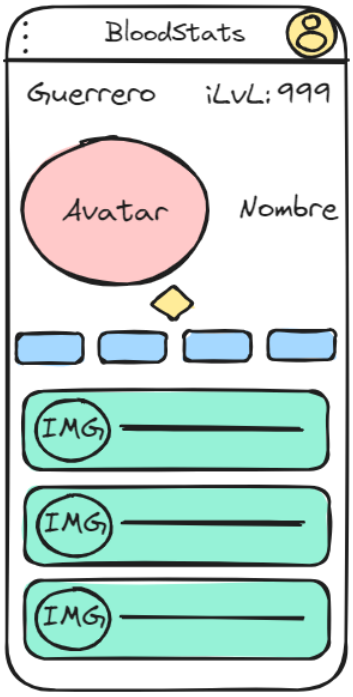


Figura 19 - Pantalla donde se muestra el equipamiento de un personaje

- **Pantalla de Atributos:** En esta pantalla se muestran los atributos principales y necesarios que el usuario necesita saber. El personaje se puede añadir a la lista de favoritos mediante un botón para tener un acceso rápido desde la pantalla de perfil.

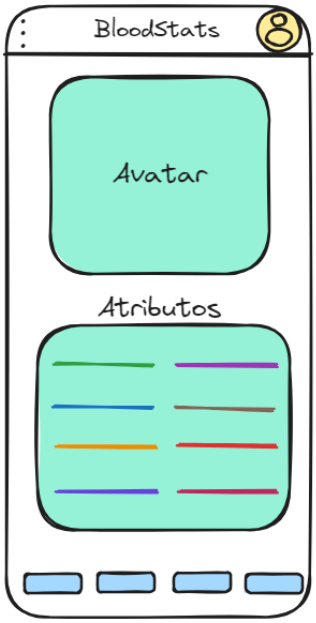


Figura 20 - Pantalla donde se muestran los atributos principales del personaje buscado

- **Pantalla de Hermandad:** En esta pantalla el usuario podrá ver el clan al que pertenece el personaje y todos los miembros que forman el clan. Si el usuario hace clic en uno de los nombres, el sistema navegará a la pantalla de personaje mostrando toda la información de este.

Figura 21 - Pantalla donde se muestra la facción y los miembros del clan del personaje buscado

- **Pantalla de Talentos:** En esta pantalla el usuario podrá ver la especialización y habilidades activas del personaje, las habilidades están descritas con toda la información necesaria para que el usuario pueda informarse de su utilidad y funcionamiento.

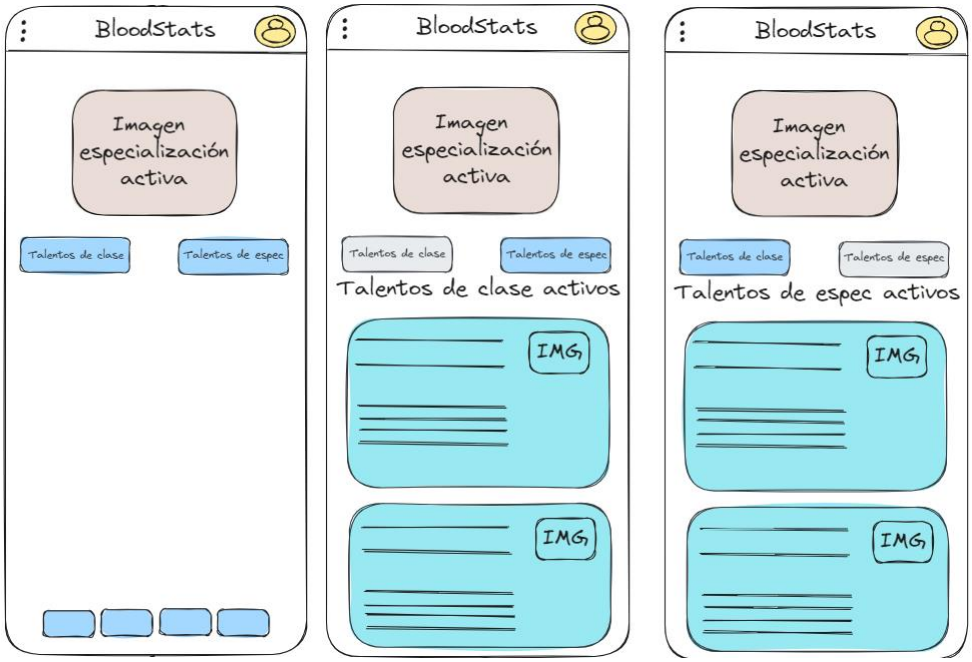


Figura 22 - Pantalla donde se muestra la especialización activa y los talentos de clase y especialización con sus descripciones

- **Pantalla de Mazmorras:** En esta pantalla se muestran las mazmorras hechas por el personaje buscado y su clasificación, este puntuaje está calculado en base al tiempo que haya tardado en hacer cada mazmorra.

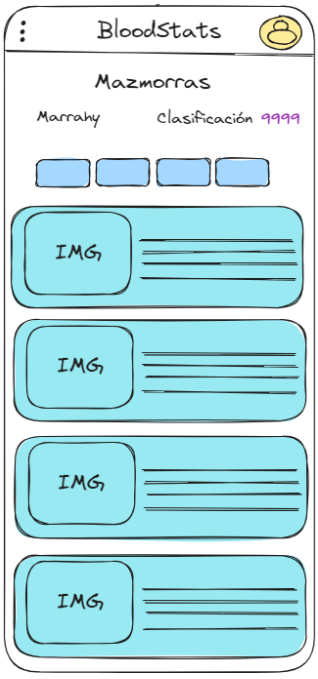


Figura 23 - Pantalla donde se muestran las mazmorras de la expansión actual

- **Pantalla de Opciones:** En esta pantalla el usuario podrá seleccionar el tema de la aplicación entre claro y oscuro.

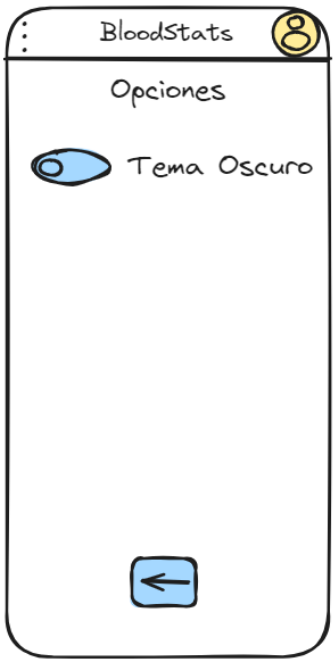


Figura 24 - Pantalla donde se muestran las opciones de la aplicación

- **Pantalla de Perfil:** En esta pantalla el usuario podrá interactuar con la lista de favoritos; haciendo clic en uno, el sistema navegará a la pantalla del personaje seleccionado. Podrá cerrar sesión y eliminar la cuenta permanentemente, así como toda la información almacenada relacionada con este usuario.

