***Hay que tener en cuenta que este no es el estilo final del documento. En la entrega final, cuando estén todos los apartados completos haré el documento con el estilo requerido.***

# 3. Arquitectura.

## 3.1. Topología de Servidores y Apps:

### Diagrama de Arquitectura:

La aplicación móvil (desarrollada en Android Studio) estará conectada a Firebase para la autenticación de usuarios, la base de datos en tiempo real y la gestión de datos. La comunicación con Firebase se realizará mediante la SDK de Firebase, que ofrece soporte para operaciones CRUD sobre la base de datos y autenticación.

- **Aplicación Móvil Android** con **Firebase Authentication** para autenticación de usuarios.

- **Aplicación Móvil Android** con **Firebase Firestore** para almacenar y recuperar datos de las builds y el estado de los planetas.

### Descripción de la Topología:

El flujo de datos será el siguiente:

1. **Consulta de los planetas:**

    - La aplicación móvil realiza una solicitud a la API de Helldivers para obtener los estados de los planetas. La respuesta es procesada y presentada al usuario.

2. **Subida de Builds:**

    - Cuando un usuario crea o sube una build, la aplicación realiza una operación en Firebase Firestore/Realtime Database para guardar los datos de la build (como las armas, armaduras, estratagemas, etc.).

3. **Sistema de Likes/Dislikes:**

    - Los usuarios pueden interactuar con las builds de otros usuarios, registrando un "like" o "dislike". Esto se realiza mediante la modificación de un campo en el documento correspondiente a la build en Firestore/Realtime Database.

## Especificación de Tecnologías:

- **Aplicación Móvil:**
    - **Android Studio 4.2**, **Kotlin 1.5**, **Firebase SDK 28.0**.
- **Base de Datos:**
    - **Firebase Firestore** (para almacenar las builds, datos de usuarios, etc.).
- **Autenticación:**
    - **Firebase Authentication** para la autenticación de usuarios.

## Justificación de las Tecnologías:

- **Firebase**: Elegido por su escalabilidad, facilidad de integración con Android y capacidad de manejo de datos en tiempo real.
- **Android Studio**: La opción natural para desarrollar aplicaciones Android, ya que es la plataforma oficial para crear apps nativas.
- **Firebase Authentication**: Proporciona un sistema robusto y fácil de integrar para gestionar la autenticación de usuarios de manera segura.

## Consideraciones de Seguridad:

- **Firebase** asegura la transmisión de datos mediante HTTPS.
- **Reglas de seguridad de Firebase** se configurarán para garantizar que los usuarios solo puedan acceder a los datos relevantes para ellos.
- **Validación en el lado del cliente y servidor** para evitar la manipulación de datos.

## Consideraciones de Escalabilidad:

- **Firebase** se escala automáticamente según el tráfico, lo que garantiza un rendimiento óptimo incluso con un aumento en el número de usuarios.

- La base de datos está diseñada para soportar consultas eficientes a gran escala, aprovechando las capacidades de Firestore/Realtime Database.

# 3.2. Descripción de las Apps.

### Descripción Detallada de la Aplicación Móvil (Android):

## Funcionalidades Principales:

- **Pantalla Principal:**
  La pantalla principal mostrará el estado de los planetas, con un **botón central** que servirá para consultar los estados de los planetas a través de la API de Helldivers 2.

- **Botón Izquierdo - Builds de Jugadores:**
  Este botón llevará a la vista que muestra las **builds** subidas por los jugadores. Los usuarios podrán ver, dar "like" o "dislike" a las builds de otros usuarios. Además, en la esquina inferior de esta pantalla, habrá un **botón flotante (+)** que permitirá a los usuarios subir sus propias builds.

- **Botón Derecho - Major Orders:**
  Este botón permitirá ver las **Major Orders**, proporcionando una lista o vista de las órdenes principales que están disponibles.

## Interfaz de Usuario (UI):

- **Barra de Navegación:**
  La barra de navegación constará de tres botones con íconos:

  1. **Botón Central:** Estado de los planetas.

  2. **Botón Izquierdo:** Builds de jugadores (con un ícono correspondiente).

  3. **Botón Derecho:** Major Orders (con su ícono).

- Cada botón tendrá un ícono representativo, y la interfaz utilizará Material Design para mantener una experiencia de usuario intuitiva y consistente.

