



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**AI-Music**



Presentado por José Ángel López Estrada  
en Universidad de Burgos  
el 6 de julio de 2023

Tutores: César García Osorio  
Alicia Olivares Gil



---

# Índice general

---

<b>Índice general</b>	<b>i</b>
<b>Índice de figuras</b>	<b>iii</b>
<b>Índice de tablas</b>	<b>v</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	4
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	11
B.3. Catálogo de requisitos . . . . .	11
B.4. Especificación de requisitos . . . . .	13
<b>Apéndice C Especificación de diseño</b>	<b>21</b>
C.1. Introducción . . . . .	21
C.2. Diseño de datos . . . . .	21
C.3. Diseño procedimental . . . . .	22
C.4. Diseño arquitectónico . . . . .	25
C.5. Diseño de la interfaz . . . . .	27
<b>Apéndice D Documentación técnica de programación</b>	<b>33</b>
D.1. Introducción . . . . .	33
D.2. Estructura de directorios . . . . .	33

D.3. Manual del programador . . . . .	34
D.4. Compilación, instalación y ejecución del proyecto . . . . .	37
D.5. Pruebas del sistema . . . . .	37
<b>Apéndice E Documentación de usuario</b>	<b>41</b>
E.1. Introducción . . . . .	41
E.2. Requisitos de usuarios . . . . .	41
E.3. Instalación y ejecución . . . . .	42
E.4. Manual del usuario . . . . .	42
<b>Bibliografía</b>	<b>51</b>

---

# Índice de figuras

---

B.1. Diagrama general de casos de uso. . . . .	13
C.1. Funcionamiento básico de la aplicación - Diagrama de secuencia	22
C.2. Extracción de características de un fichero de audio - Diagrama de secuencia . . . . .	23
C.3. Extracción de características de los datos de entrenamiento - Diagrama de secuencia . . . . .	24
C.4. Proceso de entrenamiento del modelo - Diagrama de secuencia .	24
C.5. Proceso de visualización de características - Diagrama de secuencia	25
C.6. Esquema con las capas de la aplicación. . . . .	27
C.7. Pantalla de inicio de la aplicación. . . . .	28
C.8. Pantalla de selección de modelos y reproducción de audio. . . .	30
C.9. Pantalla de visualización de resultados. . . . .	31
D.1. Consola de Git. . . . .	35
D.2. Visual Studio Code. . . . .	36
D.3. Jupyter Notebook. . . . .	36
D.4. Código del test unitario que comprueba el número correcto de generación de predicciones. . . . .	38
D.5. Resultado de la ejecución del test unitario. . . . .	38
D.6. Resultado obtenido. Es correcto. . . . .	39
D.7. Resultado obtenido. Es correcto. . . . .	39
E.1. Subida de canciones. . . . .	43
E.2. Información sobre la canción subida. . . . .	44
E.3. Selección de algoritmo. . . . .	45
E.4. Visualización del resultado: Onda de audio. . . . .	46
E.5. Visualización del resultado: Espectrograma. . . . .	47

E.6. Visualización del resultado: Cromograma . . . . .	48
E.7. Visualización del resultado: MFCC. . . . .	49

---

# Índice de tablas

---

A.1. Costes de hardware . . . . .	5
A.2. Costes de software para el desarrollo del proyecto . . . . .	6
A.3. Otros costes para el desarrollo del proyecto . . . . .	6
A.4. Costes totales de desarrollo del proyecto . . . . .	6
A.5. Licencias de las herramientas utilizadas . . . . .	8
B.1. CU-1 Carga de la canción . . . . .	14
B.2. CU-2 Extracción de características de audio . . . . .	15
B.3. CU-3 Ejecución del modelo de machine learning . . . . .	16
B.4. CU-4 Visualización de la predicción . . . . .	17
B.5. CU-5 Reproducción de la canción . . . . .	18
B.6. CU-6 Carga de una nueva canción . . . . .	19
E.1. Requisitos para el funcionamiento local del sistema. . . . .	41





## *Apéndice A*

---

# Plan de Proyecto Software

---

### A.1. Introducción

Este plan de proyecto de software se refiere al desarrollo de una aplicación que clasifica automáticamente los géneros musicales basándose en características de audio. Este proyecto ha sido concebido como un trabajo de fin de grado y ha implicado la implementación de técnicas de aprendizaje automático y procesamiento de señales de audio digitales para lograr el objetivo. La aplicación busca proporcionar una solución efectiva para clasificar pistas musicales de forma automática.

En este apéndice se describirá la planificación sobre la que se ha desarrollado el proyecto.

### A.2. Planificación temporal

El desarrollo del proyecto se organizó siguiendo la metodología ágil de SCRUM, con iteraciones semanales o «sprints». Cada sprint implicó una serie de actividades que culminaron con una entrega incremental del proyecto.

**Sprints totales:** 14

#### **Sprint 1 (08/03/2023 - 15/03/2023)**

En el Sprint 1, se realizó la configuración inicial del proyecto, añadiendo una descripción detallada en el archivo `README.md` y estableciendo las bases de la documentación. Asimismo, se implementaron componentes básicos y

se trabajó en el diseño de la aplicación, aprovechando las funcionalidades de la biblioteca *librosa* para el procesamiento de audio.

### **Sprint 2 (15/03/2023 - 22/03/2023)**

En el Sprint 2, se configuró el entorno de desarrollo utilizando Poetry, que posteriormente se descartó ante el uso de entornos virtuales clásicos, y se implementó el módulo `train_classifier` para entrenar el clasificador de géneros musicales. De igual manera, también se llevó a cabo la extracción de características MFCC y se realizó un preprocesamiento básico de los datos.

### **Sprint 3 (22/03/2023 - 05/04/2023)**

En el Sprint 3, se realizaron pequeños cambios en el código e investigación sobre el área.

### **Sprint 4 (05/04/2023 - 12/04/2023)**

En el Sprint 4, se realizaron diversas tareas. En primer lugar, se agregó un archivo `requirements.txt` para gestionar las dependencias del proyecto y se creó la estructura inicial de Flask para desarrollar la aplicación web. Complementariamente, se diseñó una aplicación Flask básica y se llevó a cabo el preprocesamiento de géneros musicales. Además, se realizaron ajustes en la estructura del proyecto y se actualizó la extracción de características MFCC. También se implementó la visualización de características de audio, como la forma de onda, el espectrograma, el cromagrama y los MFCC. Finalmente, se realizaron mejoras en el estilo y se añadió un modelo de aprendizaje automático inicial.

### **Sprint 5 (12/04/2023 - 19/04/2023)**

En el Sprint 5, se trabajó en los aspectos visuales del proyecto. Se crearon mockups e imágenes de prueba para la interfaz de usuario, permitiendo una mejor comprensión de cómo se presentaría la aplicación.