Figura 25 - Pantalla donde se muestra el perfil del usuario que ha iniciado sesión

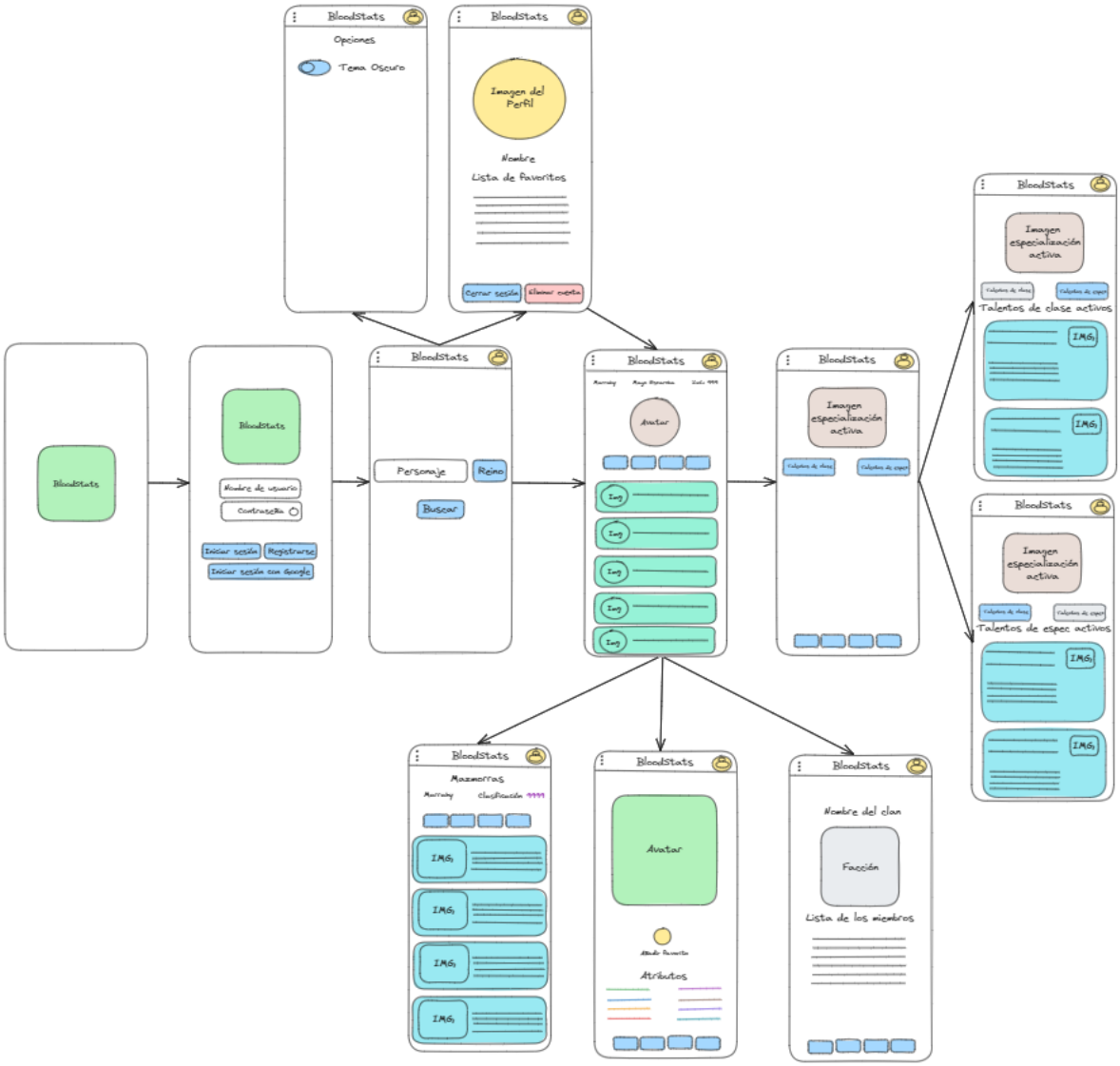


Figura 26 - Flujo de la navegación de las pantallas

4.3. Implementación y desarrollo

En este apartado se va a mostrar algunas funciones del proyecto para explicar el flujo de trabajo que se ha contemplado.

Metodología de trabajo - Ágil

Se ha decidido utilizar una metodología de trabajo Ágil por la fuerte capacidad para adaptarse a cambios en los requisitos, además, al ser un proyecto pequeño no requiere de tanta gestión para la escalabilidad que pueda llegar a necesitar. La forma de trabajar se basa en sprints y periodos cortos de trabajo intensivos.

Para la metodología de diseño se ha utilizado Modelo - Vista - VistaModelo (MVVM). El proyecto está divido entre el Modelo; Maneja la lógica de negocio y los datos de la app (SQLite, la API de Blizzard, la API de Google), la Vista; Es responsable de actualizar los cambios de datos en la UI, en este caso se utilizan Composables, y

ViewModel; Gestiona todos los datos que se actualizan en consecuencia a la interacción del usuario, en este caso se ha utilizado LiveData y StateFlow.

- Se ha utilizado la página web Logo Maker para hacer el logo oficial de la aplicación. Este sitio web utiliza una IA para generar varios logos en base al nombre del proyecto, de esta manera se ha conseguido recortar tiempo para darle más importancia a otras cosas más importantes.

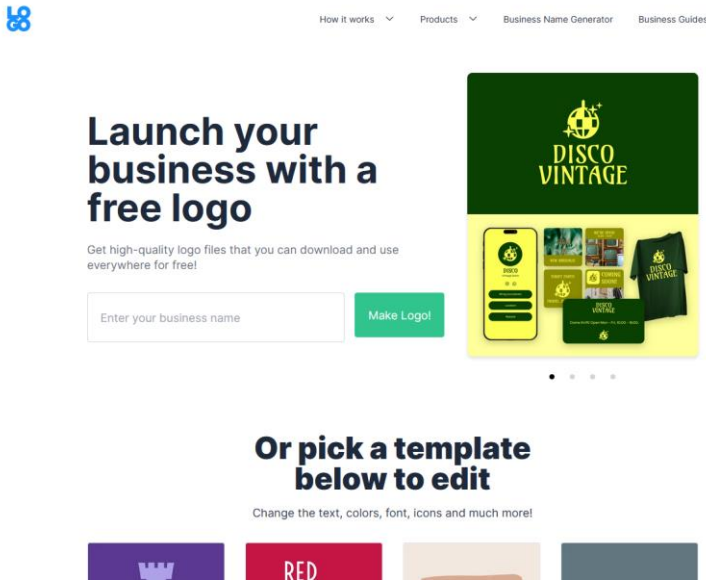


Figura 27 - Página web de Logo

- Para la selección de la paleta de colores se ha escogido la herramienta Material Fundation. Material Fundation ofrece una generación de temas personalizados para aplicaciones Android, pudiendo exportar la carpeta *ui* para reemplazar el tema por defecto que se genera al crear un proyecto. La herramienta incluye el tema oscuro en base a la paleta de colores elegida desde un tema claro y viceversa. Por otra parte, se puede escoger la fuente de la letra de la app y los iconos si se desea.

