TAANIO

JXIX

MANUAL TÉCNICO

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA
Departamento de Informática

PROYECTO

TAANID

Manual Técnico

Autor: Mario López Morales
Curso Académico: 2024/2025

ÍNDICE

1 Introducción

TAANID es una aplicación móvil desarrollada como parte del Proyecto Final del Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma (DAM). Su objetivo principal es ofrecer una plataforma intuitiva y funcional para la gestión y visualización de contenido cinematográfico, permitiendo a los usuarios interactuar con una base de datos de películas de forma dinámica y personalizada.

La aplicación está diseñada para dispositivos Android y ha sido desarrollada utilizando el lenguaje de programación **Java**, junto con servicios en la nube como **Firebase** y **Cloudinary**, que permiten una gestión eficiente de datos y archivos multimedia. TAANID no solo permite a los usuarios explorar películas, sino también interactuar con ellas mediante comentarios, puntuaciones y favoritos, fomentando así una experiencia más participativa.

Funcionalidades principales de TAANID:

- **Visualización de películas**: Los usuarios pueden acceder a una lista de películas con sus respectivas imágenes, sinopsis (en varios idiomas), tráilers y enlaces de reproducción.
- **Sistema de comentarios y puntuaciones**: Cada usuario puede dejar su opinión sobre una película, puntuándola con estrellas y escribiendo un comentario. También puede editar o eliminar sus propios comentarios.
- **Favoritos**: Los usuarios registrados pueden marcar películas como favoritas para acceder a ellas rápidamente desde su perfil.
- **Gestión de perfil**: Los usuarios pueden editar su nombre, fecha de nacimiento y avatar. La aplicación permite subir imágenes personalizadas como avatar mediante Cloudinary.
- **Soporte multilingüe**: TAANID detecta el idioma del dispositivo y muestra la sinopsis de las películas en español o inglés, según disponibilidad.
- **Acceso como invitado**: Se permite el acceso anónimo a la aplicación, con funcionalidades limitadas (sin posibilidad de comentar, puntuar, editar tu perfil, ver películas o marcar favoritos).
- **Gestión de contenido por administradores**:
  - Subida de nuevas películas con imagen, sinopsis en varios idiomas, tráiler y enlace de reproducción.
  - Edición y eliminación de películas existentes.
  - Moderación de comentarios (eliminación de comentarios de otros usuarios).

TAANID ha sido diseñada siguiendo principios de modularidad y escalabilidad, utilizando componentes como **Fragments**, **RecyclerView**, y **WebView**, lo que

permite una navegación fluida y una experiencia de usuario moderna. La arquitectura del proyecto se basa en el patrón **Modelo-Vista-Controlador (MVC)**, lo que facilita la organización del código y su mantenimiento.
Este manual técnico tiene como finalidad describir en detalle el desarrollo de la aplicación, incluyendo los objetivos, tecnologías utilizadas, análisis y diseño, proceso de desarrollo, pruebas realizadas, despliegue y posibles mejoras futuras. También se incluye una bibliografía con los recursos consultados durante el desarrollo del proyecto.

2 Objetivos

El objetivo general del proyecto TAANID es desarrollar una aplicación móvil para dispositivos Android que permita a los usuarios explorar, valorar y gestionar anime de forma sencilla, intuitiva y personalizada. La aplicación está orientada tanto a usuarios finales como a administradores, ofreciendo funcionalidades diferenciadas según el tipo de usuario.

Objetivos generales:

- Diseñar y desarrollar una aplicación Android funcional y atractiva que permita la visualización de películas, y en un futuro, la serie.
- Implementar un sistema de autenticación de usuarios que permita el acceso mediante correo electrónico, así como el uso de la aplicación en modo invitado.
- Integrar una base de datos en la nube (Firebase Firestore) para almacenar y gestionar la información de usuarios, películas, comentarios y favoritos.
- Permitir la gestión de contenido por parte de administradores, incluyendo la subida, edición y eliminación de películas, también pueden gestionar eliminando comentarios.
- Ofrecer una experiencia de usuario personalizada mediante la edición del perfil, selección de avatar y gestión de favoritos.
- Garantizar la compatibilidad multilingüe (español e inglés) para mejorar la accesibilidad de la aplicación.

Objetivos específicos:

- **Visualización de películas**: Mostrar una lista dinámica de películas con imagen, título, sinopsis, tráiler y enlace de reproducción.
- **Sistema de comentarios y puntuaciones**: Permitir a los usuarios registrados comentar y puntuar películas, así como editar sus propios comentarios.
- **Gestión de favoritos**: Implementar una funcionalidad que permita a los usuarios marcar películas como favoritas y filtrarlas desde la lista principal.
- **Gestión de perfil de usuario**: Permitir a los usuarios modificar su nombre, fecha de nacimiento y avatar, con almacenamiento en la nube.
- **Acceso como invitado**: Permitir el uso limitado de la aplicación sin necesidad de registro, restringiendo ciertas funcionalidades como comentar o puntuar.
- **Panel de administración**: Habilitar funcionalidades exclusivas para administradores, como la subida de nuevas películas, edición de datos y eliminación de películas/comentarios.
- **Carga y almacenamiento de imágenes**: Integrar Cloudinary para la gestión eficiente de imágenes (avatares y carátulas de películas).
- **Interfaz adaptable**: Diseñar una interfaz moderna y responsive que se adapte a diferentes tamaños de pantalla y orientaciones.
- **Filtrado y búsqueda**: Implementar un sistema de búsqueda y filtrado de películas por título o sinopsis.
- **Persistencia de datos en la nube**: Asegurar que todos los datos se almacenen y incronicen correctamente mediante Firebase.

