Pestaña 1

**miNoise**

Documento de Arquitectura

Integrantes:  
Adolfo Maza

Alfredo Ulloa

Carla Sepulveda

Índice

[**Introducción 3**](#_2b7jvn733fzg)

[**Objetivo General 3**](#_cb09qsb7ochn)

[Objetivos Específicos 3](#_2fqpwc8bwvt2)

[**Arquitectura 4**](#_6so00bkg2tnz)

[Capa de Usuario (Interfaz 3D) 4](#_4zn2iq2q5eod)

[Capa de Presentación (Frontend) 4](#_vdd0vlojpc0l)

[Capa de Aplicación (Backend Flask) 5](#_dvb01tthxn0b)

[Capa de Integraciones Externas 5](#_s4tgslkrpao8)

[Capa de Datos 7](#_psacqknonjv4)

[Flujo general del sistema 8](#_pmcj1cqpz33v)

# Introducción

El presente documento de arquitectura describe la estructura técnica y los componentes del proyecto miNoise, un sistema interactivo de exploración musical en 3D. Este documento tiene como finalidad detallar cómo se organizan las distintas capas del sistema —desde la interfaz de usuario hasta la integración con APIs externas—, describiendo sus funciones, responsabilidades y flujos de información. Además, proporciona información suficiente para comprender la interacción entre los módulos, la persistencia de datos y los procesos de análisis y visualización musical.

La arquitectura presentada busca asegurar que el sistema sea modular, escalable y mantenible, permitiendo futuras ampliaciones y optimizaciones sin comprometer la funcionalidad principal: ofrecer un repositorio musical abierto y visualmente interactivo que facilite la exploración de géneros y artistas de manera innovadora.

# Objetivo General

Describir la arquitectura técnica del sistema miNoise, detallando sus componentes, capas, flujos de información y dependencias externas, con el propósito de garantizar un entendimiento claro de su funcionamiento y facilitar su desarrollo, mantenimiento y escalabilidad.

## Objetivos Específicos

1. Documentar las capas del sistema y sus componentes principales: usuario, presentación, aplicación, procesamiento, datos e integraciones externas.
2. Definir los flujos de información y procesos dentro del sistema, desde la extracción de datos hasta la visualización en 3D.
3. Especificar las tecnologías y librerías utilizadas en cada capa, justificando su elección.
4. Establecer las interacciones con APIs externas y los métodos de integración.
5. Proporcionar una guía de referencia para el mantenimiento y futuras ampliaciones del sistema.

# Arquitectura

## Capa de Usuario (Interfaz 3D)

**Componente:** Navegador web

**Rol**: Permitir la interacción con el sistema a través de una interfaz visual inmersiva.

**Tecnologías**:

* React
* Vite
* @react-three/fiber
* @react-three/drei
* Shader personalizado (DitherShader)

**Funciones clave:**

* Visualizar canciones como puntos/esferas 3D.
* Mostrar información al seleccionar una canción (nombre, artista, género, año).
* Permitir navegación libre (OrbitControls) y exploración del mapa musical 3D.

## Capa de Presentación (Frontend)

**Componente**: Aplicación React/Vite

**Rol**: Renderizar la visualización 3D y consumir los datos generados por el backend Flask.

**Entradas**:

* Archivo JSON resultante del procesamiento ( dataset.json )
* Endpoints expuestos por Flask (por ejemplo, /import\_dataset\_hybrid o /api/songs )
* Salidas:
* Mapa musical tridimensional basado en las coordenadas PCA (PC1, PC2, PC3).
* Tooltips dinámicos con información de cada canción.
* Fondo animado Shader Dither con estética retro.

**Comunicación**:

* Solicitudes HTTP mediante fetch o axios .
* Proxy configurado en vite.config.js hacia http://localhost:5000 .

## Capa de Aplicación (Backend Flask)

**Componente:** Servidor Flask

**Rol**: Gestionar las solicitudes, ejecutar el algoritmo principal y generar el archivo JSON con los datos

procesados.

**Módulos principales:**

* app.py — Controlador principal de la API.
* features\_librosa.py — Extracción de características de audio.

**Endpoints relevantes:**

* /import\_dataset\_hybrid → Ejecuta el pipeline de procesamiento.
* /api/songs → Devuelve el JSON final para el frontend.

## 

## Capa de Integraciones Externas

**Servicios conectados:**

| **Servicio** | **Propósito** | **Librería** |
| --- | --- | --- |
| Spotify | Obtener artistas y popularidad | requests |
| Last.fm | Obtener tags y artistas similares | requests |
| YouTube Music | Buscar canciones | ytmusicapi |
| YouTube | Descargar audio en WAV | yt-dlp |

Capa de Procesamiento y Análisis

**Componente:** import\_dataset\_logic()

**Rol:** Recolectar, enriquecer, analizar y reducir la información musical a un formato visualizable.

**Flujo interno:**

1. **Carga de semillas (``)**

Define los géneros base para la búsqueda inicial (≈30 por defecto).