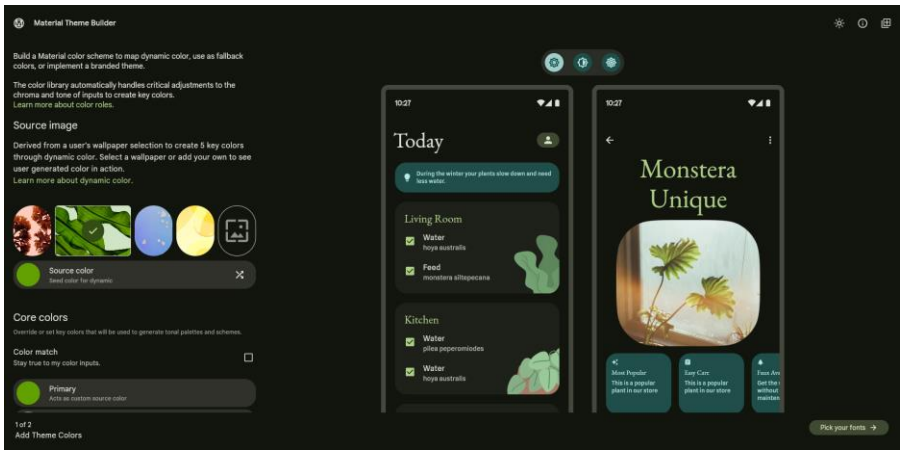


Figura 28 - Página web de Material Theme Builder

- Visual Estudio Code y Android Studio permiten el desarrollo de aplicaciones Android, pero finalmente se ha elegido el uso del IDE Android Studio, ya que está específicamente dedicado a ello. Ofrece un conjunto completo de herramientas y características optimizadas para la creación, depuración y despliegue de aplicaciones Android. Este IDE incluye un emulador de Android permitiendo probar la aplicación en diferentes tipos de dispositivos.

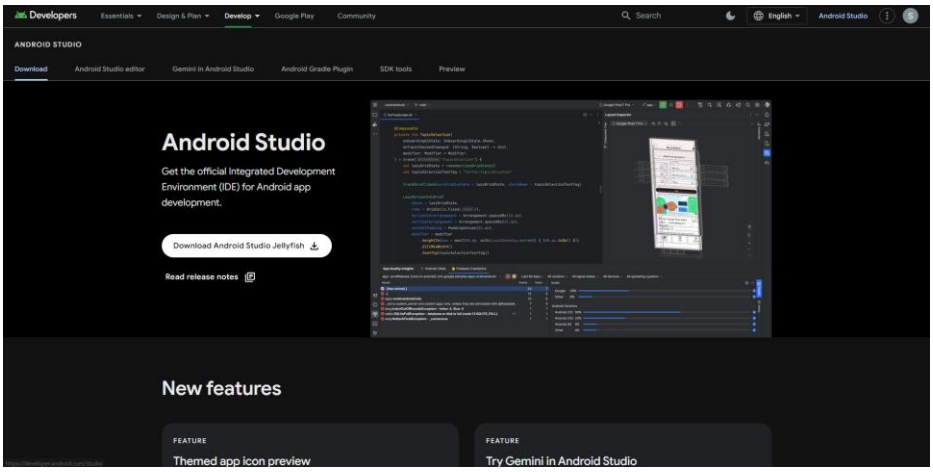


Figura 29 - Página web de Android Studio

- Se ha elegido Kotlin como lenguaje de programación por la concisión y legibilidad, reduciendo así la probabilidad de errores y mejorando la productividad. La seguridad que ofrece Kotlin frente a los nulos es una de las mayores razones ya que previene errores comunes como la excepción "*NullPointerException"*, lo que mejora la estabilidad y confiabilidad en la aplicación. El uso de coroutines y la programación asíncrona permiten que esta aplicación funcione correctamente por el uso de APIs.

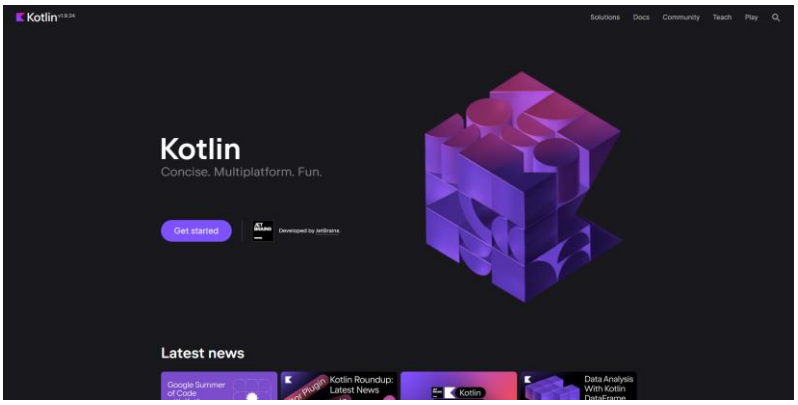


Figura 30 - Página web de Kotlin.

- Como JVM/emulador se ha decidido utilizar un dispositivo móvil Android de la marca Pixel, modelo 4 con el sistema operativo Android 14 (Target API Level 34 // MinTarget API Level 26) y con acceso a los servicios de Google. También se ha probado en los dispositivos Pixel 7 Pro y en el Pixel Tablet para revisar el diseño adaptativo.