3 Tecnologías involucradas

En el desarrollo de la aplicación **TAANID** se han utilizado diversas tecnologías, herramientas y servicios que permiten implementar tanto la lógica de negocio como la interfaz de usuario, el almacenamiento de datos y la gestión de archivos multimedia. A continuación, se detallan las principales tecnologías empleadas, así como un breve marco teórico de aquellas que no se han trabajado en profundidad durante el ciclo formativo.

Lenguaje de programación

- **Java**: Lenguaje principal utilizado para el desarrollo de la aplicación Android. Java es un lenguaje orientado a objetos ampliamente utilizado en el desarrollo de aplicaciones móviles, especialmente en el entorno Android. Su robustez, portabilidad y compatibilidad con el SDK de Android lo convierten en una elección adecuada para este tipo de proyectos.

Entorno de desarrollo

- **Android Studio**: Entorno de desarrollo integrado (IDE) oficial para Android. Proporciona herramientas para el diseño de interfaces, depuración, emulación y gestión de dependencias mediante Gradle.

Servicios en la nube

- **Firebase** (Google):
  - **Firebase Authentication**: Servicio que permite gestionar el registro, inicio de sesión y autenticación de usuarios, incluyendo sesiones anónimas.
  - **Firebase Firestore**: Base de datos NoSQL en tiempo real utilizada para almacenar información de usuarios, películas, comentarios y favoritos.

- **Cloudinary**:
  - Plataforma de gestión de imágenes en la nube. Se ha utilizado para subir, almacenar y recuperar imágenes (avatares de usuario y carátulas de películas). Cloudinary permite optimizar y transformar imágenes de forma dinámica mediante URLs.

Librerías externas

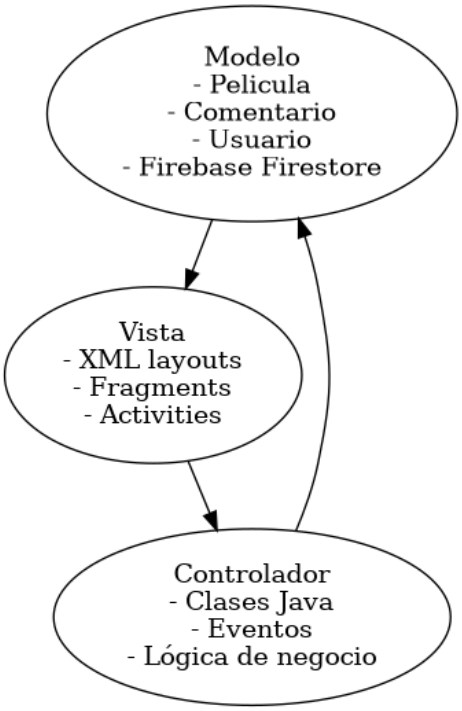
- **Glide**:
  - Librería de código abierto para la carga y visualización eficiente de imágenes en Android. Permite mostrar imágenes desde URLs, aplicar transformaciones y gestionar la caché de forma automática.

Componentes de Android utilizados

- **Fragments**: Componentes reutilizables que permiten dividir la interfaz en secciones modulares. Se han utilizado para mostrar listas, detalles y formularios.
- **RecyclerView**: Componente para mostrar listas de elementos de forma eficiente, con soporte para scroll y adaptadores personalizados.
- **WebView**: Componente que permite mostrar contenido web dentro de la aplicación. Se ha utilizado para reproducir vídeos de YouTube embebidos.
- **AlertDialog y PopupWindow**: Utilizados para mostrar diálogos de confirmación, edición y menús contextuales.
- **RatingBar**: Componente visual que permite a los usuarios puntuar películas mediante estrellas.

Diseño y arquitectura

- **Patrón MVC (Modelo-Vista-Controlador)**:
  - Se ha seguido este patrón para separar la lógica de negocio (modelo), la interfaz de usuario (vista) y la gestión de eventos (controlador), facilitando así el mantenimiento y la escalabilidad del proyecto.



Internacionalización

- **Soporte multilingüe (i18n)**:
  - La aplicación detecta el idioma del dispositivo y muestra la sinopsis de las películas en español o inglés, según disponibilidad. Esto se ha implementado mediante el uso de Map<String, String> en los datos de sinopsis y la API de localización de Java.

Diseño de la base de datos

La base de datos de TAANID está implementada con **Firebase Firestore**, una base de datos NoSQL orientada a documentos. Su estructura jerárquica permite organizar los datos en colecciones y subcolecciones, lo que facilita la escalabilidad y el acceso eficiente a la información.

Colección: DatosUsuario

Cada documento representa un usuario registrado. Los campos principales son:

- Nombre: nombre del usuario.
- FechaNacimiento: fecha de nacimiento (tipo timestamp).
- Avatar: URL del avatar del usuario.
- admin: booleano que indica si el usuario es administrador.

Subcolección: Favoritos Contiene las películas que el usuario ha marcado como favoritas. Cada documento incluye:

- id, imagenUrl, nombrePeli, sinopsis (mapa con claves es e en), urlPelicula, urlTrailer.