1. **Extracción de datos desde APIs externas**
2. **Spotify API** → Popularidad, géneros y seguidores.
3. **Last.fm API** → Tags, artistas similares, oyentes.
4. **YouTube Music API (ytmusicapi)** → Búsqueda de canciones y artistas.
5. **Descarga de audio**
6. Se realiza con yt-dlp.
7. Archivos .wav temporales en audio\_cache/.
8. **Extracción de características acústicas (Librosa)**
9. MFCCs, Spectral Centroid, Rolloff, Tempo.
10. Se genera un vector de características por canción.
11. **Transformación y reducción dimensional (PCA)**
12. Normalización con StandardScaler.
13. Reducción a tres componentes principales ( PC1 , PC2 , PC3 ).
14. **Exportación final**
15. Se genera el archivo dataset.json.
16. Contiene artistas, canciones, features y coordenadas PCA.

## Capa de Datos

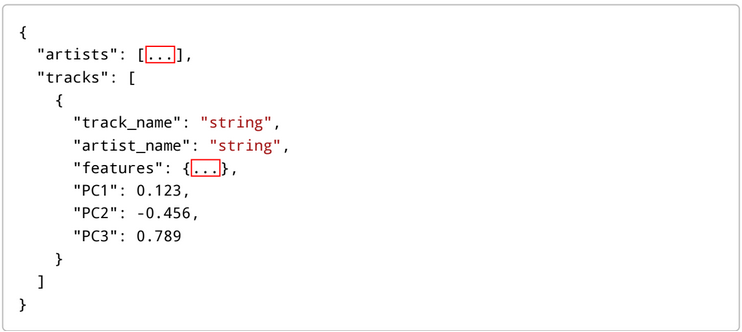
**Componente**: Archivo dataset.json

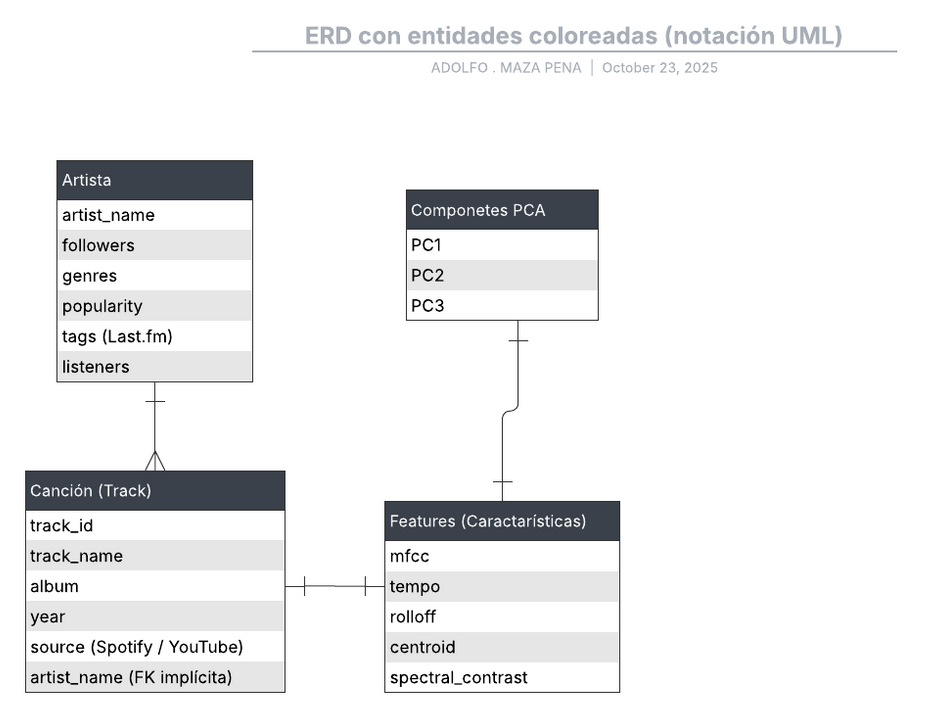
**Rol**: Actuar como una fuente única y temporal de datos.

**Características**:

* No se utiliza base de datos SQL.
* Persistencia local en formato JSON.
* Representa el resultado final del pipeline ETL.
* Puede regenerarse al ejecutar nuevamente import\_dataset\_logic() .

**Estructura general:**

****

****

## Flujo general del sistema

1. Flask ejecuta import\_dataset\_logic() .
2. Se conectan las APIs externas y se descargan audios.
3. Librosa analiza las características sonoras.
4. Se aplica PCA para obtener tres ejes representativos.
5. El sistema exporta el archivo dataset.json .
6. El frontend React lo consume y visualiza el mapa musical 3D interactivo.