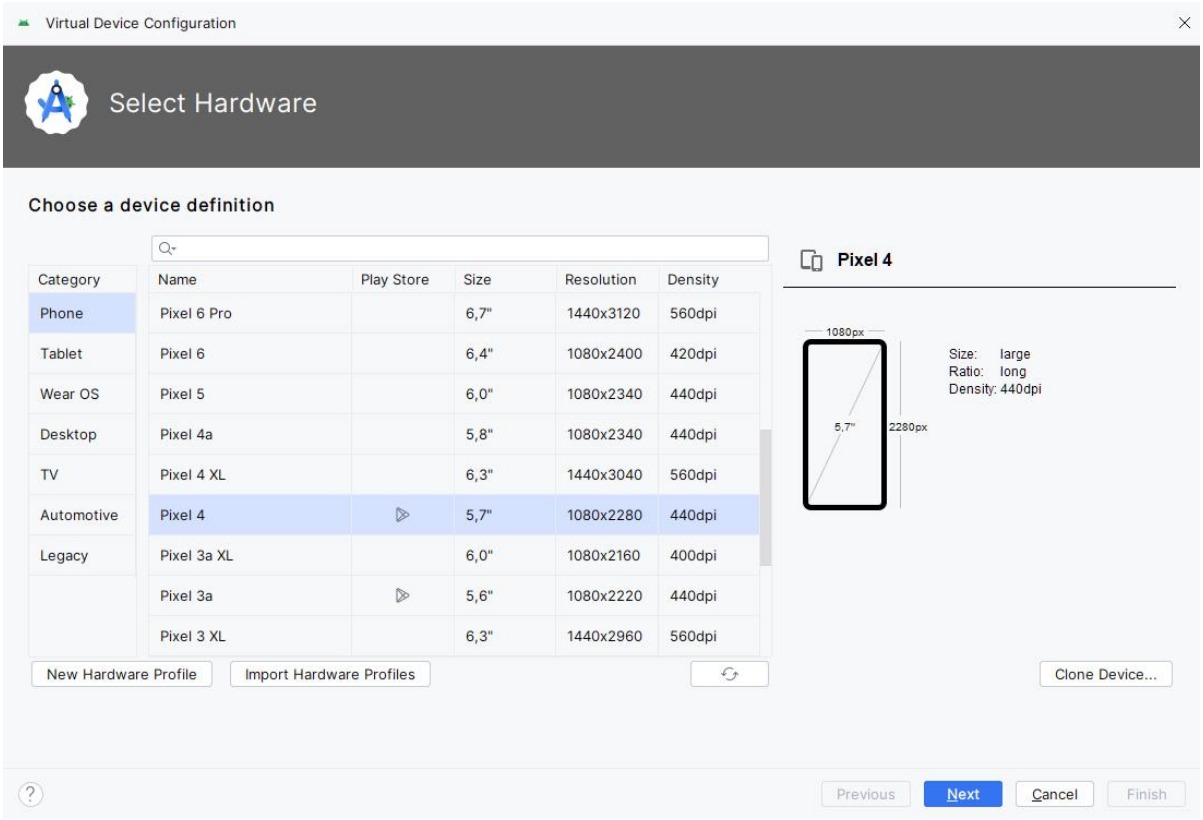


Figura 31 - Selector del emulador

Google Pixel 4	
Procesador	Qualcomm Snapdragon 855 con Adreno 640
Pantalla	5,7 pulgadas Flexible OLED 90Hz Resolución FHD+ 2280 x 1080p 80% de superficie útil
Memoria RAM	6 GB LPDDR4X
Conectividad	Bluetooth 5.0 LE Wi-Fi 802.11 b/g/n/ac 5 GHz GPS, Galileo, A-GPS, GLONASS, Beidou
Almacenamiento	64 GB no ampliables
Batería	2800 mAh Li-Polymer
Dimensiones	68,8 x 147,1 x 8,2 mm
Sistema Operativo	Android 7.0 N

La aplicación se ha probado en un dispositivo móvil real de la marca Redmi Note 5 con una versión de Android 9 para asegurar el correcto funcionamiento de todas las herramientas y sistemas de la aplicación.

- Como Base de Datos se utiliza SQLite por varias razones:
  i. **Persistencia de datos local:** Esto permite guardar las opciones de un usuario y cargarlas nada más inicia sesión).
  ii. **Compatibilidad y estabilidad:** Al estar integrada directamente al SO Android, garantiza la total compatibilidad con la aplicación.
  iii. **Eficiencia y rendimiento:** SQLite al ser una Base de Datos tan ligera y eficiente, permitirá el funcionamiento optimizado en dispositivos de gama baja o con recursos limitados.
  iv. **Flexibilidad y potencia:** SQLite admite consultas SQL complejas y transacciones ACID a pesar de ser una solución rápida y sencilla.
  v. **Integración con Jetpack Compose y Kotlin:** Jetpack Compose al ser una herramienta para el diseño de la GIU, no proporciona funcionalidades para la gestión de datos, es por esto por lo que con SQLite se integra perfectamente en esta aplicación haciendo uso de la librería Room, aplicando así una capa de abstracción y simplificando la interacción con la Base de Datos desde el código.

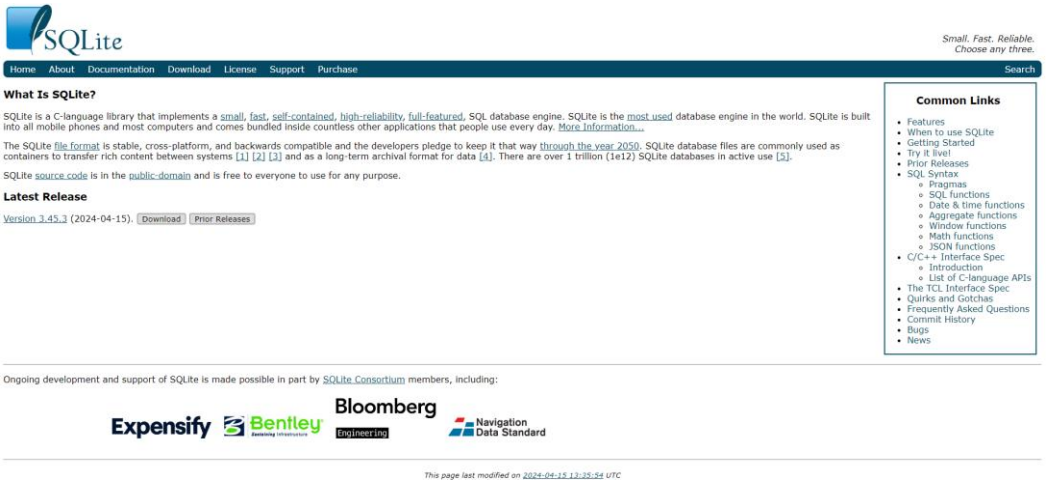


Figura 32 - Página web de SQLite

- Para agregar las dependencias que la aplicación necesita para funcionar correctamente se ha utilizado una herramienta de automatización de compilación, el Gradle. Su función principal es gestionar las dependencias y compilar el código fuente. En este proyecto se utiliza el Gradle como sistema de compilación principal donde los archivos son scripts de configuración escritos en Kotlin que definen cómo se compila, prueba y empaqueta el proyecto.

i. **com.google.gms.google-services:** Dependencia utilizada para integrar y configurar servicios de Firebase de manera sencilla.
ii. **com.squareup.retrofit2:retrofit:2.11.0:** Esta dependencia se encarga de las llamadas a la API de Blizzard definiendo interfaces de una manera simple e intuitiva.
iii. **com.squareup.retrofit2:converter-gson:2.11.0:** La dependencia *converter-gson* es utilizada para serializar o deserializar objetos Java en y desde formato JSON.
iv. **com.google.firebase:firebase-bom:32.8.1:** El BOM de Firebase simplifica la administración de dependencias de Firebase en el proyecto.
v. **com.google.firebase:firebase-auth-ktx:** Firebase auth es un servicio de Firebase que permite la autenticación de usuario en la aplicación de forma segura y rápida y además es específica de Kotlin. Hay muchas formas de configurar esta parte, pero se ha decidido solo configurar el inicio de sesión con el proveedor de Google.
vi. **com.google.android.gms:play-services-auth:21.1.0:** Esta dependencia es utilizada para la integración de la autenticación de Google en el proyecto. Permite el a los usuarios autenticarse en la app utilizando sus cuentas de Google.
vii. **android.room:room-runtime:2.6.1:** Con esta dependencia se integra Room en el proyecto. Room aplica una capa de abstracción sobre SQLite para que el acceso a la base de datos sea más fluido.
viii. **androidx.room:room-compiler:2.6.1:** Esta dependencia incluye el compilador de Room en el momento que se ejecuta el proyecto. Sin esta dependencia no se podrían aplicar ni procesar las anotaciones de Room y generar el código necesario para la compilación.