Colección: Peliculas

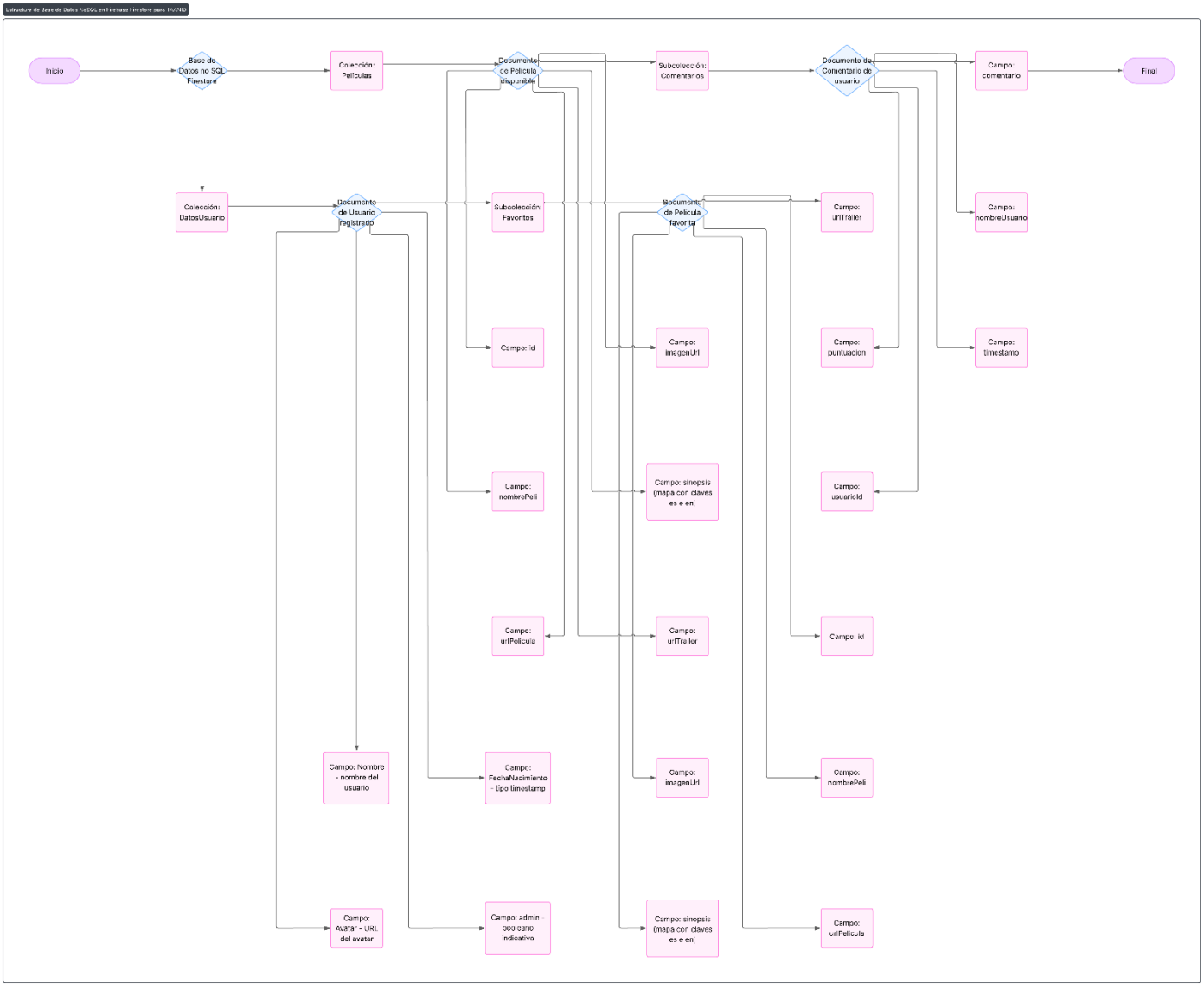
Cada documento representa una película disponible en la app. Contiene los mismos campos que los favoritos:

- id, imagenUrl, nombrePeli, sinopsis (mapa con claves es e en), urlPelicula, urlTrailer.

Subcolección: Comentarios Incluye los comentarios de los usuarios sobre cada película. Cada documento contiene:

- comentario, nombreUsuario, puntuacion, timestamp, usuarioId.

Este diseño permite una gestión clara y eficiente de los datos, manteniendo la relación entre usuarios, películas y sus interacciones.



4 Proceso de desarrollo

4.1 Análisis

Antes de comenzar con la implementación de la aplicación **TAANID**, se realizó un análisis funcional y estructural para definir los requisitos del sistema, los actores involucrados y las relaciones entre los distintos componentes. Este análisis sirvió como base para el diseño y desarrollo posterior.

Actores del sistema

Se identificaron tres tipos de usuarios que interactúan con la aplicación:

- **Usuario registrado**: puede acceder a todas las funcionalidades principales, como comentar, puntuar, marcar películas como favoritas y editar su perfil.
- **Administrador**: tiene los mismos permisos que un usuario registrado, pero además puede subir, editar y eliminar películas, así como moderar comentarios.
- **Usuario invitado**: puede navegar por la aplicación y visualizar trailers, pero no puede realizar acciones que requieran autenticación, como comentar o puntuar.

Casos de uso

A partir de los actores definidos, se identificaron los siguientes casos de uso principales:

- Registrarse / Iniciar sesión / Acceder como invitado
- Ver lista de películas
- Ver detalles de una película
- Comentar y puntuar una película
- Marcar o desmarcar como favorita
- Editar perfil de usuario
- Subir, editar o eliminar películas (solo administrador)
- Eliminar comentarios (solo administrador)

Estos casos de uso permiten cubrir todas las funcionalidades esenciales de la aplicación, tanto para usuarios comunes como para administradores.