### **Sprint 6 (19/04/2023 - 26/04/2023)**

En el Sprint 6, se avanzó en la sección de carga del proyecto y se añadió una página de inicio. Además, se implementó un reproductor de música que permitía a los usuarios escuchar las pistas seleccionadas.

**Sprint 7 (26/04/2023 - 10/05/2023)**

En el Sprint 7, se llevaron a cabo varias tareas importantes. Se añadió un nuevo modelo al sistema y se actualizó la documentación correspondiente. Paralelamente, también se trabajó en la implementación de las páginas de registro e inicio de sesión para los usuarios, aunque finalmente no pudieron ser llevadas a cabo. También se realizaron ajustes en el guardado del modelo y se añadieron las páginas de inicio de sesión y registro al índice del proyecto.

**Sprint 8 (10/05/2023 - 17/05/2023)**

En el Sprint 8, se dedicó tiempo a la búsqueda de referencias y documentación relevante para el proyecto. Se recopilaron trabajos relacionados, papers e información general que proporcionaron una base sólida para la consolidación de la aplicación.

**Sprint 9 (17/05/2023 - 24/05/2023)**

En el Sprint 9, se realizaron mejoras en la extracción de características y se llevó a cabo un proceso de limpieza, eliminando archivos de borrador que ya no eran necesarios.

**Sprint 10 (24/05/2023 - 02/06/2023)**

En el Sprint 10, se añadió documentación adicional para complementar el proyecto, asegurando que todos los aspectos importantes estuvieran debidamente explicados.

**Sprint 11 (02/06/2023 - 09/06/2023)**

En el Sprint 11, se implementaron generadores para la fase de entrenamiento del modelo, lo que permitió una mejor gestión de los datos. También se actualizó la representación de géneros predichos.

**Sprint 12 (09/06/2023 - 16/06/2023)**

En el Sprint 12, se realizaron cambios en el diseño de la aplicación, mejorando la representación visual de las características de audio. Se llevaron a cabo actualizaciones en el diseño de los colores utilizados en la interfaz.

### **Sprint 13 (16/06/2023 - 23/06/2023)**

En el Sprint 13, se realizaron varios cambios significativos. Por un lado se implementaron diferentes mejoras en el procesamiento y entrenamiento de los datos. De igual modo, se actualizaron el reproductor de audio y la información de las pistas. Además, se realizaron múltiples actualizaciones en la memoria, los anexos y la aplicación web. Eventualmente, se llevaron a cabo correcciones menores de estilo para mejorar la presentación general.

### **Sprint 14 (23/06/2023 - 05/06/2023)**

En el Sprint 14, se continuaron realizando actualizaciones y mejoras en la documentación. Se realizaron cambios específicos en la sección B de los anexos, relacionada con los requisitos funcionales y no funcionales. También se actualizaron las secciones A, D y E de los anexos, que incluyen el plan del proyecto, el manual del programador y el manual de usuario, respectivamente. Se agregó un diagrama general de casos de uso y se llevaron a cabo actualizaciones adicionales en la documentación, la aplicación web y los anexos. Además se desplegó la aplicación web en un servidor en la nube, concretamente en la plataforma PythonAnywhere.

## **A.3. Estudio de viabilidad**

### **Viabilidad económica**

La viabilidad económica de un proyecto de software se refiere a la capacidad para generar ingresos tanto para cubrir el desarrollo como para proporcionar una rentabilidad a medio y largo plazo. El presente proyecto no está planteado con un objetivo económico, por lo que esta sección se va a dedicar a explorar un posible plan de monetización de forma ficticia, explorando los costes totales de desarrollo de software y potenciales métodos de monetización.

### **Costes de hardware**

Los costes de hardware se refieren al gasto económico que se realiza para obtener los diferentes elementos hardware que se necesitan para poder llevar a cabo el desarrollo del proyecto. Los elementos hardware pueden ser ordenadores, periféricos o componentes internos por ejemplo. En este caso se ha planteado la compra de un ordenador portátil de unos 1000€. Esta decisión se toma teniendo en cuenta los siguientes factores:

- **Rendimiento:** Este proyecto requiere un uso de algoritmos de aprendizaje automático y procesamiento de datos, por lo que se demanda una cierta capacidad de computación. Un equipo dentro del rango de precio planteado viene equipado con un procesador potente, como un Intel Core i7 (12th o 13th Gen) o un AMD Ryzen 7 (6th Gen). Además suelen incluir una tarjeta gráfica dedicada, como una NVIDIA RTX 3060 con 6 GB de VRam, con la ventaja en rendimiento que esto conlleva en el uso de bibliotecas de *Deep Learning* como TensorFlow.
- **Durabilidad:** Los ordenadores de este rango de precio suelen contar con componentes de mayor calidad asegurando una mayor vida útil. La durabilidad es importante en proyectos como este ya que un fallo de hardware podría tener un impacto significativo en la productividad.
- **Soporte:** Los fabricantes proporcionan un soporte de mayor calidad en ordenadores situados en este rango de precio. Este soporte puede incluir mejores condiciones de garantía o actualizaciones más duraderas en partes vitales del sistema como la BIOS.

Concepto	Coste
Ordenador portátil	1000€
Total	1000€

Tabla A.1: Costes de hardware

## Costes de software

Los costes de software se refieren al gasto económico que se realiza en la adquisición de las distintas licencias de software que son necesarias para la realización del proyecto.

- **Sistema Operativo:** Se ha elegido el sistema operativo Windows 11 Home como base para realizar el desarrollo del proyecto. Se podrían plantear alternativas gratuitas como alguna distribución de Linux pero por sencillez y extensión de uso se ha elegido Windows.
- **Servicios en la nube:** Durante las etapas intensivas de entrenamiento de modelos y almacenamiento de datos, podría llegar a ser necesario el uso de servicios de computación en la nube como AWS [1] o Microsoft Azure [2]. El coste puede variar según el uso necesario. Por ejemplo en un proyecto de estas características, contando con unos 100 GB de datos en AWS, se estimaría un costo de almacenamiento de unos 2€/mensuales y un costo de procesamiento de unos 30€/mensuales,

utilizando 10 horas en una instancia `p3.2xlarge`, la cual es comúnmente utilizada para labores de aprendizaje automático. En total, el coste de la computación en la nube a lo largo del proyecto ha sido de 192€.