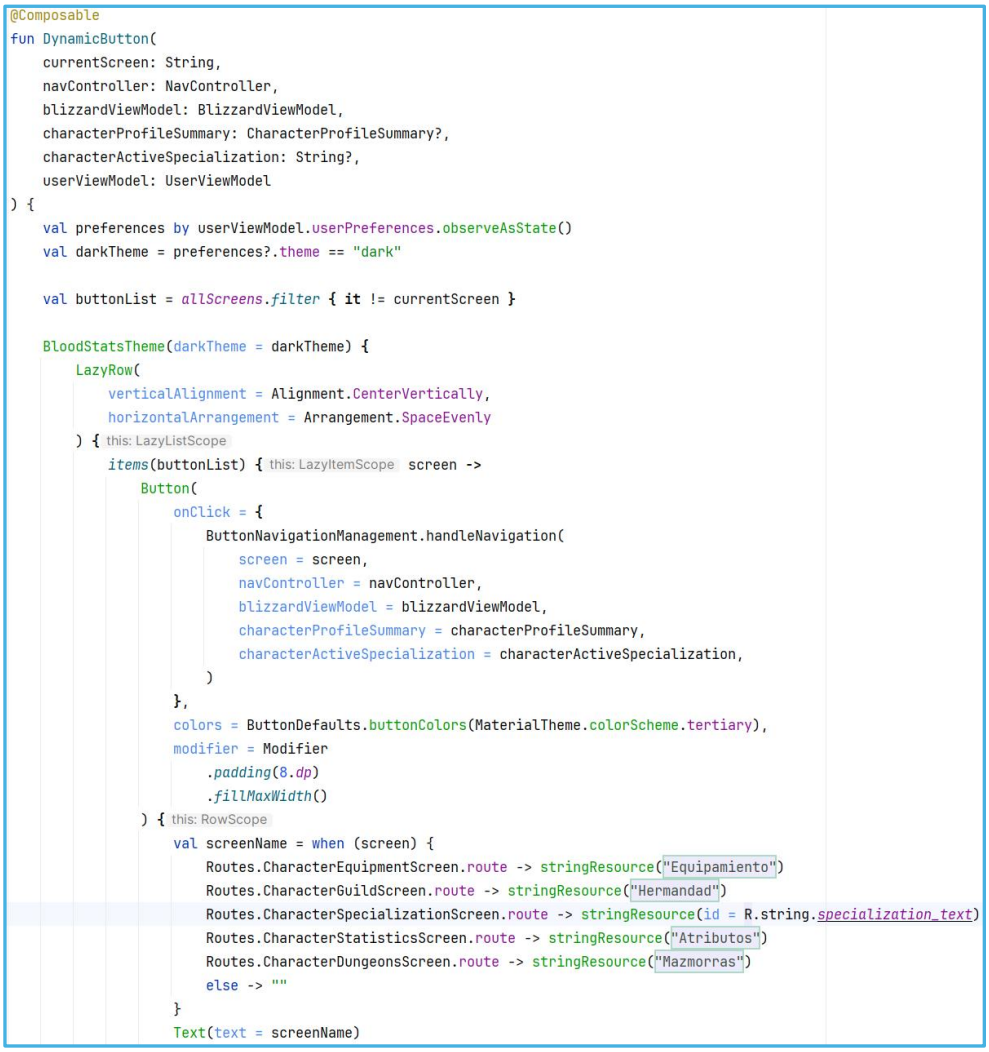
En esta captura se pueden ver las dependencias agregadas al proyecto para el correcto funcionamiento de los sistemas desarrollados.

```
//Navigation
implementation("androidx.navigation:navigation-compose:2.7.7")

//LiveData
implementation("androidx.compose.runtime:runtime-livedata:1.6.6")

//Retrofit2
implementation("com.squareup.retrofit2:retrofit:2.11.0")

//Serialize/Deserialize
implementation("com.squareup.retrofit2:converter-gson:2.11.0")

//LifeCycle
implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.7.0")
implementation("androidx.lifecycle:lifecycle-runtime-compose:2.7.0")

//Firebase
implementation(platform("com.google.firebase:firebase-bom:32.8.1"))
implementation("com.google.firebase:firebase-auth-ktx")
implementation("com.google.android.gms:play-services-auth:21.1.0")
implementation("com.google.firebase:firebase-auth")

//Icons
implementation("androidx.compose.material:material-icons-extended:1.5.4")

//Room
implementation("androidx.room:room-runtime:2.6.1")
ksp("androidx.room:room-compiler:2.6.1")
implementation("androidx.room:room-ktx:2.6.1")

//Profile picture remote load
implementation("io.coil-kt:coil-compose:2.5.0")

//Responsive
implementation("androidx.compose.material3:material3-window-size-class-android:1.2.1")
```

Figura 33 - Algunas de las dependencias agregadas al proyecto

Gestión de la navegación entre ventanas

Se ha desarrollado una forma dinámica de la navegación para que sea todo lo intuitiva y amigable posible.



```
@Composable
fun DynamicButton(
    currentScreen: String,
    navController: NavController,
    blizzardViewModel: BlizzardViewModel,
    characterProfileSummary: CharacterProfileSummary?,
    characterActiveSpecialization: String?,
    userViewModel: UserViewModel
) {
    val preferences by userViewModel.userPreferences.observeAsState()
    val darkTheme = preferences?.theme == "dark"

    val buttonList = allScreens.filter { it != currentScreen }

    BloodStatsTheme(darkTheme = darkTheme) {
        LazyRow(
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceEvenly
        ) { this: LazyListScope
            items(buttonList) { this: LazyItemScope screen ->
                Button(
                    onClick = {
                        ButtonNavigationManagement.handleNavigation(
                            screen = screen,
                            navController = navController,
                            blizzardViewModel = blizzardViewModel,
                            characterProfileSummary = characterProfileSummary,
                            characterActiveSpecialization = characterActiveSpecialization,
                        )
                    },
                    colors = ButtonDefaults.buttonColors(MaterialTheme.colorScheme.tertiary),
                    modifier = Modifier
                        .padding(8.dp)
                        .fillMaxWidth()
                ) { this: RowScope
                    val screenName = when (screen) {
                        Routes.CharacterEquipmentScreen.route -> stringResource("Equipamiento")
                        Routes.CharacterGuildScreen.route -> stringResource("Hermandad")
                        Routes.CharacterSpecializationScreen.route -> stringResource(id = R.string.specialization_text)
                        Routes.CharacterStatisticsScreen.route -> stringResource("Atributos")
                        Routes.CharacterDungeonsScreen.route -> stringResource("Mazmorras")
                        else -> ""
                    }
                    Text(text = screenName)
```

