
## Modelo de implementación

**Proyecto:**

# HaramMusic

# 1. Requisitos del ERS que SÍ se implementan en la entrega

## 1.1 Requisitos funcionales implementados

**RF-01 – Inicio de sesión**
El sistema permite que usuarios registrados inicien sesión mediante credenciales almacenadas localmente.

**RF-02 – Registro de usuarios**
La aplicación permite el registro de nuevos usuarios, validando campos obligatorios y evitando duplicados.

**RF-03 – Reproducción de música**
El usuario puede reproducir canciones almacenadas localmente en el dispositivo.
Incluye controles básicos de reproducción (play / pause).

**RF-04 – Buscar y explorar canciones**
El usuario puede buscar canciones almacenadas en la base de datos local mediante texto.

**RF-05 – Buscar y explorar canciones**
El usuario puede buscar canciones utilizando filtros por artista y álbum.

**RF-06 – Creación de playlists**
El usuario puede crear playlists propias y asociar canciones locales.

**RF-07 – Edición de playlists**
El usuario puede editar playlists existentes (añadir o eliminar canciones).

**RF-08 – Visualización de datos de canciones**
Se muestran datos básicos de las canciones (título, artista, álbum). Si los administradores han añadido información extra, también se mostrará.

**RF-09 – Gestión de contenidos (modo administrador)**
El administrador puede:

- Editar información de canciones, álbumes y artistas
- Eliminar fichas de canciones, álbumes y artistas.
- Editar reseñas de usuarios
- Eliminar reseñas de usuarios

**RF-10 – Gestión de roles**
El sistema distingue entre usuarios normales y administradores, mostrando funcionalidades distintas.

**RF-11 – Valoración de contenidos**
Los usuarios pueden valorar con estrellas para dar una puntuacion del 0.5 al 5.0 y reseñar las canciones.

## 1.2 Requisitos no funcionales implementados

**RNF-2 – Seguridad (parcial)**

- Control de acceso por roles (usuario / administrador).
- Autenticación local.

**RNF-5 – Mantenibilidad**

- Código modular.
- Separación entre lógica, datos y UI.

**RNF-6 – Portabilidad**

- Aplicación Android desarrollada en Kotlin.
- Compatible con versiones modernas de Android.

# 2. Requisitos del ERS que no se implementan o se han modificado

## 2.1 Requisitos no implementados

**RC-IS-1 – Integración con la API de Spotify**

- **Motivo***:* la API de Spotify dejó de estar disponible para el tipo de uso previsto durante el desarrollo del proyecto.
- **Decisión tomada:** sustituir la dependencia externa por un sistema de datos y reproducción local.

**RF-5 – Buscar playlists de otros usuarios**
La aplicación trabaja únicamente con datos locales del propio usuario.

**RF-9 – Notificaciones**
No se han implementado notificaciones del sistema.

**RF-12 a RF-15 – Importación y exportación de datos**
Las funcionalidades de importación/exportación en CSV/JSON no están disponibles en esta versión.

## 2.2 Requisitos modificados

**RF-1 – Reproducción de música**

- **ERS:** reproducción mediante API externa (Spotify).
- **Entrega final:** reproducción de archivos de audio locales almacenados en el dispositivo.

**RC-IC-3 – Comunicación con API externa**

- Eliminado debido a la sustitución por arquitectura local.

# 3. Requisitos nuevos implementados (no aparecían en el ERS)

**RN-01 – Persistencia local completa**
Se implementa una base de datos local para:

- Usuarios
- Canciones
- Playlists

**RN-02 – Reproductor desacoplado de servicios externos**
La reproducción de música no depende de conexión a Internet.

**RN-03 – Inicialización de datos de ejemplo**
La aplicación incluye canciones precargadas para facilitar pruebas y demostraciones.

# 4. Comentarios sobre cambios realizados en el proyecto

- Se adaptó la arquitectura para eliminar dependencias externas (API Spotify).
- Se simplificó el alcance funcional priorizando estabilidad y demostrabilidad.
- Se revisó el control de roles para evitar accesos incorrectos a funciones administrativas.

# 5. Patrones de diseño GOF utilizados

- Singleton: Usado para la gestión de la base de datos local, garantizando una única instancia compartida.

- Repository para la base de datos.
- DAO (Data Access Object): Separación clara entre lógica de negocio y acceso a datos persistentes.
- MVC / MVVM (aproximación)
    - **Model:** entidades de usuario, canción y playlist.
    - **View:** Activities y layouts.
    - **Controller/ViewModel:** lógica de interacción.