Concepto	Coste
Windows 11 Home [3]	145€
Computación en la nube	192€
<b>Total</b>	<b>337€</b>

Tabla A.2: Costes de software para el desarrollo del proyecto

## Otros costes

Concepto	Coste
Electricidad (mensual)	50€
Internet (mensual)	35€
Espacio de trabajo (mensual)	350€
<b>Total (mensual)</b>	<b>435€</b>
<b>Total (6 meses)</b>	<b>2610€</b>

Tabla A.3: Otros costes para el desarrollo del proyecto

## Costes totales

Teniendo en cuenta una duración de desarrollo de proyecto de 6 meses, el coste total ha sido:

Concepto	Coste
Costes de hardware	50€
Costes de software	32€
Otros costes	2610€
<b>Total</b>	<b>2692€</b>

Tabla A.4: Costes totales de desarrollo del proyecto

## Ingresos

La idea del proyecto es generar ingresos hasta el punto de rentabilizar el gasto de desarrollo como mínimo. Existen diversas formas en las que la aplicación podría generar ingresos como por ejemplo:

- **Sistema de suscripción:** Se puede ofrecer la aplicación con distintas características según el *tier* o el tipo de suscripción. Podrían existir tres *tier*:
  - **Suscripción gratuita:** límite 5 canciones/diarias
  - **Suscripción silver 5€/mes:** canciones ilimitadas.
  - **Suscripción gold 15€/mes:** canciones ilimitadas y análisis con más detalle.
- **Publicidad:** Se puede monetizar el servicio a través de anuncios. Los usuarios que usen la versión gratuita de la aplicación verán anuncios en distintas partes de la interfaz, mientras que si pagan una mensualidad (5€/mes) tiene la opción de eliminar anuncios.
- **Integraciones de API:** Se puede ofrecer un servicio de uso de la API a aplicaciones de terceros para utilizar la aplicación. Esta API puede ser gratuita, con limitaciones en el número de canciones a predecir, o de pago, con un menor número de limitaciones.

## Viabilidad legal

La viabilidad legal del proyecto se refiere al cumplimiento de diversas leyes y obligaciones a la hora de desarrollar el software.

## Derechos de autor

El tema de los derechos de autor en el mundo de la música es un tema complicado. Los derechos de autor de la música son un área legal compleja que envuelve partes como artistas, productores y discografías. Este proyecto va a utilizar un conjunto de datos musicales libres de derechos de autor lo que facilita el trabajo eliminando esta parte del proceso de desarrollo.

A la hora de la subida de archivos musicales para su clasificación existe un problema, el usuario final puede subir cualquier fichero de audio, tenga derechos de autor o no. Por lo tanto, el procedimiento a realizar será la eliminación de cualquier fichero musical de la aplicación tras realizar el proceso de predicción.

## Licencias de software

Según el software utilizado, se tienen en cuenta las siguientes licencias [4, 5]:

Tabla A.5: Licencias de las herramientas utilizadas

Herramienta	Licencia
Python [6]	PSF
Librosa [7]	ISC
TensorFlow [8]	Apache 2.0
ScikitLearn [9]	BSD-3-Clause
Flask [10]	BSD-3-Clause
FMA Dataset [11]	CC BY-NC-SA 4.0
Matplotlib [12]	PSF
Pandas [13]	BSD-3-Clause
Numpy [14]	BSD-3-Clause

Según las herramientas utilizadas, se ha elegido la licencia MIT para la distribución y uso del software desarrollado. Esta es una licencia de código abierto permisiva, la cual permite a los usuarios utilizar, modificar y distribuir el software bajo ciertas condiciones, requiriendo la inclusión del aviso de derechos de autor y la renuncia de responsabilidad en el software distribuido. Una de las características principales de la licencia MIT es su flexibilidad. Permite a los usuarios utilizar el software con pocas restricciones y sin imponer condiciones adicionales. Además, ofrece libertad para modificar y distribuir el software tanto en proyectos de código abierto como en proyectos comerciales. La elección de la licencia MIT se basa en su adecuación a las necesidades y especificidades de este proyecto. Algunas razones por las cuales es la licencia que mejor se ajusta son:

1. **Flexibilidad y colaboración:** La licencia MIT [5] permite que otras personas utilicen, modifiquen y distribuyan el software sin imponer restricciones excesivas. Esto fomenta la colaboración y la participación de la comunidad, lo cual es beneficioso para un proyecto de desarrollo de software.
2. **Compatibilidad:** La licencia MIT es compatible con la mayoría de las otras licencias de código abierto. Esto es importante, ya que el proyecto utiliza diversas herramientas con diferentes licencias, asegurando que no haya conflictos y permitiendo una integración fluida entre ellas.
3. **Reconocimiento y protección de derechos de autor:** La licencia MIT incluye disposiciones que protegen los derechos de autor y exigen



la inclusión del aviso de derechos de autor en el software distribuido. Esto garantiza el reconocimiento adecuado del autor original del proyecto y protege sus derechos.



## *Apéndice B*

---

# Especificación de Requisitos

---

### B.1. Introducción

Esta sección describe la especificación de requisitos para el proyecto. Se proporciona información detallada sobre los requisitos funcionales (RF) y requisitos no funcionales (RNF) así como el detalle de los casos de uso (CU).

### B.2. Objetivos generales

- Desarrollar un sistema de reconocimiento de estilos musicales utilizando inteligencia artificial.
- Diseñar e implementar una aplicación web que permita usar el modelo de una forma sencilla.
- Obtener conclusiones y conocimiento a partir de los datos.

### B.3. Catálogo de requisitos

#### Requisitos funcionales

Un requisito funcional es una especificación que describe lo que un sistema debe hacer o cómo debe comportarse. Los requisitos funcionales pueden incluir detalles como cálculos, manipulación de datos, interacción con el usuario, etc [15]. A continuación se listan los requisitos funcionales extraídos para este proyecto:

- **RF-1 Predicción de estilo musical:** la aplicación debe ser capaz de predecir el estilo musical de una canción.
  - **RF-1.1 Extracción de características de audio:** la aplicación debe ser capaz de extraer características relevantes de la canción que se utilizarán para predecir el estilo musical.
  - **RF-1.2 Implementar un modelo de machine learning para realizar la predicción:** la aplicación debe implementar un modelo de *machine learning* que se entrenará con las características de un conjunto de datos musical.
- **RF-2 Presentación de la información:** la aplicación debe mostrar información relevante sobre la predicción realizada.
  - **RF-2.1 Detalles de la predicción:** la aplicación debe ser capaz de mostrar datos sobre los detalles de la predicción como la confianza (probabilidad) o posibles géneros musicales alternativos.
  - **RF-2.2 Información de la canción:** la aplicación debe mostrar información relevante sobre la pista de audio o canción utilizada como la duración, formato o frecuencia de muestreo.
- **RF-3 Implementación de una interfaz de usuario:** la aplicación debe contener una interfaz de usuario para facilitar su uso.
  - **RF-3.1 Funcionalidad de carga de archivos:** la interfaz de usuario debe incluir una función de carga de archivos que permita seleccionar y cargar una canción para realizar la predicción.
  - **RF-3.2 Representación de los resultados:** la interfaz de usuario debe incluir sección donde se presenten de forma clara los resultados de la predicción.