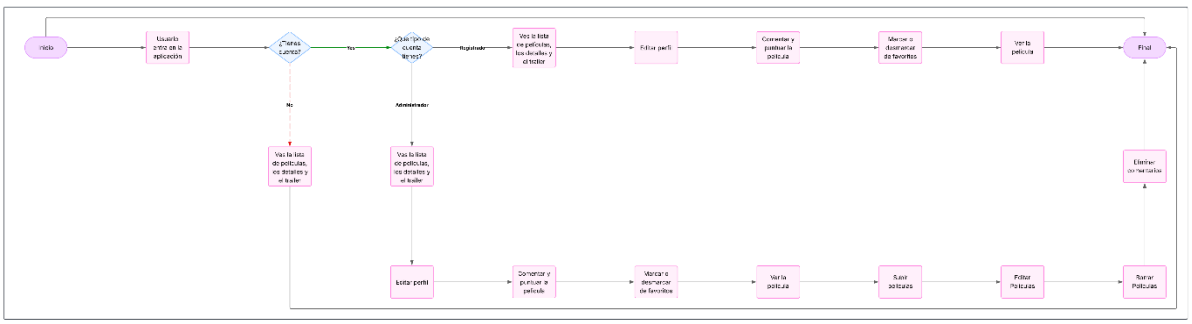


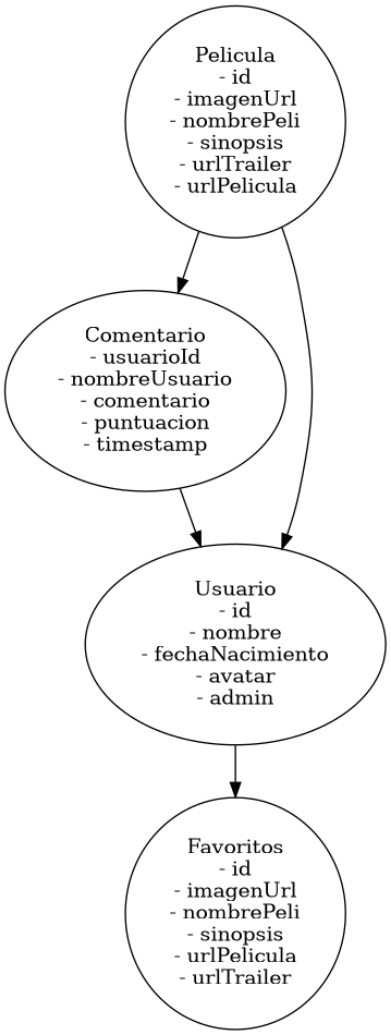
Diagrama de clases

Durante el análisis estructural, se definieron las clases principales que forman parte del modelo de datos de la aplicación. Estas clases representan los elementos clave del sistema y sus relaciones:

- **Pelicula**: contiene los datos de cada película (ID, título, sinopsis, imagen, enlaces).
- **Comentario**: representa los comentarios realizados por los usuarios.
- **Usuario**: gestionado a través de Firebase Authentication y Firestore.
- **RecyclerAdapter**: adaptador para mostrar películas en listas.
- **Fragmentos**: componentes de interfaz que gestionan la navegación y visualización de contenido.

Las relaciones entre estas clases incluyen:

- Un usuario puede escribir varios comentarios.
- Una película puede tener múltiples comentarios.
- Un usuario puede marcar varias películas como favoritas.
- Los fragmentos muestran películas, comentarios y permiten editar el perfil del usuario.

