**MusicApp**

**Descripción del Proyecto**

Aplicación nativa para Android desarrollada en **Kotlin** que funciona como un cliente de sincronización y descarga de música. Se conecta a una API REST para obtener un catálogo de canciones, compara los datos con una base de datos local y verifica el almacenamiento físico del dispositivo para realizar descargas inteligentes de archivos .mp3 en segundo plano.

**Funcionalidades Principales**

- **Consumo de API REST:** Obtención del catálogo musical (título, artista, álbum, carátula y URL del MP3) en formato JSON.
- **Sincronización Inteligente de 3 vías:** La aplicación cruza los datos del servidor, la base de datos local (SQLite) y el almacenamiento físico del teléfono (carpeta Music). Si un archivo se elimina manualmente del dispositivo, la app lo detecta y lo marca para su próxima descarga.
- **Gestor de Descargas Nativo:** Uso de DownloadManager de Android para gestionar la descarga de archivos .mp3 en segundo plano, restringido a conexiones Wi-Fi y con notificaciones del sistema.
- **Interfaz Declarativa:** Interfaz de usuario construida 100% con **Jetpack Compose** y Material Design 3, incluyendo navegación por pestañas (BottomNavigationBar) y listas perezosas (LazyColumn) para rendimiento óptimo.
- **Dashboard Estadístico:** Pantalla de inicio con un resumen en tiempo real que contrasta el catálogo del servidor con los archivos presentes en el dispositivo y las descargas faltantes.

**Tecnologías y Arquitectura**

El proyecto fue construido aplicando principios modernos de desarrollo Android:

- **Lenguaje:** Kotlin
- **UI Toolkit:** Jetpack Compose & Navigation Compose.
- **Programación Asíncrona:** Kotlin Coroutines (Dispatchers.IO, Dispatchers.Main) para no bloquear el hilo de la interfaz durante las descargas y consultas.
- **Persistencia de Datos: Room Database** para el almacenamiento en caché de los metadatos de las canciones y estado de descarga.
- **Peticiones de Red: Retrofit2** y Gson Converter.
- **Carga de Imágenes: Coil** para la carga y caché asíncrona de las carátulas de los álbumes (AsyncImage).

**Retos Superados**

- **Manejo de Nulos e Integridad de Datos:** Adaptación del modelo de datos (Entity) para soportar respuestas asimétricas de la API utilizando nulabilidad segura en Kotlin (String?), previniendo caídas (crashes) de la aplicación.
- **Codificación de URLs y Nombres de Archivo:** Implementación de limpieza de caracteres especiales (como #) en las URLs y nombres de archivo para evitar errores de sintaxis HTTP (%23) al interactuar con el servidor de descargas.
- **Desfase de Estado (State Mismatch):** Solución de problemas de sincronización aislando el estado de Compose mediante el uso reactivo de variables State (mutableIntStateOf) combinadas con callbacks y funciones LaunchedEffect, logrando que las estadísticas en pantalla se actualicen inmediatamente tras una descarga o sincronización.

**Capturas de Pantalla**