## Requisitos no funcionales

Los requisitos no funcionales se centran en las características del sistema que no están directamente relacionadas con su comportamiento, por ejemplo rendimiento o seguridad [15]. A continuación se listan los requisitos no funcionales extraídos para este proyecto:

- **RNF-1 Rendimiento:** la aplicación debe ser capaz de procesar y analizar una canción y devolver los resultados dentro de un tiempo razonable, idealmente unos pocos segundos.
- **RNF-2 Seguridad:** la aplicación debe cumplir con protocolos de seguridad como cifrado de comunicaciones o la eliminación de la canción después de su procesamiento.

- **RNF-3 Compatibilidad:** la aplicación debe ser compatible con diversos navegadores y sistemas operativos.
- **RNF-4 Usabilidad:** la aplicación debe ser fácil de usar para el usuario. El usuario debería ser capaz de realizar el proceso de subida de archivos y comprobar la predicción y resultados de una forma sencilla.
- **RNF-5 Portabilidad:** la aplicación debe ser compatible con diversos dispositivos como PCs, tablets o dispositivos móviles.

## B.4. Especificación de requisitos

Los casos de uso son descripciones detalladas del funcionamiento de una parte del sistema desde la perspectiva del usuario.

En esta sección se mostrará el diagrama general de casos de uso y se explicará en detalle cada uno de ellos.

### Diagrama general de casos de uso

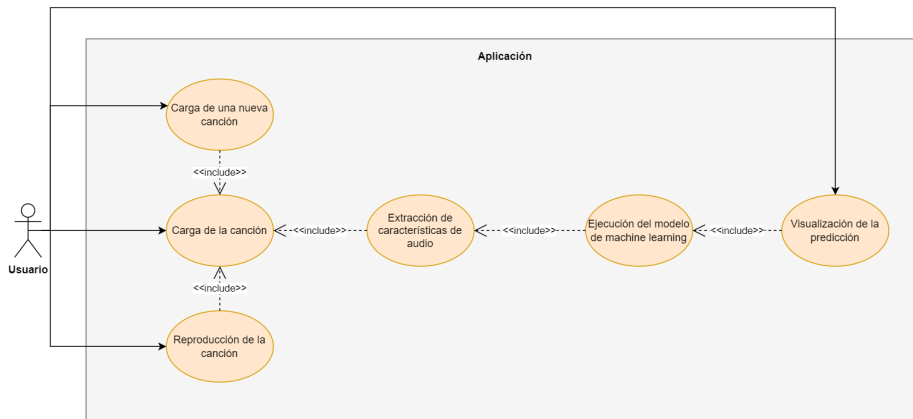


Figura B.1: Diagrama general de casos de uso.

### Casos de uso

A continuación se exponen las especificaciones de los casos de uso que forman la aplicación.

CU-1	Carga de la canción
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-1.1, RF-3.1
<b>Descripción</b>	El usuario utiliza la función de carga de archivo para seleccionar y cargar una canción en la aplicación.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación se encuentra en funcionamiento.</li> <li>2. El usuario tiene acceso a los archivos de música que desea cargar.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción para cargar una canción pulsando el botón <i>Upload File</i>.</li> <li>2. El usuario busca y selecciona la canción que desea cargar desde su almacenamiento local.</li> <li>3. La canción se carga en la aplicación.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. La canción se encuentra cargada en la aplicación.</li> <li>2. La aplicación está lista para extraer características y realizar el proceso de predicción de la canción cargada.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. El archivo cargado no es un fichero de audio.</li> <li>2. La canción tiene un formato incompatible.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.1: CU-1 Carga de la canción

CU-2	Extracción de características de audio.
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-1.1
<b>Descripción</b>	La aplicación extrae las características relevantes del audio para su posterior análisis.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. La canción se ha cargado correctamente en la aplicación.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación inicia el proceso de extracción de características del archivo de audio cargado.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. Las características relevantes de la canción se han extraído correctamente.</li> <li>2. La aplicación está lista para utilizar las características extraídas para realizar la predicción del estilo musical.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. Ocurrió un error durante la extracción de las características del audio.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.2: CU-2 Extracción de características de audio

CU-3	Ejecución del modelo de machine learning
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-1.2
<b>Descripción</b>	La aplicación ejecuta el modelo de machine learning para realizar la predicción del estilo musical según las características extraídas.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. Las características de audio de la canción se han extraído correctamente.</li> <li>2. Las características de audio de la canción son suficientes para realizar el proceso de predicción.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El modelo procesa las características extraídas de la canción.</li> <li>2. Se obtiene el resultado de la predicción, el cual consiste en un vector de probabilidades para diferentes estilos musicales.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación ha generado una predicción sobre el estilo musical de la canción.</li> <li>2. La aplicación está lista para mostrar los detalles de la predicción al usuario.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. El modelo no puede procesar las características extraídas.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.3: CU-3 Ejecución del modelo de machine learning



CU-4	Visualización de la predicción
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-2.1, RF-2.2
<b>Descripción</b>	Tras la ejecución del modelo de <i>machine learning</i> , la aplicación muestra la predicción del estilo musical al usuario.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación ha generado una predicción.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación muestra las tres predicciones más probables de los estilos musicales en la interfaz.</li> <li>2. El usuario puede cambiar entre distintas visualizaciones de forma dinámica.</li> <li>3. El usuario visualiza las predicciones.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. El usuario ha obtenido la predicción del estilo musical de la canción.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. Se produce un error al intentar mostrar la predicción.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.4: CU-4 Visualización de la predicción

CU-5	Reproducción de la canción
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-1.2, RF-1.3
<b>Descripción</b>	La aplicación permite al usuario reproducir la canción cargada.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. La canción ha sido cargada correctamente en la aplicación.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción de reproducir la canción mediante el botón <i>play</i> del reproductor.</li> <li>2. La aplicación reproduce la canción cargada.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. La canción está siendo reproducida en la aplicación.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. Se produce un error en la reproducción de la canción.</li> </ol>
<b>Importancia</b>	Media

Tabla B.5: CU-5 Reproducción de la canción

CU-6	Carga de una nueva canción
<b>Versión</b>	1.0
<b>Autor</b>	José Ángel López
<b>Requisitos asociados</b>	RF-1.1, RF-1.3
<b>Descripción</b>	El usuario puede cargar una nueva canción sin tener que cerrar o reiniciar la aplicación.
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. La aplicación ha realizado la predicción de una canción.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona el nombre de la aplicación, en la zona superior izquierda de la pantalla, para volver a la página inicial.</li> <li>2. El usuario pulsa el botón <i>Upload File</i>.</li> <li>3. El usuario busca y selecciona la nueva canción que desea cargar desde su almacenamiento.</li> </ol>
<b>Postcondiciones</b>	<ol style="list-style-type: none"> <li>1. La nueva canción se carga en la aplicación.</li> </ol>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. El archivo cargado no es un fichero de audio.</li> <li>2. La canción tiene un formato incompatible.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.6: CU-6 Carga de una nueva canción



## Apéndice C

---

# Especificación de diseño

---

### C.1. Introducción

En esta sección se define cómo se han implementado las especificaciones técnicas vistas en el apéndice B. En concreto cómo se han implementado las especificaciones técnicas y cómo se han estructurado los datos y los procedimientos. También se describirá la arquitectura general del sistema.