- **Pantalla de Builds:**
  La pantalla de builds tendrá un diseño tipo lista, donde se mostrará la información de cada build. El botón flotante (+) en la esquina inferior permitirá a los usuarios añadir nuevas builds. Esta funcionalidad se basará en una interacción fluida y fácil de usar.

## Experiencia de Usuario (UX):

- La aplicación será intuitiva, con botones de navegación accesibles y fáciles de entender.

- Las animaciones de transición serán suaves para mejorar la experiencia.

- Al subir o ver una build, se podrá ver un listado con la opción de dar "like" o "dislike" a otras builds, haciendo que la interacción sea dinámica.

## Interacción con la API:

- La aplicación se conectará a la **API de Helldivers 2** para consultar los estados de los planetas. Además, se comunicará con Firebase para gestionar la autenticación de usuarios y para interactuar con las builds subidas por los usuarios (subir nuevas builds, dar like/dislike).

- Retrofit se utilizará para hacer las solicitudes HTTP a la API de Helldivers y Firebase para el manejo de la base de datos y autenticación.

## Gestión de Sesiones:

- Los usuarios podrán iniciar sesión mediante **Firebase Authentication**. Además, se implementará la opción de cerrar sesión y de recordar al usuario.

## 3.3. Diagrama de despliegue.

En este proyecto, el sistema está compuesto por una aplicación móvil Android que interactúa con varios servicios en la nube, incluyendo Firebase y una API externa (Helldivers).

La **aplicación móvil Android** se despliega en los dispositivos móviles de los usuarios en formato APK o AAB. Esta aplicación utiliza la SDK de Firebase para interactuar con los servicios en la nube, como **Firestore** para almacenar y recuperar datos de los usuarios (por ejemplo, las builds subidas por los jugadores) y **Firebase Authentication** para gestionar la autenticación de los usuarios. La comunicación con **Firestore** es en tiempo real, lo que garantiza que las actualizaciones, como la subida de nuevas builds, se sincronicen de inmediato entre los dispositivos. La autenticación también se realiza de forma segura a través de Firebase, permitiendo que los usuarios se registren, inicien sesión y mantengan su sesión activa de manera eficiente.

Adicionalmente, la **aplicación móvil** realiza peticiones HTTP a la **API de Helldivers** para obtener información actualizada sobre los planetas y sus características. Esta API RESTful devuelve respuestas en formato JSON, que luego son procesadas y utilizadas dentro de la aplicación móvil. La API de Helldivers proporciona datos dinámicos como el estado de los planetas, que pueden variar con el tiempo, lo que justifica el uso de esta fuente externa de datos.

La **API de Helldivers** se encuentra en un servidor remoto y está diseñada para responder a las solicitudes HTTP con información en formato JSON. La aplicación móvil realiza solicitudes GET para obtener los datos de los planetas, que se actualizan regularmente según la información disponible en la API externa. Esto permite que los usuarios siempre tengan acceso a datos precisos y actualizados sobre los planetas y sus estados.

El **servicio de Firebase** maneja la infraestructura de la base de datos en tiempo real (Firestore) y la autenticación de los usuarios (Firebase Authentication). Al elegir Firebase, se aprovecha su escalabilidad y sus funcionalidades en tiempo real para asegurar que los datos de los usuarios y las actualizaciones de las builds se sincronicen automáticamente entre todos los dispositivos. Además, Firebase es una solución de fácil implementación que permite evitar la necesidad de gestionar servidores adicionales, lo que reduce la complejidad del sistema.

En cuanto a la **seguridad**, Firebase proporciona reglas de seguridad para Firestore, lo que asegura que los datos de los usuarios estén protegidos y que cada usuario solo tenga acceso a su propia información, como sus builds. Además, **Firebase Authentication** garantiza que solo los usuarios autenticados puedan interactuar con los servicios de la aplicación, protegiendo así la privacidad y la integridad de los datos.

En resumen, la arquitectura de este sistema se basa en una aplicación móvil que se conecta tanto a Firebase para la gestión de usuarios y almacenamiento en tiempo real como a la API de Helldivers para obtener información externa sobre los planetas. Esta integración permite una experiencia de usuario fluida y dinámica, con datos actualizados y sincronizados automáticamente.