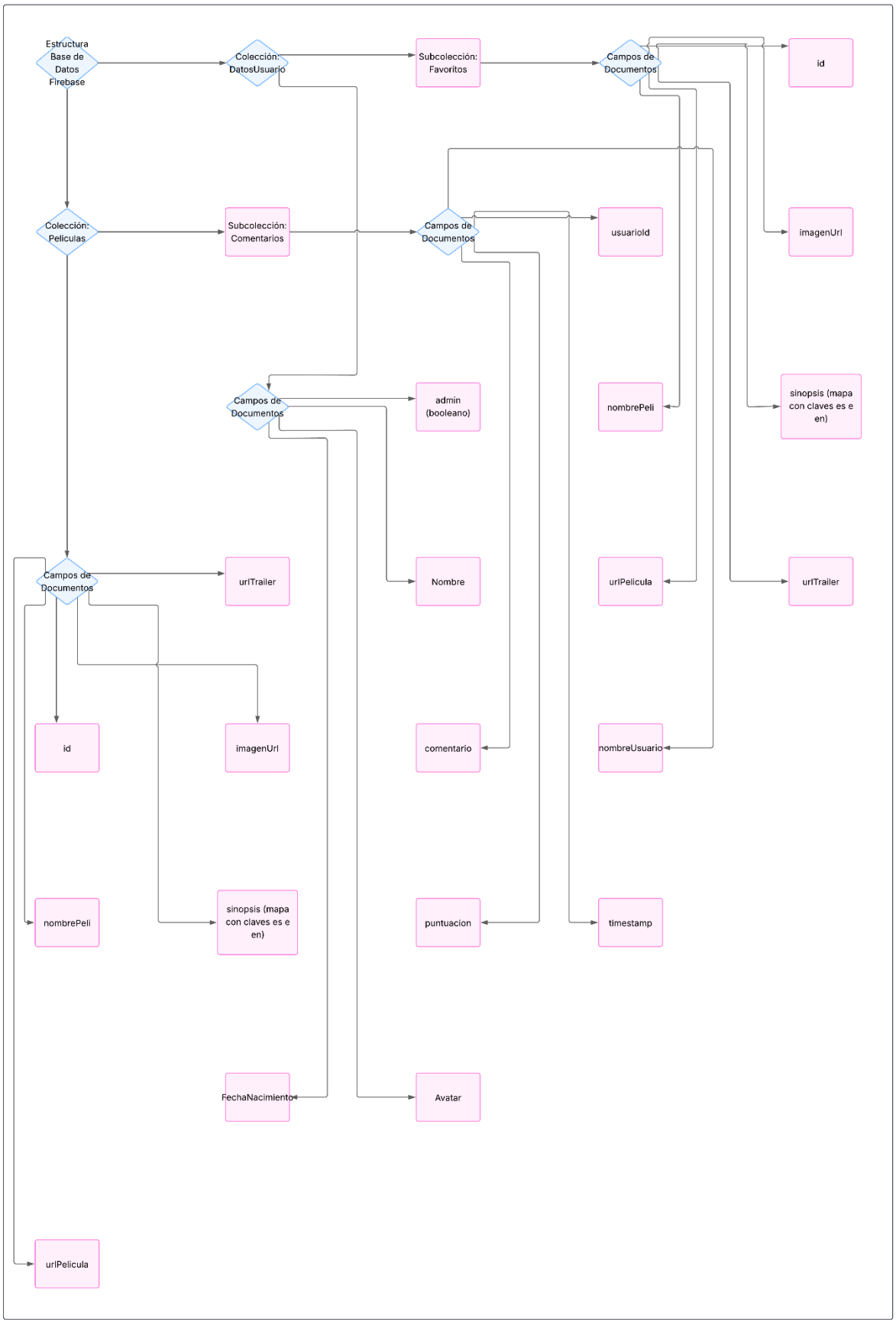


Diseño de la base de datos (estructura Firestore)

La base de datos de TAANID está implementada con **Firebase Firestore**, una base de datos NoSQL orientada a documentos. Su estructura jerárquica permite organizar los datos en colecciones y subcolecciones, lo que facilita la escalabilidad y el acceso eficiente a la información.

- **Colección DatosUsuario**: contiene documentos con los datos de cada usuario.
  - **Campos**: Nombre, FechaNacimiento, Avatar, admin (booleano).
  - **Subcolección Favoritos**: contiene documentos con los datos de las películas marcadas como favoritas por el usuario.
    - **Campos**: id, imagenUrl, nombrePeli, sinopsis (mapa con claves es e en), urlPelicula, urlTrailer.
- **Colección Peliculas**: contiene los datos de cada película disponible en la aplicación.
  - **Campos**: id, imagenUrl, nombrePeli, sinopsis (mapa con claves es e en), urlPelicula, urlTrailer.
  - **Subcolección Comentarios**: contiene los comentarios realizados por los usuarios sobre cada película.
    - **Campos**: comentario, nombreUsuario, puntuacion, timestamp, u suarioId.

Este diseño permite una gestión clara y eficiente de los datos, manteniendo la relación entre usuarios, películas y sus interacciones.


Estructura Base de Datos Firebase
Colección: DatosUsuario
Subcolección: Favoritos
Campos de Documentos
id
Colección: Películas
Subcolección: Comentarios
Campos de Documentos
usuarioId
imagenUrl
Campos de Documentos
admin (booleano)
nombrePeli
sinopsis (mapa con claves es e en)
Campos de Documentos
urlTrailer
Nombre
urlPelicula
urlTrailer
id
imagenUrl
comentario
nombreUsuario
nombrePeli
sinopsis (mapa con claves es e en)
puntuacion
timestamp
FechaNacimiento
Avatar
urlPelicula

4.2 Desarrollo

El desarrollo de la aplicación TAANID se ha realizado de forma modular, siguiendo una estructura clara basada en el patrón Modelo-Vista-Controlador (MVC). A continuación, se describen los principales aspectos del proceso de desarrollo, organizados por funcionalidades clave y componentes de la aplicación.

4.2.1 Interfaz de usuario (UI)

La interfaz de usuario se ha diseñado utilizando componentes nativos de Android, con un enfoque en la usabilidad y la experiencia del usuario. Se han utilizado Fragments para dividir la aplicación en secciones reutilizables, como:

- **Lista de películas** (Lista): muestra todas las películas disponibles mediante un RecyclerView.
- **Detalles de película** (Detalles): muestra la información completa de una película, incluyendo sinopsis, tráiler, enlace de reproducción, comentarios y puntuación.
- **Formulario de registro** (FragmentoRegistro): permite a nuevos usuarios registrarse, incluyendo la subida de avatar.
- **Edición de perfil** (EditarPerfilActivity): permite modificar nombre, fecha de nacimiento y avatar.
- **Subida y edición de películas** (SubirPeliculaActivity, EditarPeliculaActivity): accesibles solo para administradores.

Se ha utilizado Material Design para mantener una estética moderna y coherente, y Glide para la carga eficiente de imágenes.