Figura 34 - Composable para la navegación dinámica entre pantallas de un personaje

Inicio de sesión y creación de usuarios con Google

En este bloque de código se muestra la creación de una cuenta de la app con la función Sign In que ofrece Firebase

```
94        val state by googleViewModel.state.collectAsStateWithLifecycle()
95        val launcher = rememberLauncherForActivityResult(
96            contract = ActivityResultContracts.StartIntentSenderForResult(),
97            onResult = { result ->
98                if (result.resultCode == Activity.RESULT_OK) {
99                    coroutineScope.launch { this: CoroutineScope
100                       val signInResult = googleAuthUiClient.signInWithIntent(
101                           intent = result.data ?: return@launch
102                       )
103                       googleViewModel.onSignInResult(signInResult)
104                       val email = signInResult.data?.userEmail ?: return@launch
105                       val userName = signInResult.data.username ?: "defaultName_${UUID.randomUUID()}"
106                       val userPassword = signInResult.data.userId
107                       val avatar = R.drawable.murlok
108
109                       userViewModel.createIfNotExists(
110                           userEmail = email,
111                           userName = userName,
112                           userPassword = userPassword,
113                           avatar = avatar
114                       ) { user ->
115                           userViewModel.saveUser(user)
116                           userViewModel.getUserWithPreferences(user.userUUID)
117                           navController.navigate(route = Routes.SearchScreen.route)
118                       }
119                   }
120               }
121           }
122       )
```

Figura 35 - Creación e inicio de sesión de un usuario

A la hora de crear un usuario de una forma tradicional se utilizan los campos “Nombre de usuario” y “Contraseña”.

```
197            OutlinedTextField(
198                value = userName,
199                onValueChange = { userName = it },
200                label = { Text(text = stringResource("Nombre de usuario")) },
201                singleLine = true,
202                colors = TextFieldDefaults.outlinedTextFieldColors(
203                    focusedBorderColor = MaterialTheme.colorScheme.tertiary,
204                    unfocusedBorderColor = MaterialTheme.colorScheme.onSurface,
205                    focusedLabelColor = MaterialTheme.colorScheme.primary,
206                    cursorColor = MaterialTheme.colorScheme.primary
207                )
208            )
209
210            Spacer(modifier = Modifier.height(8.dp))
211
212            OutlinedTextField(
213                value = userPassword,
214                onValueChange = { userPassword = it },
215                label = { Text(text = stringResource("Contraseña")) },
216                singleLine = true,
217                visualTransformation = if (passwordIsVisible) VisualTransformation.None else PasswordVisualTransformation(),
218                keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Password),
219                trailingIcon = {
220                    val image =
221                        if (passwordIsVisible) Icons.Filled.Visibility else Icons.Filled.VisibilityOff
222                    IconButton(
223                        onClick = {
224                            passwordIsVisible = !passwordIsVisible
225                        }
226                    ) {
227                        Icon(
228                            imageVector = image,
229                            contentDescription = if (passwordIsVisible) "Hide password" else "Show password"
230                        )
231                    }
232                },
233                colors = TextFieldDefaults.outlinedTextFieldColors(
234                    focusedBorderColor = MaterialTheme.colorScheme.tertiary,
235                    unfocusedBorderColor = MaterialTheme.colorScheme.onSurface,
236                    focusedLabelColor = MaterialTheme.colorScheme.primary,
237                    cursorColor = MaterialTheme.colorScheme.primary
238                )
239            )
```

Figura 36 - Creación de un usuario de la app

Búsqueda de un personaje

Para la búsqueda de un personaje se recoge el input del usuario, tanto el nombre del personaje como el reino para así realizar la petición a la API de Blizzard con esos valores. El Composable SearchBox también carga la lista de reinos disponibles en Europa Central para que el usuario pueda seleccionar el reino donde esté alojado.

```
96                    ) { this: ColumnScope
97                        SearchBox(
98                            blizzardViewModel = blizzardViewModel,
99                            searchedEntity = searchedEntity,
100                           onNameChange = { searchedEntity = it },
101                           onRealmChange = { realm = it },
102                           onClickPress = {
103                               if (searchedEntity.isNotBlank() && realm.isNotBlank()) {
104                                   navController.navigate(route = Routes.LoadingScreen.route)
105                                   blizzardViewModel.clearCharacterData()
106                                   blizzardViewModel.loadCharacterProfileSummaryEquipmentMedia(
107                                       searchedEntity,
108                                       realm
109                                   )
110                               } else {
111                                   showError = !showError
112                               }
113                           },
114                           userViewModel = userViewModel
115                       )
116                   }
117               }
118           }
119       )
          }
121   }
```

Figura 37 - Composable para la búsqueda de un personaje.

Desde BlizzardViewModel se instancia el Cliente de Retrofit para hacer la llamada a las funciones que construyen la URL para mandar la petición a la API de Blizzard y así deserializar los JSON que devuelve el servidor y montar los objetos generados para estructurar los datos en pantalla. Todos estos datos se guardan en variables LiveData para no tener que volver a hacer la petición y hacer así la aplicación más escalable. Toda esta ejecución es un proceso asíncrono ya que la respuesta del servidor no es inmediata.

```
116     /**
117      * This function makes various retrofit request to Blizzard's API and gets the ProfileSummary, CharacterMedia,
118      * CharacterGuild, Character primary stat, CharacterEquipment and the CharacterMedia
119      *
120      * @param characterName Name of the searched character
121      * @param realmSlug Realm of the searched character
122      */
123     fun loadCharacterProfileSummaryEquipmentMedia(characterName: String, realmSlug: String) {
124         clearCharacterData()
125         viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
126             _isLoading.postValue( value: true)
127             try {
128                 _characterProfileSummary.postValue(
129                     retrofitService.getCharacterProfileSummary(
130                         accessToken = accessToken.value!!,
131                         characterName = characterName,
132                         realmSlug = realmSlug
133                     )
134                 )
135
136                 _characterEquipment.postValue(
137                     retrofitService.getCharacterEquipment(
138                         accessToken = accessToken.value!!,
139                         characterName = characterName,
140                         realmSlug = realmSlug,
141                         locale = deviceLanguage.value!!
142                     )
143                 )
144
145                 _characterMedia.postValue(
146                     retrofitService.getCharacterMedia(
147                         accessToken = accessToken.value!!,
148                         characterName = characterName,
149                         realmSlug = realmSlug,
150                         locale = deviceLanguage.value!!
151                     )
152                 )
153
154                 _characterEquipment.value?.equipped_items?.let { equippedItems ->
155                     _equippedItems.postValue(equippedItems)
156                     equippedItems.forEach { equippedItem ->
157                         retrofitService.getItemDataById(
158                             accessToken = accessToken.value!!,
159                             itemId = equippedItem.item.id,
```

Figura 38 - Función que recoge datos del personaje buscado y los guarda en variables LiveData

Todas las pantallas reciben y guardan los datos con funciones similares a la de la Figura 45.