# 4. Diseño.

## 4.1. Modelo de datos. Firebase Store.

Se han definido tres colecciones principales para organizar los datos de forma flexible y escalable:

### Colección: usuarios

Contiene un documento por cada usuario registrado.

```
{
  "userID": "usuario123",
  "email": "usuario123@gmail.com",
  "profilePic": "https://.../profile.jpg",
  "signUpDate": "2025-04-27T14:20:00Z"
}
```

### Colección: builds

Contiene builds personalizadas subidas por los usuarios, utilizando datos que pueden consultarse de la API de Helldivers.

```
{
  "userID": "usuario123",
  "armor": "SC-34 Infiltrator",
  "booster": "Hellpod Space Optimization",
  "enemyType": "Terminids",
  "primaryWeapon": "Railgun",
  "secondaryWeapon": "Ultimatum",
  "stratagems": [
    "M-105 Stalwart",
    "Orbital Airburst Strike",
    "Eagle Strafing Run",
```

```
    "LIFT-850 Jump Pack"
  ],
  "like": 0,
  "dislike": 0,
  "time": "2025-04-30T15:55:05Z"
}
```

### Colección: votos

Almacena los votos individuales de cada usuario para evitar votos múltiples por build.

```
{
  "userID": "",
  "buildID": "build789xyz",
  "type": "like"
}
```

## Relaciones y Referencias

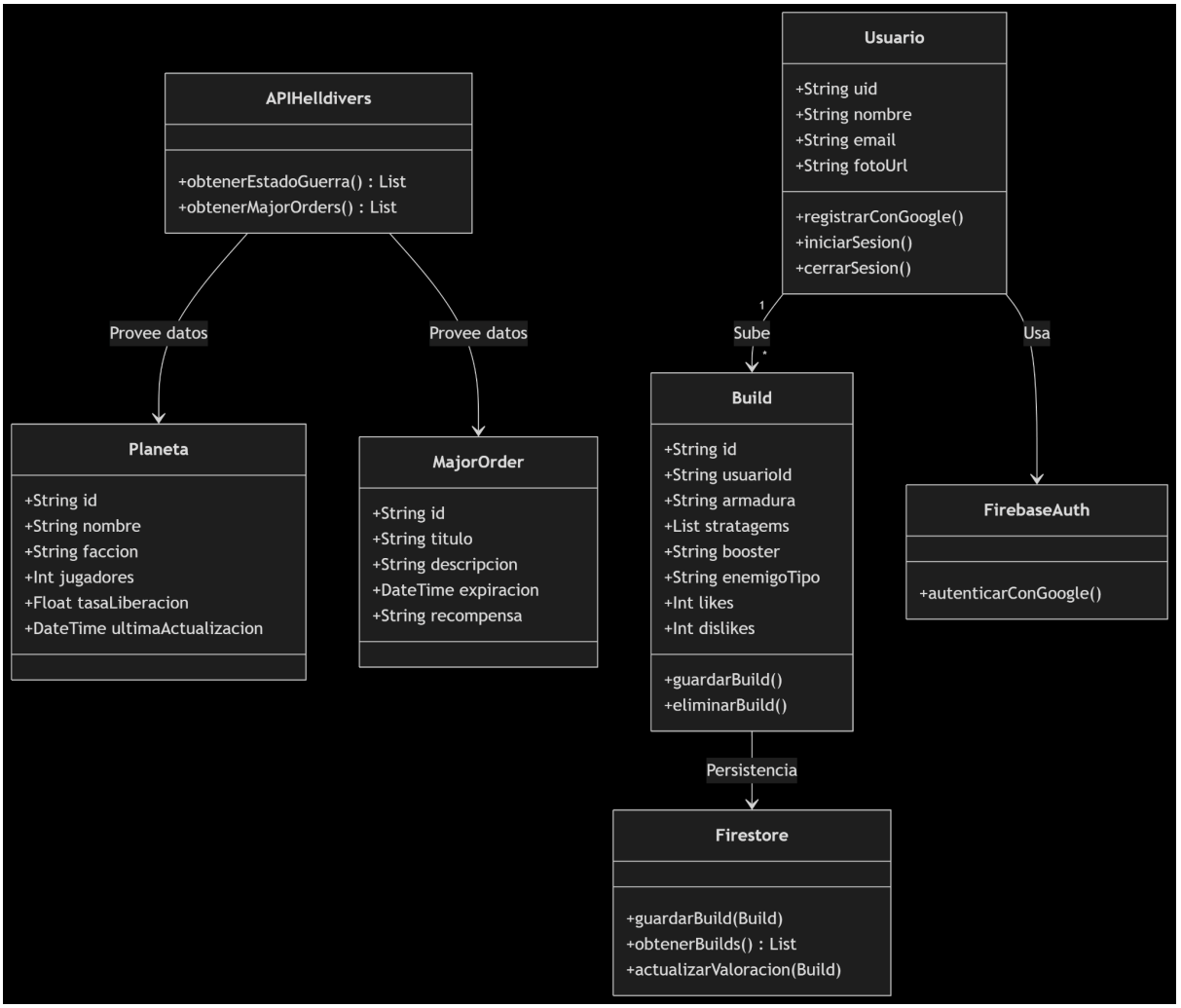
- Cada documento en builds guarda el ID del usuario que la creó (userID), referenciando a un documento en usuarios.
- Cada documento en votos guarda:
  - userID: el usuario que ha votado.
  - buildID: la build que ha sido votada.
  - type: tipo de voto (like o dislike).