### C.2. Diseño de datos

La aplicación maneja varios tipos de datos, que se describen a continuación:

- **Archivos CSV:** Los archivos CSV son utilizados para almacenar información sobre los metadatos de los archivos de audio [16]. En concreto, se trabaja con el fichero `tracks.csv` que contiene una gran cantidad de información sobre cada pista de audio y con el fichero `raw_genres.csv` que contiene información sobre cada género musical. Estos ficheros se procesan para generar el archivo `track_genres.csv`, que contiene el identificador de la pista de audio y sus géneros, y es más sencillo para su uso a la hora de introducir las características y entrenar el modelo.
- **Archivos de audio:** Los archivos de audio, en formato `.mp3` [17], son los datos de entrada principales en la aplicación. Estos son analizados y procesados para extraer las características necesarias para su predicción.

- **Archivos pickle:** Se ha optado por el uso de este tipo de ficheros para almacenar el conjunto de datos de entrenamiento procesado, `track_genres_mfcc.pkl`, debido a que mantiene mejor la información de las características que un fichero CSV tradicional. Este fichero se utiliza para entrenar el modelo de aprendizaje automático [18].

### C.3. Diseño procedimental

#### Procedimiento de funcionamiento básico

El diseño procedimental de la aplicación sigue un flujo de trabajo definido de la siguiente forma:

1. **Carga de datos:** El usuario carga un fichero de audio que se desea clasificar.
2. **Procesamiento de audio:** La aplicación procesa el audio extrayendo características necesarias para su clasificación.
3. **Clasificación de audio:** El modelo de aprendizaje automático realiza la predicción del género musical del fichero de audio.
4. **Visualización del resultado:** La aplicación devuelve el resultado de las predicciones al usuario.

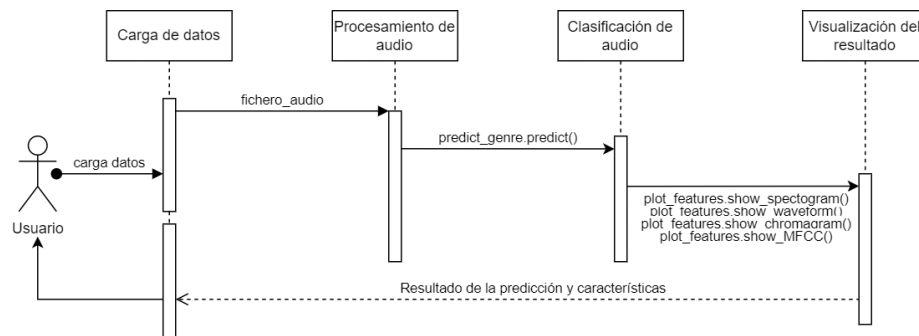


Figura C.1: Funcionamiento básico de la aplicación - Diagrama de secuencia

#### Procedimiento de extracción de características de audio de un fichero

1. **Carga de datos:** El archivo de audio es cargado por la aplicación.

2. **Procesamiento de audio:** Se realiza un procesado del archivo recortando su longitud.
3. **Extracción de características:** Mediante el uso de la biblioteca *librosa* se extraen las características MFCC del fichero de audio.
4. **Normalización de las características:** Las características extraídas se normalizan para que tengan el mismo peso en el entrenamiento o predicción.

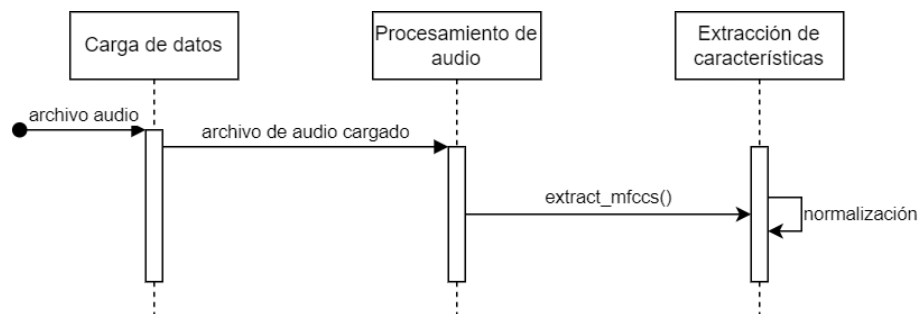


Figura C.2: Extracción de características de un fichero de audio - Diagrama de secuencia

## Procedimiento de extracción de características de audio para los ficheros de entrenamiento

1. **Carga de datos:** Se recorren de forma iterativa todos los ficheros de audio en la carpeta `/data/raw` y se realiza el proceso siguiente individualmente en cada fichero.
2. **Procesamiento de audio:** Se realiza un procesado de los archivos recortando su longitud.
3. **Extracción de características:** Mediante el uso de la biblioteca *librosa* se extraen las características MFCC de los ficheros de audio.
4. **Normalización de las características:** Las características extraídas se normalizan para que tengan el mismo peso en el entrenamiento o predicción.
5. **Salida en archivo:** Las características extraídas se guardan junto a sus géneros musicales en un archivo.