4.2.2 Lógica de negocio

La lógica de negocio se ha implementado principalmente en Java, separando claramente las responsabilidades:

- **Adaptadores** (RecyclerAdapter): gestionan la visualización de listas de películas y la interacción con los elementos (favoritos, clics).
- **Controladores de eventos**: gestionan las acciones del usuario, como enviar comentarios, marcar favoritos o editar datos.
- **Validaciones**: se han implementado validaciones para evitar campos vacíos, duplicados y errores de formato (por ejemplo, en fechas).

4.2.3 Gestión de usuarios

La autenticación se ha implementado con **Firebase Authentication**, permitiendo:

- Registro con correo electrónico y contraseña.
- Inicio de sesión.
- Acceso como invitado (modo anónimo).
- Almacenamiento de datos adicionales del usuario en Firestore (DatosUsuario), como nombre, avatar, fecha de nacimiento y rol (admin).

4.2.4 Gestión de películas

Las películas se almacenan en la colección Peliculas de Firestore. Cada documento contiene:

- Título, sinopsis (en varios idiomas), imagen, tráiler y enlace de reproducción.

Los administradores pueden:

- Subir nuevas películas.
- Editar películas existentes.
- Eliminar películas.

Se ha implementado un sistema de validación para evitar duplicados por título o enlaces.

4.2.5 Comentarios y puntuaciones

Cada película tiene una subcolección Comentarios, donde los usuarios registrados pueden:

- Escribir comentarios.
- Puntuar con estrellas.
- Editar o eliminar sus propios comentarios.
- Los administradores pueden eliminar cualquier comentario.

4.2.6 Favoritos

Los usuarios registrados pueden marcar películas como favoritas. Estas se almacenan en la subcolección Favoritos dentro de su documento en DatosUsuario. Desde la interfaz, se puede alternar entre ver todas las películas o solo las favoritas.

4.2.7 Reproducción de contenido

Para ver una película, se utiliza un WebView que embebe el vídeo desde YouTube. Solo los usuarios registrados pueden acceder a esta funcionalidad, como medida de control.

4.2.8 Internacionalización

La aplicación detecta el idioma del dispositivo y muestra la aplicación, incluyendo la sinopsis, de las películas en español o inglés, según disponibilidad. Esto se gestiona mediante un Map<String, String> en el campo sinopsis de cada película, aquí dejo algunos ejemplos de los archivos strings que se usan en la aplicación.

Show All Keys | Show All Locales

Key	Resource Folder	Untranslatable	Default Value	English (en)
app_name	app/src/main/res	☐	TAANID	TAANID
usuario	app/src/main/res	☐	Email	Email
contraseña	app/src/main/res	☐	Contraseña	Password
botonLogin	app/src/main/res	☐	Iniciar Sesión	Login
botonInvitado	app/src/main/res	☐	Iniciar como invitado	Start as a guest
botonRegistro	app/src/main/res	☐	Registrarse	Register
nombre	app/src/main/res	☐	Nombre	Name
fechaNac	app/src/main/res	☐	Fecha de Nacimiento	Birthdate
botonAvatar	app/src/main/res	☐	Elegir Avatar	Choose Avatar
leyenda	app/src/main/res	☐	Los campos que llevan * son o	Fields with * are required
imagen_pelicula	app/src/main/res	☐	Imagen de la película	Picture of the movie
avatar_usuario	app/src/main/res	☐	Avatar del usuario	User avatar
btnGuardar	app/src/main/res	☐	Guardar cambios	Save changes
CambiarAvatar	app/src/main/res	☐	Cambiar avatar	Change avatar
esAdmin	app/src/main/res	☐	¿Es administrador?	Are you an administrator?
btnTrailer	app/src/main/res	☐	Ver Tráiler	See Trailer
btnPelicula	app/src/main/res	☐	Ver Película	See film
comentarios	app/src/main/res	☐	Comentarios	Comments
escribirComent	app/src/main/res	☐	Escribe un comentario...	Write a comment ...
btnEnviar	app/src/main/res	☐	Enviar	Send
eliminarComent	app/src/main/res	☐	Eliminar comentario	Delete comment
advertenciaComent	app/src/main/res	☐	¿Estás seguro de que deseas e	Are you sure you want to delet
btnCancelar	app/src/main/res	☐	Cancelar	Cancel
btnSi	app/src/main/res	☐	Sí	Yes
volverInicio	app/src/main/res	☐	Volver al inicio	Back to home
advertenciaInicio	app/src/main/res	☐	¿Seguro que quieres cerrar se	Are you sure you want to log o
btnEditarperfil	app/src/main/res	☐	Editar perfil	Edit profile
btnCerrarSesion	app/src/main/res	☐	Cerrar sesión	Close session
nombrePeli	app/src/main/res	☐	Nombre de la película	Film name
sinopsis	app/src/main/res	☐	Sinopsis	Sinopsis
urlTrailer	app/src/main/res	☐	URL del tráiler	Trailer URL
urlPeli	app/src/main/res	☐	URL de la película	Film URL
btnSeleccionarImagen	app/src/main/res	☐	Seleccionar imagen	Select image
btnSubirPeli	app/src/main/res	☐	Subir película	Upload film
btnEliminar	app/src/main/res	☐	Eliminar película	Delete film
btnEditarPeli	app/src/main/res	☐	Editar película	Edit film
busquedaPeli	app/src/main/res	☐	Buscar por título o sinopsis	Search by title or synopsis
btnFavorito	app/src/main/res	☐	Mostrar Favoritos	Show Favorites
completarImagen	app/src/main/res	☐	Por favor, completa todos los c	Please complete all fields and s

4.3 Pruebas realizadas

Detallar las pruebas que se han realizado indicando:
• Componente (clase, método,…) al que se le han realizado.
• Juego de datos de entrada.
• Resultado de salida.
• Herramientas de pruebas usadas.