## Justificación del uso de Firebase Firestore:

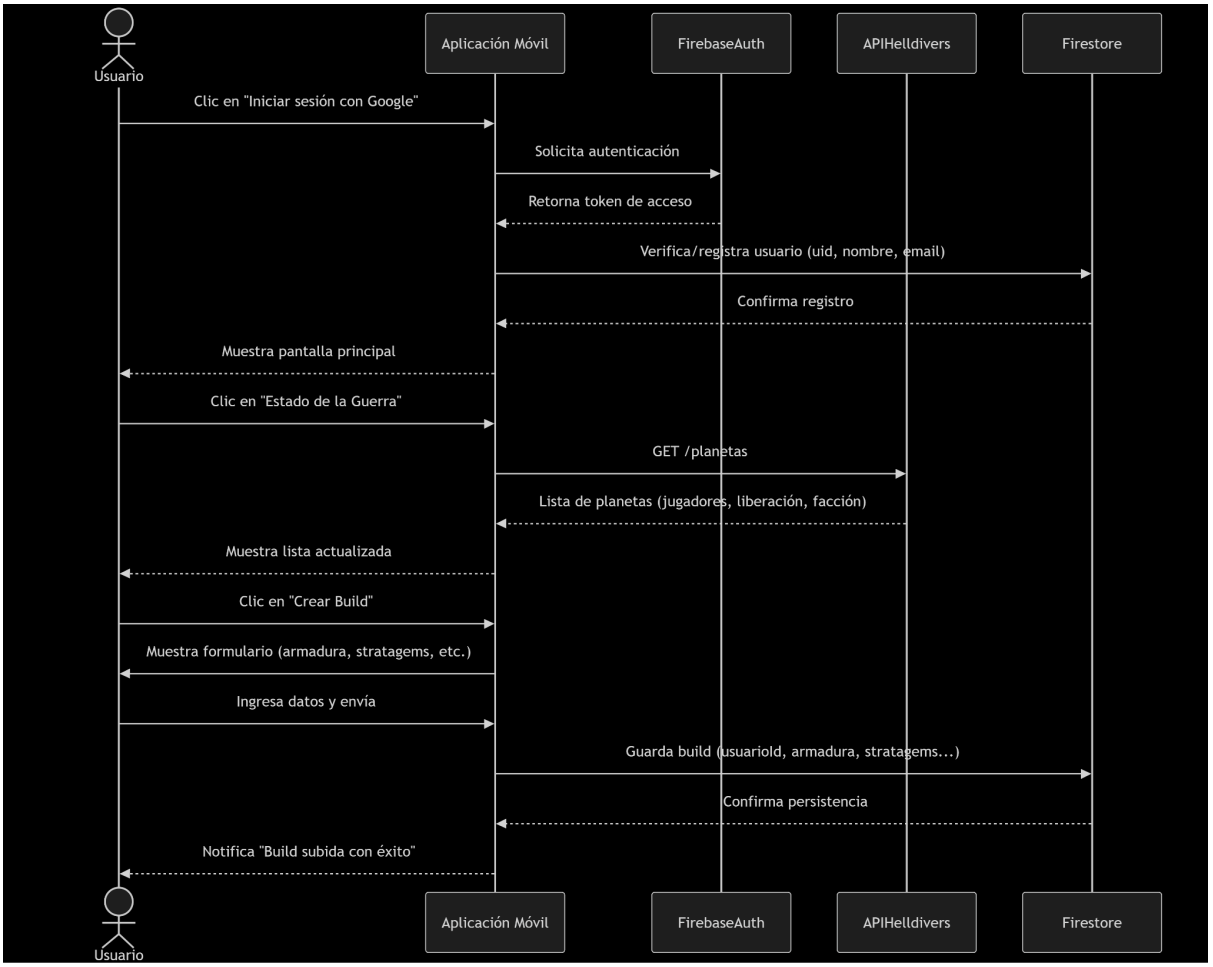
Se ha elegido Firebase Firestore como base de datos para esta aplicación por su estructura flexible, ideal para trabajar con colecciones y documentos como *builds* o usuarios, sin necesidad de esquemas rígidos como en MySQL. Firestore permite trabajar en tiempo real, mostrando cambios como votos o nuevas builds al instante en la app.