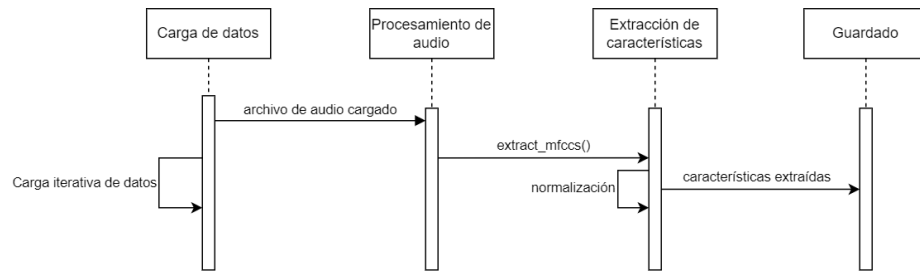


Figura C.3: Extracción de características de los datos de entrenamiento - Diagrama de secuencia

## Procedimiento de entrenamiento del modelo

1. **Carga de datos:** Los datos de entrenamiento (características MFCC y géneros musicales) son cargados en memoria.
2. **Partición de los datos:** Se divide el conjunto de datos en un conjunto de entrenamiento, test y validación.
3. **Creación del modelo:** Se crea una instancia de la red neuronal que se va a utilizar para realizar el entrenamiento.
4. **Compilación del modelo:** Se elige un optimizador, una función de pérdida y la métrica objetivo que se desea utilizar y se compila el modelo.
5. **Entrenamiento del modelo:** Se entrena y valida el modelo con los datos de entrenamiento y de validación.
6. **Guardado del modelo:** Una vez termina el proceso de entrenamiento se guarda el modelo en el sistema.

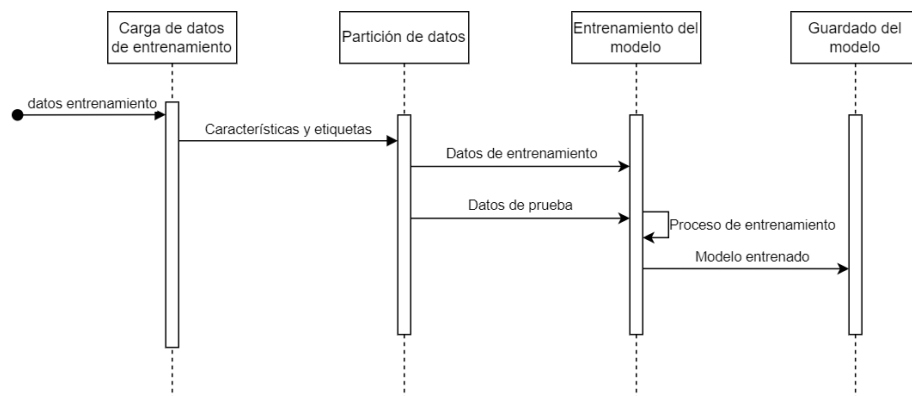


Figura C.4: Proceso de entrenamiento del modelo - Diagrama de secuencia



## Visualización de características

1. **Carga de datos:** El archivo de audio se carga en memoria.
2. **Extracción de características:** Mediante el uso de la biblioteca `librosa` se extraen las características principales del archivo de audio.
3. **Generación de gráficos:** Mediante el uso de la biblioteca de visualización `Matplotlib`, se generan gráficos que representan visualmente las características extraídas.
4. **Visualización de gráficos:** Los gráficos generados se envían al servidor web para que el usuario pueda visualizarlos.

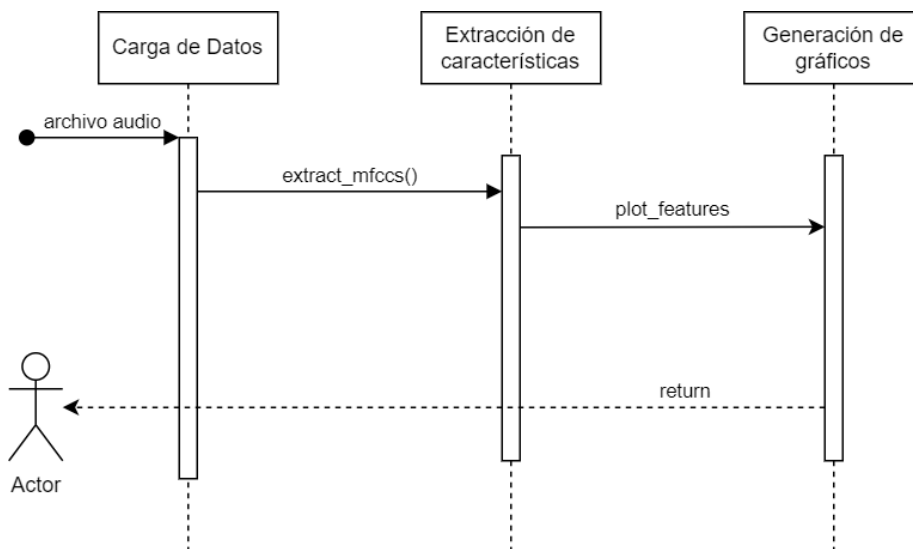


Figura C.5: Proceso de visualización de características - Diagrama de secuencia

## C.4. Diseño arquitectónico

El diseño arquitectónico de la aplicación sigue un modelo de capas [19]. El modelo de capas es un arquitectura de software en las que las responsabilidades del software se dividen en capas apiladas. Cada capa ofrece servicios a la capa superior y los recibe de la capa inferior.

### Capa de presentación

La capa de presentación se ocupa de la interacción e interfaz de usuario. Utiliza la biblioteca `Flask` para funcionar y permite a los usuarios cargar

archivos musicales, iniciar el proceso de predicción y visualizar los resultados finales. Es en esta capa donde se renderizan los archivos HTML [20] y donde se realiza el intercambio de mensajes mediante el protocolo HTTP [21]. Esta capa está comunicada con la capa de lógica, que le devolverá la información para poder mostrar los resultados.

## Capa de lógica

La capa de lógica o lógica de negocio es donde se produce el funcionamiento interno de la aplicación. El funcionamiento de la lógica de la aplicación funciona bajo el lenguaje de programación `Python` y varias bibliotecas. En concreto se ejecutan las siguientes funcionalidades:

- **Extracción de características de audio**
- **Procesamiento de los datos de audio**
- **Entrenamiento del modelo**

## Capa de acceso a datos

Esta capa se ocupa del almacenamiento y recuperación de datos. Los datos con los que trabaja esta capa son: archivos de audio, archivos CSV, archivos de características, etc.

Esta capa se comunica con la capa de lógica, proporcionándole los datos necesarios para su funcionamiento y almacenando sus resultados.



Figura C.6: Esquema con las capas de la aplicación.

## C.5. Diseño de la interfaz

La interfaz de usuario proporciona la interacción directa entre el usuario y el sistema. En la aplicación la interfaz de usuario está centrada en la facilidad de uso y la visualización de los resultados.

### Interfaz de usuario

El diseño de la interfaz de usuario está construido utilizando el lenguaje de marcado HTML. Flask se utiliza para definir las solicitudes y las respuestas HTTP y que de este modo, el navegador pueda renderizar los archivos HTML que forman la interfaz de usuario. Las imágenes utilizadas en la interfaz de usuario se han obtenido de la plataforma *Flaticon* [22], la fuente utilizada es Poppins y ha sido extraída de *Google Fonts* [23].