4.4. Pruebas

Durante el desarrollo de la aplicación BloodStats han surgido muchos problemas de implementación de varias herramientas:

Firebase authentication: En esta implementación es obligatorio proporcionar al servicio de Firebase un SHA-1 (Función criptográfica de hash para agregar seguridad y gestionar la integración de datos). Para poder obtener este SHA-1 hay varias maneras de hacerlo, pero la más sencilla es ejecutando un proceso que el Gradle de Android Studio proporciona para generar el SHA-1. Este proceso se encuentra en las herramientas Gradle → app → Tareas → Android, en este path se encuentra el proceso *signinReport*, al ejecutar este proceso te muestra por consola el SHA-1 del proyecto el cual se debe proporcionar al servicio de Firebase para vincular el proyecto y funcionen correctamente los servicios de Google.

Retrofit 2: Anteriormente para generar las peticiones a la API y recibir las respuestas, se utilizaba la librería OkHttp para entender cómo funcionaba todo el proceso que esta Retrofit 2 facilita a la hora de implementarla, esta librería agrega una capa de abstracción extra que simplifica la definición de las peticiones que queramos hacer. El problema reside en el contenido ofrecido por la comunidad y la escasa documentación en la página web de Retrofit, han generado un gran problema a la hora de implementar la petición de la generación del token de acceso que Blizzard necesita obligatoriamente para hacer uso de su API por temas de seguridad. Al no haber muchos ejemplos o proyectos parecidos para poder aprender de ello, se han hecho muchas pruebas para poder generar el token y poder realizar las peticiones correctamente.

Documentos JSON extensos: Cuando se generaba una petición a la API de Blizzard de tipo SEARCH (quiere decir que devuelve todas las coincidencias que tenga con el valor pasado) el documento JSON que devuelve el servidor es muy amplio, por lo que se tuvo que replantear la recogida de datos para no sobrecargar el dispositivo con datos innecesarios que en un principio ni si quiera se iban a ver. Además de esto, se quería desarrollar el generar una petición para conseguir la imagen de la entidad por cada objeto que existiera en ese documento, y esto no era nada viable, ya que se podía llegar a las 100 peticiones que limita Blizzard por segundo. La solución a este problema fue simplemente no realizar las peticiones para conseguir las imágenes de las entidades hasta que el usuario no hiciese clic en un objeto, de esta manera se realizan N peticiones dependiendo de la cantidad de ítems equipados que tenga el personaje hasta un máximo de 16 para conseguir el nombre e imágenes de todos los objetos proporcionados, entonces cuando el usuario quiera ver un objeto en específico se generará una petición para mostrar la imagen de la entidad junto a su descripción.

Atributos desconocidos: Uno de los problemas que surgieron al desarrollar la pantalla del equipamiento de un personaje, fue el relacionar el atributo principal del personaje con la clase y especialización específica. Se hizo un Excel para tener claro que atributo era correspondiente con la Clase y Especialización Activa, la tabla siguiente muestra la Clase, Especialización, el Rol y el Atributo Primario en correspondencia:

Clase	Especialización	Rol	Atributo Primario
Caballero de la muerte	Sangre	Tanque	Fuerza
Caballero de la muerte	Escarcha	DPS	Fuerza
Caballero de la muerte	Profano	DPS	Fuerza
Paladín	Protección	Tanque	Fuerza
Paladín	Represalia	DPS	Fuerza
Guerrero	Armas	DPS	Fuerza
Guerrero	Furia	DPS	Fuerza
Guerrero	Protección	Tanque	Fuerza
Cazador de demonios	Estragos	DPS	Agilidad
Cazador de demonios	Venganza	Tanque	Agilidad
Druida	Feral	DPS	Agilidad
Druida	Guardián	Tanque	Agilidad
Monje	Maestro cervecero	Tanque	Agilidad
Monje	Caminante del viento	DPS	Agilidad
Cazador	Maestro de bestias	DPS	Agilidad
Cazador	Puntería	DPS	Agilidad
Cazador	Supervivencia	DPS	Agilidad
Pícaro	Asesino	DPS	Agilidad
Pícaro	Forajido	DPS	Agilidad
Pícaro	Sutileza	DPS	Agilidad
Chamán	Mejora	DPS	Agilidad
Druida	Balance	DPS	Intelecto
Druida	Restauración	Sanador	Intelecto
Evocador	Devastación	DPS	Intelecto
Evocador	Conservación	Sanador	Intelecto
Mage	Arcano	DPS	Intelecto
Mago	Fuego	DPS	Intelecto
Mago	Escarcha	DPS	Intelecto
Monje	Tejedor de niebla	Sanador	Intelecto

Paladín	Sagrado	Sanador	Intelecto
Sacerdote	Disciplina	Sanador	Intelecto
Sacerdote	Sagrado	Sanador	Intelecto
Sacerdote	Sombra	DPS	Intelecto
Chamán	Elemental	DPS	Intelecto
Chamán	Restauración	Sanador	Intelecto
Brujo	Aflicción	DPS	Intelecto
Brujo	Demonología	DPS	Intelecto
Brujo	Destrucción	DPS	Intelecto

Esta información se ha recogido en la página Atributos Primarios hecha y actualizada diariamente por la comunidad. Para solucionar este problema se hizo un hashmap en una función que devuelve el atributo principal correspondiente.

4.5. Documentación de la app

En este apartado se van a mencionar los recursos que complementan este proyecto:

1. Instrucciones para la instalación del APK en un dispositivo móvil Android.
2. Instrucciones para replicar el entorno de trabajo y ejecutar el código fuente en un equipo.
3. Memoria del proyecto en formato PDF.
4. Archivo ZIP del proyecto.

Con estos documentos se podrá compilar el proyecto desde otro equipo que previamente tenga instalado Android Studio, un emulador preferentemente con acceso a los servicios de Google, Android 14 o inferior (hasta Android 8 Oreo) con la API targetSdkVersion 34 y minSdkVersion 26.

5. Trabajos futuros

En este apartado se van a mencionar las mejoras y futuras implementaciones que se podrían desarrollar para mejorar la aplicación:

Chat: Como primer trabajo futuro se propone desarrollar un sistema de amigos en el cual dos usuarios podrían mantener una conversación para compartir ideas, opiniones e incluso enlazar la búsqueda de un objeto, ya sea un personaje, artículo o NPC.