Además, se integra fácilmente con Firebase Authentication, lo que simplifica el manejo de usuarios y sus acciones dentro del sistema (por ejemplo, quién sube o vota una build). También es escalable y gestionado en la nube, lo que evita preocuparse por mantenimiento o servidores. Por estas razones, es una solución adecuada para una app social y colaborativa como esta.

## 4.2. Diagrama de clases.

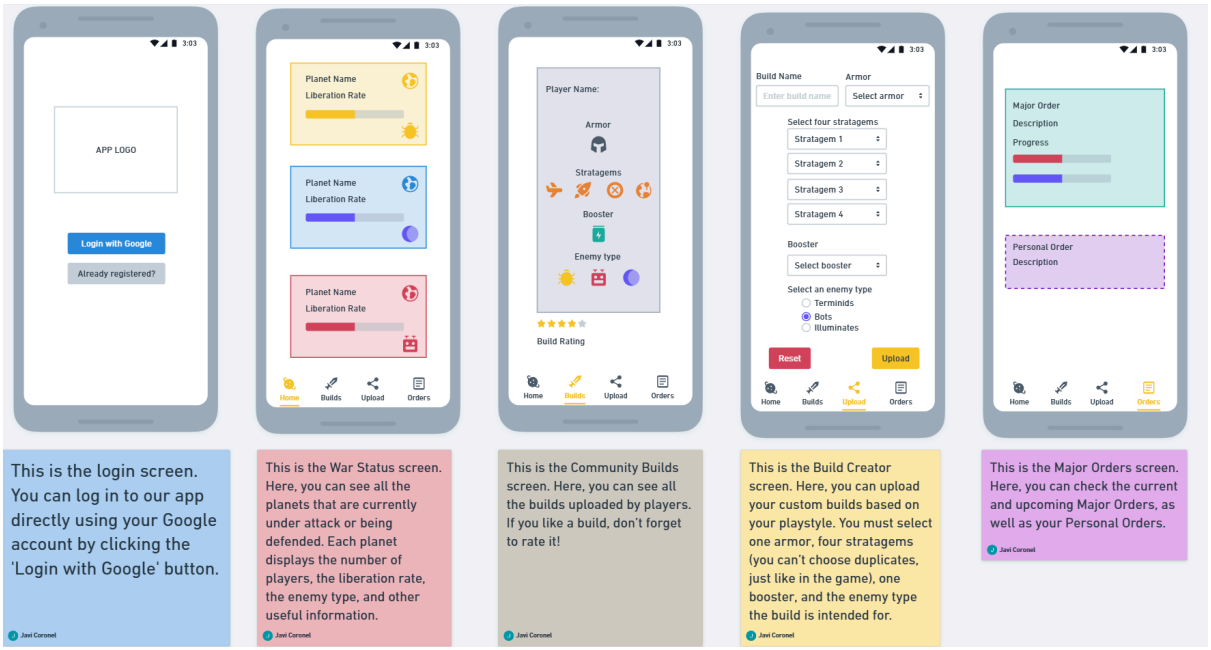


## 4.3. Diagrama de secuencia.




## 4.4. Diagrama de casos de uso.

## 4.5. Mockup.




Para más claridad visitar el enlace:
https://whimsical.com/helldivers-buildhub-RYNohu5EvfwqDBurAHWgeB