Incluye los siguientes elementos:

**Pantalla de inicio:** La pantalla de inicio es lo primero que ve el usuario al iniciar la aplicación. Proporciona una breve introducción sobre el propósito y funcionamiento de la aplicación y permite cargar un archivo de audio para iniciar el proceso de predicción musical.

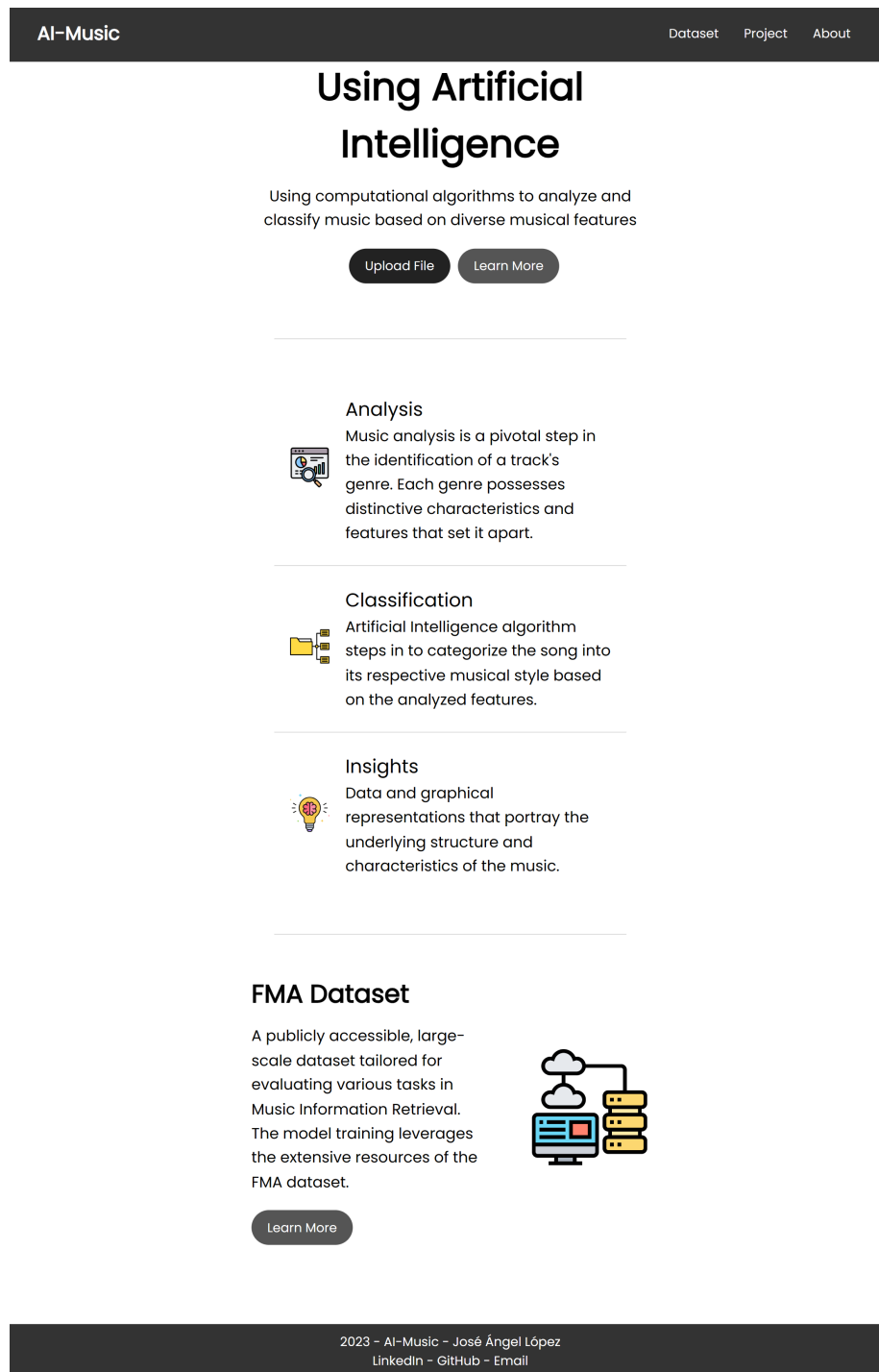


Figura C.7: Pantalla de inicio de la aplicación.

**Selección de modelos y reproducción de audio:** Esta sección permite al usuario ver información básica sobre la pista de audio subida, como la duración o la frecuencia de muestreo, y elegir qué modelo entrenado se va a utilizar para realizar la predicción musical.

AI-Music


DatasetProjectAbout

Select your algorithm

Initial Model

Submit

Information




Name: Juan RIOS  cosmos


Sample rate: 22050Hz

Format: .mp3

Duration: 3:14 min


0:00 / 3:14






Analysis

Music analysis is a pivotal step in the identification of a track's genre. Each genre possesses distinctive characteristics and features that set it apart.



Classification

Artificial Intelligence algorithm steps in to categorize the song into its respective musical style based on the analyzed features.




Insights

Data and graphical representations that portray the underlying structure and characteristics of the music.

FMA Dataset

A publicly accessible, large-scale dataset tailored for evaluating various tasks in Music Information Retrieval. The model training leverages the extensive resources of the FMA dataset.



Learn More

2023 - AI-Music - José Ángel López

LinkedIn - GitHub - Email

Figura C.8: Pantalla de selección de modelos y reproducción de audio.

**Visualización de resultados:** Sección donde el usuario puede ver el resultado de la predicción y una serie de características sobre la pista de audio con una breve descripción de cada una. La presentación de la información es tanto en modo texto como con gráficas.

