StellarSound

Realizado por: Pablo Salamanca Sola

1. Descripción y justificación del proyecto

1.1. Descripción del proyecto

StellarSound es una aplicación de streaming de música online, cuyo diseño permitirá a los usuarios explorar, reproducir y gestionar sus propias librerías musicales alojados en un servidor privado. A diferencia de aplicaciones similares como Spotify o Apple Music, StellarSound ofrecerá un entorno personalizado y controlado sin restricciones en la experiencia de escucha musical.

El sistema contará con una API desarrollada en Laravel que servirá como puente entre la app móvil y la base de datos de MySQL, donde se almacenará la información pertinente de toda la aplicación. Además, Laravel Breeze será usado para la autenticación y gestión de inicio de sesión, para garantizar la seguridad de la app. También se integrará un panel administrativo donde los administradores podrán hacer gestión de usuarios, agregar canciones y realizar configuraciones del sistema.

El backend será alojado en un servidor privado, donde se almacenarán tanto los datos como los archivos de audio. Permitiendo a los administradores del sistema un control total del contenido y evitar dependencias de servicios externos.

1.2. Justificación del proyecto

En la actualidad, las plataformas de streaming musical han dominado el mercado con millones de usuarios que acceden a contenido en línea diariamente. Pero estas plataformas presentan las siguientes variaciones de las demás:

- Suscripciones mensuales y dependencias de las mismas, donde los usuarios podrán acceder a funciones avanzadas como la descarga de canciones o la eliminación de anuncios.
- Restricción de contenido, ciertas canciones pueden no estar disponibles por restricción de licencia.
- Falta de personalización, los usuarios deben ajustarse a la interfaz y características de la aplicación.
- Publicidad invasiva, las versiones gratuitas incluyen constantes anuncios que interrumpen la experiencia del usuario.

Con este contexto StellarSound busca ofrecer una alternativa más flexible y personalizable.

Beneficios del proyecto

- Autonomía y control, al usar API y bbdd propias los administradores pueden gestionar contenido sin una dependencia de terceros.
- Accesibilidad, no se requerirá ninguna suscripción para acceder a todo el contenido de la app.
- Personalización total, donde los usuarios pueden gestionar sus bibliotecas sin estar atados a algoritmos limitadores de selección de canciones.

- Privacidad, la información de los usuarios no será compartida con anunciantes de terceros.

2. Alcance del Proyecto

2.1. Backend con Laravel Breeze y MySQL

- Creación de API con Laravel.
- Implementación de Laravel Breeze para autenticación.
- Base de datos relacional en MySQL y Docker con estructura optimizada.
- Gestión de usuarios, roles y permisos (usuario y administrador).

2.2. Aplicación Android en Android Studio

- Interfaz intuitiva y atractiva.
- Consumo de API de autenticación y reproducción de canciones.
- Gestión de playlist y favoritos.
- Reproductor de música integrado con controles básicos.

2.3. Panel de Administración

- Subida de canciones al servidor.
- Gestión de usuarios y roles.
- Visualización de estadísticas y métricas de uso.

3. Tecnologías Utilizadas

Área	Tecnología
Backend	Laravel Breeze (PHP) → Framework potente para desarrollo de API REST rápida y segura. Breeze es un sistema de autenticación básica pero sin sobrecarga técnica. Se usará para el acceso a la base de datos y la gestión de comunicación de Request entre este y Android Studio.
Base de Datos	MySQL y Docker → Base de datos relaciones eficiente y con mucho soporte. Docker permite un fácil despliegue y escalabilidad, reduciendo los problemas de generación de usuarios por IP típicos de MySQL.
Frontend	Android Studio Java → IDE que permite la creación de apps para Android, con una fácil integración de APIs. Se usará para el visionado y gestión de canciones y playlists respectivamente.
API	RESTful API con Laravel → Estilo de arquitectura de API que permite la conexión cliente-servidor mediante peticiones HTTP. Será usado para las peticiones de datos entre Android Studio y Laravel
Control de Versiones	GitHub → Sistema de subida de versiones del proyecto.
Servidor	Ubuntu 24 → Sistema operativo libre y seguro, además de ligero, cuyo alojamiento de datos estará centrado en la base de datos y los archivos multimedia (canciones y portadas).
Autenticación	Laravel Breeze con Sanctum → Permite la autenticación mediante tokens personales, para una comunicación de usuarios entre la app y Laravel.

4. Objetivos

- Desarrollo de app Android intuitiva para exploración y reproducción de música.
- Implementación de un backend robusto con Laravel para gestionar la autenticación, usuarios, roles, canciones y playlists.
- Administración de la base de datos y archivos (canciones y portadas) en un servidor Linux
- Garantizar una arquitectura segura y escalable usando Docker.
- Habilitar una vista de gestión y administración de usuarios, contenido y estadísticas básicas.

5. Casos de uso

5.1. Autenticación de usuario

- Registro / Inicio mediante email y contraseña.
- Apreciación de la diferenciación de roles.

5.2. Exploración de canciones

- Visualización de la lista de canciones (por género, artista, recientes).
- Reproducción de audio mediante streaming en la app.

5.3. Gestión de playlists

- Crear, editar y eliminar playlists personales por usuario.
- Añadir o quitar canciones de la misma playlist.

5.4. Marcado de favoritos (likes)

- Permitir a los usuarios marcar sus canciones favoritas para un acceso rápido y generar estadísticas en base a esto.

5.5. Administración

- Vista de Laravel de administrador que permita hacer un CRUD de canciones y usuarios,

6. Requisitos del sistema

6.1. Funcionales

- Registro, login y autenticación de usuarios.
- CRUD de canciones (administrador).
- CRUD de playlists (usuario).
- Reproducción de canciones.
- Likes de canciones
- Diferenciación de roles y middleware de acceso.

6.2. No funcionales

- Escalabilidad para futuras ampliaciones.
- Alta disponibilidad y rendimiento de streaming.
- Seguridad: cifrado de contraseñas, protección CSRF en backend.

6.3. Requisitos de interfaz

- App Android simple, responsiva, moderna y minimalista.
- Navegación fluida y mínima carga visual.
- Backend con endpoints bien documentados.