Componente	Tipo de prueba	Datos de entrada	Resultado esperado	Herramientas utilizadas
SubirPeliculaActivity	Instrumentada	Botón pulsado sin completar campos	Se muestra un Toast indicando que falta la imagen	Espresso, ActivityScenario
RecyclerAdapter	Unitaria	Filtro con "Película 1"	Se muestra solo "Película 1"	JUnit, Mockito
Pelicula	Unitaria	Setters y getters con valores de prueba	Los valores se recuperan correctamente	JUnit
MyApp / CloudinaryManager	Instrumentada	Inicialización del contexto	MediaManager no es nulo	AndroidJUnit4
MainActivity	Instrumentada	Lanzamiento de actividad	Todos los elementos de UI están visibles	Espresso
Lista (Fragmento)	Instrumentada	Lanzamiento del fragmento	RecyclerView y botones visibles	FragmentScenario
InicioSi	Instrumentada	Lanzamiento de actividad	Botones y avatar visibles	ActivityScenario
FragmentoRegistro	Instrumentada	Lanzamiento del fragmento	Todos los campos de registro visibles	FragmentScenario
FragmentActivity	Instrumentada	Lanzamiento de actividad	Fragmento cargado correctamente	ActivityScenario
EditarPerfilActivity	Instrumentada	Lanzamiento con usuario anónimo	Campos de edición visibles	FirebaseAuth, ActivityScenario

EditarPeliculaActivity	Instrumentada	Intent con objeto Pelicula	Campos rellenados con datos de la película	ActivityScenario, Glide
Detalles (Fragmento)	Instrumentada	Objeto Pelicula y esAdmin=true	Vistas cargadas y datos mostrados correctamente	FragmentScenario, Glide
Comentario	Unitaria	Setters y getters	Datos correctamente almacenados y recuperados	JUnit

5 Proceso de Despliegue

El despliegue de la aplicación TAANID está orientado a dispositivos Android y se puede realizar tanto en entornos de desarrollo (emulador) como en dispositivos físicos. A continuación, se detallan los pasos necesarios para ejecutar correctamente la aplicación, así como los requisitos de software y configuración.

5.1 Requisitos de software

Para poder compilar, ejecutar y desplegar la aplicación, se necesita el siguiente entorno:

- **Android Studio** (versión recomendada: 2022.3 o superior)
- **Java Development Kit (JDK)** versión 11 o superior
- **Gradle** (integrado en Android Studio)
- **Conexión a Internet** (para acceder a Firebase y Cloudinary)
- **Cuenta de Firebase** con un proyecto configurado
- **Archivo google-services.json** descargado desde Firebase Console e incluido en la carpeta app/ del proyecto

5.2 Dependencias necesarias

El archivo build.gradle incluye las siguientes dependencias clave:

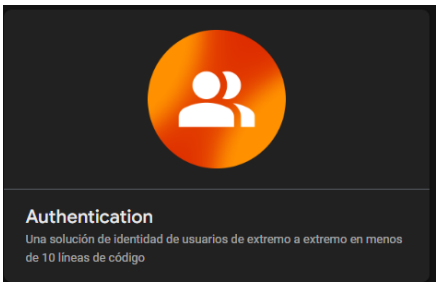
- Firebase Authentication
- Firebase Firestore
- Firebase UI
- Cloudinary SDK para Android
- Glide (para carga de imágenes)
- Material Components

```
dependencies {
    implementation platform('com.google.firebase:firebase-bom:33.13.0')
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-firestore'
    implementation 'com.google.firebase:firebase-database'
    implementation 'com.google.firebase:firebase-storage'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.11.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.navigation:navigation-fragment:2.5.3'
    implementation 'androidx.media3:media3-exoplayer:1.3.1'
    implementation 'androidx.media3:media3-ui:1.3.1'
    implementation 'com.github.bumptech.glide:glide:4.15.1'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.15.1'
    implementation 'com.cloudinary:cloudinary-android:2.3.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Estas dependencias se descargan automáticamente al sincronizar el proyecto en Android Studio.

5.3 Configuración de Firebase

1. Crear un proyecto en Firebase Console.
2. Activar los siguientes servicios:
    - Authentication (correo y contraseña, acceso anónimo)



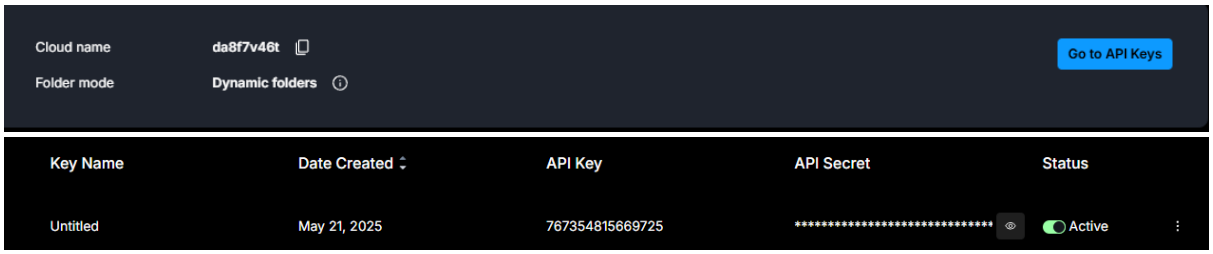
    - Firestore Database

3. Descargar el archivo google-services.json y colocarlo en la carpeta app/.
4. Configurar las reglas de seguridad de Firestore según el entorno (desarrollo o producción).

5.4 Configuración de Cloudinary

1. Crear una cuenta en Cloudinary.
2. Obtener las credenciales:
    - cloud_name
    - api_key
    - api_secret

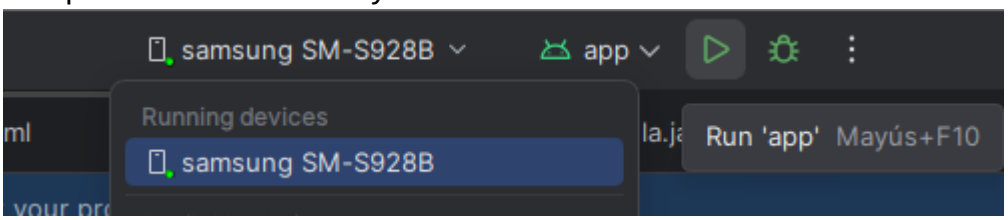
Obtienes el cloud_name entrando a https://console.cloudinary.com y dándole a Home, cuando ya tienes el cloud_name le das al botón "Go to API Keys" y obtienes api_key y api_secret



3. Configurar estos valores en la clase CloudinaryManager.java del proyecto.

5.5 Despliegue en dispositivo físico o emulador

1. Conectar un dispositivo Android con depuración USB activada o iniciar un emulador desde Android Studio.
2. Ejecutar el proyecto desde Android Studio con el botón "Run".
3. La aplicación se instalará y abrirá automáticamente.



5.6 Hosting y distribución (opcional)

Si se desea distribuir la aplicación:

- Generar un archivo .apk o .aab desde Android Studio (Build > Build Bundle(s) / APK(s)).
- Subirlo a Google Play Console (requiere cuenta de desarrollador).
- También se puede compartir directamente el .apk para instalación manual.

6 Propuesta de mejora o trabajos futuros

Aunque la aplicación TAANID cumple con los objetivos funcionales establecidos, existen diversas oportunidades de mejora y expansión que podrían implementarse en futuras versiones para enriquecer la experiencia del usuario, aumentar la sostenibilidad del proyecto y ampliar su alcance. A continuación, se detallan algunas propuestas:

6.1 Monetización y sostenibilidad

- **Integración de anuncios**: Incorporar anuncios mediante plataformas como Google AdMob permitiría generar ingresos pasivos.
- **Financiación de licencias**: Los ingresos obtenidos podrían destinarse a la adquisición de licencias legales para la distribución de películas y series, mejorando así la legalidad y calidad del contenido.
- **Versión de pago**: Se podría ofrecer una versión premium sin anuncios o con funcionalidades exclusivas, como contenido anticipado o filtros avanzados.

6.2 Seguridad y control de acceso

- **Restricción de privilegios de administrador**: Actualmente, cualquier usuario puede marcarse como administrador al registrarse. Se recomienda eliminar esta opción del formulario y gestionar los permisos desde la base de datos por parte del desarrollador o mediante un panel de control seguro.

6.3 Ampliación del catálogo

- **Incorporación de series**: Ampliar la base de datos para incluir no solo películas, sino también series, con sus respectivas temporadas y episodios.
- **Estructura jerárquica**: Adaptar el modelo de datos para soportar múltiples niveles (serie → temporada → episodio).

6.4 Mejora de la experiencia de usuario

- **Filtrado por categorías**: Implementar un sistema de clasificación por géneros (acción, comedia, drama, etc.) y permitir al usuario filtrar o buscar contenido por categoría.
- **Sistema de recomendaciones**: Sugerir películas o series basadas en las valoraciones del usuario o en sus favoritos.
- **Modo oscuro**: Añadir un tema oscuro para mejorar la experiencia visual en entornos con poca luz.

6.5 Escalabilidad y mantenimiento

- **Panel de administración web**: Desarrollar una interfaz web para que los administradores puedan gestionar el contenido de forma más cómoda y segura.
- **Notificaciones push**: Informar a los usuarios sobre nuevas películas, actualizaciones o recomendaciones personalizadas.

7 Bibliografía

A lo largo del desarrollo del proyecto TAANID se han consultado diversas fuentes de información, documentación oficial y recursos técnicos que han servido de apoyo para la implementación de funcionalidades, resolución de problemas y toma de decisiones técnicas. A continuación, se enumeran las principales referencias utilizadas:

Documentación oficial

- Firebase Documentation: https://firebase.google.com/docs
- Firebase Authentication: https://firebase.google.com/docs/auth
- Firebase Firestore: https://firebase.google.com/docs/firestore
- Cloudinary for Android: https://cloudinary.com/documentation/android_integration
- Glide Image Loading Library: https://github.com/bumptech/glide
- Android Developers (Material Design, Fragments, RecyclerView, WebView, etc.): https://developer.android.com/

Recursos adicionales

- Stack Overflow (consultas técnicas específicas): https://stackoverflow.com
- GitHub (repositorios de ejemplo y librerías): https://github.com
- YouTube (tutoriales sobre Firebase y Android Studio)
- Cursos y apuntes del Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma (DAM)