Planificador de equipo: Si un usuario ya experimentado en el juego quisiera crear y teorizar un equipo, este sistema podría cubrir este requisito. Este sistema tiene varias funcionalidades:

- Aplicar filtros basados en la clase y la especialización del personaje. De esta manera, el usuario no tendrá que divagar en la ingente cantidad de piezas de equipo existentes en el juego.
- Agregar piezas de equipo en cada hueco que se muestra en la pantalla, estos huecos están ordenados desde la cabeza hasta las botas, en total habrían 16 huecos; Casco, Colgante, Pechera, Capa, Brazaletes, Pulsera, Guantes, Anillos x2, Misceláneos x2, Cinturón, Pantalones, Botas, Arma principal, Arma secundaria.

Mejoras de UI y UX: Para hacer que la aplicación sea mucho más vistosa y agradable de utilizar, se recomendaría mejorar la experiencia de usuario y la interfaz gráfica.

Mantener una base de datos con toda la información del WoW: Para un futuro esta opción si fuese viable, sería muy atractiva ya que se podría desarrollar una base de datos y una API para crear una aplicación muchísimo más completa. De esta forma, problemas como la falta de información en las APIs de Blizzard se solucionarían ya que simplemente junto con la API desarrollada se harían consultas a nuestra base de datos donde se alojaría toda la información del WoW.

6. Conclusiones

En conclusión, este proyecto de fin de curso ha servido para adquirir muchos conocimientos sobre el desarrollo de aplicaciones Android y sobre el consumo de APIs utilizando varias tecnologías actuales, ya que, en un futuro, serán de gran ayuda. También, se han aprendido los procesos que se realizan cuando se mandan peticiones, URLs, el tipo de verificación OAuth 2.0 para el acceso a toda la información que se necesite consumir con Retrofit 2, el desarrollo de la base de datos en Android implementando Room, el uso de Coroutines y la gestión de hilos.

La aplicación tiene muchos aspectos mejorables, pero es una buena base por dónde empezar a desarrollar otras funciones para completarla y que sea lo más versátil posible, es decir, que tanto jugadores nuevos como jugadores experimentados, le saquen partido para mejorar en cualquier aspecto sus personajes.

Se han presentado muchos desafíos durante todo el proceso de desarrollo de BloodStats, cómo, por ejemplo, el uso de la API de Blizzard, que ofrecen una API y una documentación muy extensa con muchas formas de recoger datos en varios idiomas y de muchas formas. Se ha optimizado todo lo posible la realización de las peticiones para que la aplicación tenga una mayor escalabilidad, y varios usuarios simultáneos puedan hacer uso de ella sin problemas.

Finalmente, este proyecto ha sido una gran experiencia que nos ha enriquecido con mucho conocimiento para en un futuro, afrontar los posibles problemas que puedan surgir durante el trayecto laboral y profesional.

7. Bibliografía y webgrafía

En este apartado se indican todas las páginas web con los autores y el día que se ha visitado para su uso y consumo:

Android Developers (17 de marzo 2024) Cómo comenzar a usar Jetpack Compose. Developer Android. https://developer.android.com/develop/ui/compose/documentation

Firebase Developers (25 de marzo de 2024) Autenticación con Google en Android. https://firebase.google.com/docs/auth/android/google-signin

Blizzard Developers (1 de abril de 2024) Utilizando OAuth. Battlenet

https://develop.battle.net/documentation/guides/using-oauth

Blizzard Developers (1 de abril de 2024) Flujo de las credenciales del cliente. Battlenet

https://develop.battle.net/documentation/guides/using-oauth/client-crendetials-flow

Blizzard Developers (1 de abril de 2024) Flujo del código de autorización. Battlenet

https://develop.battle.net/documentation/guides/using-oauth/authorization-code-flow

Tony (2 de abril de 2024) Las tareas del Gradle no se muestran en la ventana de herramientas del Gradle en Android Studio 4.2. stackoverflow.

https://stackoverflow.com/questions/67405791/gradle-tasks-are-not-showing-in-the-gradle-tool-window-in-android-studio-4-2

Raithatha K. (4 de abril de 2024) Cómo consumir una APIs en Jetpack Compose. Medium. https://medium.com/@kathankraithatha/how-to-use-api-in-jetpack-compose-10d11b8f166f

Tovar D. (1 de mayo de 2024) Seleccionar, Insertar, Indexar y Claves Ajenas en las migraciones de Room. Medium. https://medium.com/knowing-android/select-insert-indexes-and-foreign-keys-on-room-migrations-2a0dc556efd3

Mudadla S. (3 de mayo de 2024) Coroutines en Jetpack Compose. Medium. https://medium.com/@sujathamudadla1213/coroutines-with-jetpack-compose-8e919cbf806f

Brian M. (6 de mayo de 2024) Guía para desarrolladores: Masterizando la arquitectura MVVM en Android. Medium. https://medium.com/@mutebibrian256/mastering-android-mvvm-architecture-developers-guide-3271e4c8908b

Bisset K. (27 de marzo de 2024) MutableState o MutableStateFlow: Una perspectiva en que utilizar en Jetpack Compose. Medium. https://medium.com/@kerry.bisset/mutablestate-or-mutablestateflow-a-perspective-on-what-to-use-in-jetpack-compose-ccec0af7abbf

Elliott T. (10 de abril de 2024) Utilizando OAuth2.0 y Retrofit para comunicarse a la API de GitHub en Android. Dev.to. https://dev.to/theplebdev/using-oauth20-and-retrofit-to-talk-to-the-github-api-on-android-k6h

ActivePlayer (6 de abril de 2024) World of Warcraft: Jugadores activos y estadísticas. ActivePlayer.io. https://activeplayer.io/world-of-warcraft/

Lackner P. (25 de marzo de 2024). Firebase Google Sign-In con Jetpack Compose y Clean Architecture – Android Studio Tutorial. YouTube. https://www.youtube.com/watch?v=zCIfBbm06QM&t=1851s

Logo (15 de mayo de 2024). Lanza tu negocio con un logo gratuito. Logo. https://logo.com/

Margeeta (15 de mayo de 2024). Material Theme Builder. Material Fundation. https://material-foundation.github.io/material-theme-builder/

Comunidad de Juegos Fandom (11 de mayo de 2024). Atributos. WoWPedia. https://wowpedia.fandom.com/wiki/Attributes

Guiterrez D. (13 de mayo de 2024). UML Casos de Uso. SlideShare. https://es.slideshare.net/slideshow/clase-11-umlcasosdeuso/20600612

Arif A. (28 de mayo de 2024). Cómo generar un APK y APKs firmadas en Android Studio. EnvatoTuts. https://code.tutsplus.com/how-to-generate-apk-and-signed-apk-files-in-android-studio--cms-37927t

Android Developers (28 de mayo de 2024). Cambiar el icono de la aplicación. Developer Android. https://developer.android.com/codelabs/basic-android-kotlin-compose-training-change-app-icon#4

Anexos:

Se proporcionan varias carpetas para la replicar el proyecto e instalar esta aplicación:

1. Carpeta *Manual de Instalación/Ejecución en un dispositivo Android.*
2. Carpeta *Instrucciones para replicar el entorno de desarrollo.*