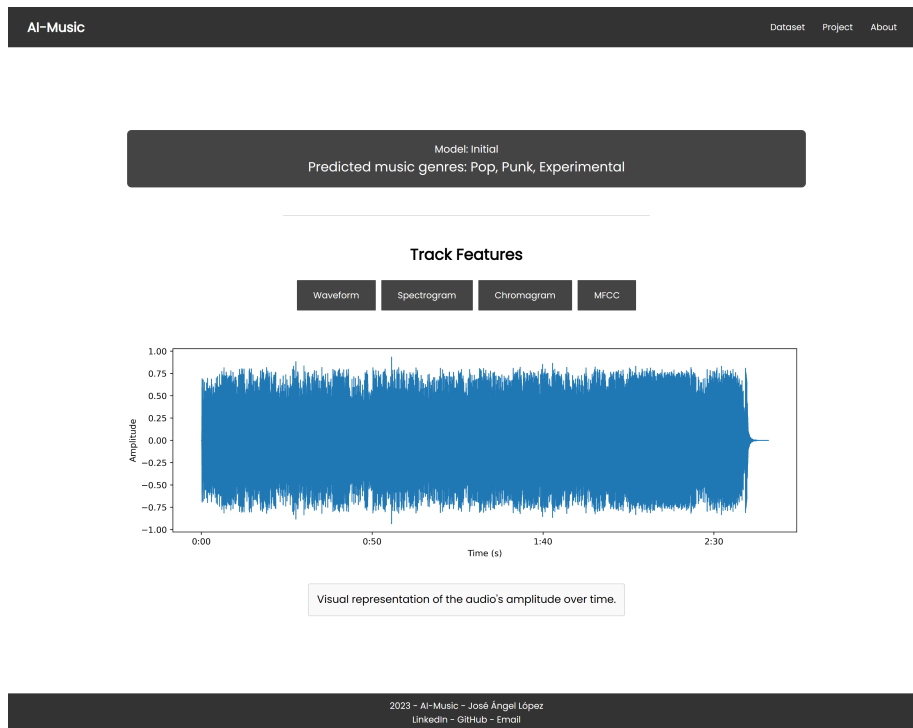


Figura C.9: Pantalla de visualización de resultados.





## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

Este apéndice aborda la documentación técnica de programación de la aplicación. En las siguientes secciones se detallan diferentes detalles importantes para trabajar en el proyecto, como pueden ser:

1. Estructura de directorios.
2. Instalación del entorno de desarrollo.
3. Obtención del código fuente.
4. Ejecución de la aplicación.
5. Pruebas realizadas.

## D.2. Estructura de directorios

La estructura de directorios de la aplicación es la siguiente:

- `/`: Directorio raíz del proyecto. Contiene el archivo `requirements.txt` con la información de las dependencias del proyecto, el archivo `README.md` con información relevante sobre el proyecto y el archivo `LICENSE` con la licencia del proyecto.
- `/app/`: Directorio con el código fuente de la aplicación. Además incluye en fichero `__init__.py` que lanza y define las rutas en la aplicación web utilizando `Flask`.

- `/app/models/`: Directorio donde se almacenan los modelos de *machine learning* entrenados que la aplicación utiliza para realizar las predicciones de los estilos musicales.
- `/app/src/`: Directorio donde se encuentra el código fuente Python. Aquí es donde se realiza todo el procesamiento interno de la aplicación.
- `/app/static/`: Directorio donde se almacenan los archivos estáticos utilizados en la interfaz web de la aplicación. Por ejemplo, archivos CSS, pistas de audio subidas o scripts de JavaScript.
- `/app/templates/`: Directorio donde se almacenan los archivos HTML que la aplicación utiliza para generar las páginas web.
- `/app/tests/`: Directorio donde se almacenan los tests unitarios y de integración utilizados para verificar el correcto funcionamiento de la aplicación.
- `/data/`: Directorio donde se almacenan los datos utilizados por la aplicación para realizar el entrenamiento o el procesado.
- `/data/processed/`: Directorio donde se almacenan los datos procesados y listos para ser entrenados en el modelo de machine learning.
- `/data/raw/`: Directorio donde se almacenan los ficheros de audio sin procesar.
- `/docs/latex/`: Documentación del proyecto en formato L<sup>A</sup>T<sub>E</sub>X.
- `/examples/`: Ejemplos de uso de las bibliotecas.

### D.3. Manual del programador

Esta sección tiene como objetivo ser una guía para que los futuros programadores puedan entender el código fuente, configurar el entorno de desarrollo y poder contribuir en el proyecto.

Para el desarrollo del proyecto se utilizó Visual Studio Code con el lenguaje de programación Python. Además, se utilizó Git para el sistema de control de versiones. Se recomienda tener ciertos conocimientos en estos sistemas antes de empezar.

Asimismo, para el diseño de la interfaz de usuario, se utilizó el lenguaje de marcado HTML, que proporciona la estructura básica de las páginas web. Para personalizar la apariencia de estas páginas, se aplicaron estilos mediante CSS y se implementaron funcionalidades interactivas a partir de JavaScript. Complementariamente, LaTeX se utilizó para la elaboración de la documentación, gracias a su capacidad para gestionar referencias y presentar ecuaciones y algoritmos de manera clara.

## Herramientas recomendadas

Para contribuir al desarrollo del proyecto se recomienda utilizar las siguientes herramientas.

- **Git**: Se trata del sistema de control de versiones utilizado para realizar un seguimiento de los cambios en el código fuente de la aplicación a lo largo del tiempo. Se puede combinar con GitHub. [24, 25]

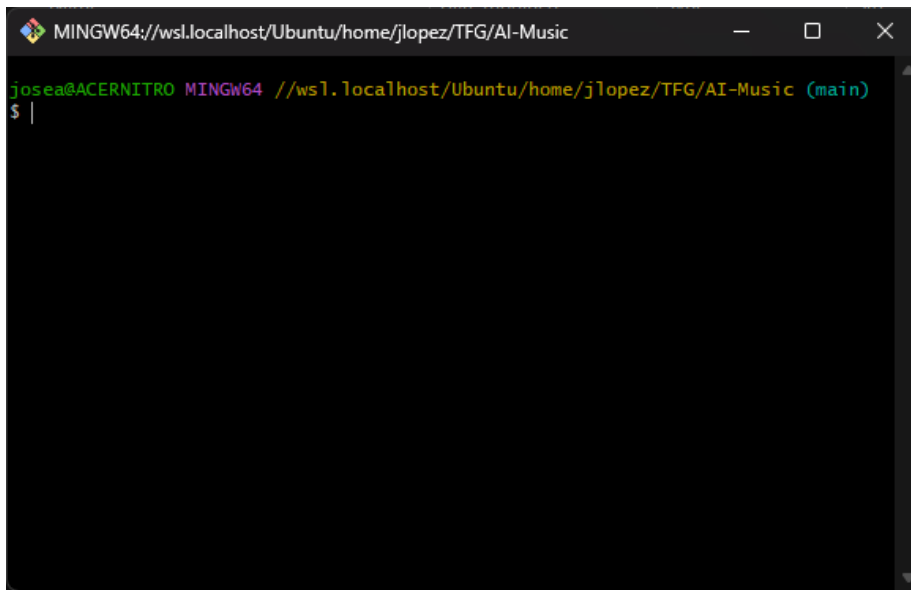


Figura D.1: Consola de Git.

- **Visual Studio Code**: Es el editor de código fuente recomendado para continuar con el desarrollo del proyecto. Se trata de un editor de código simple y ligero pero con una gran cantidad de extensiones que facilitan el desarrollo. Por ejemplo, incluye *linting* de código, formateo automático del código y soporte con Jupyter Notebooks o Docker [26].

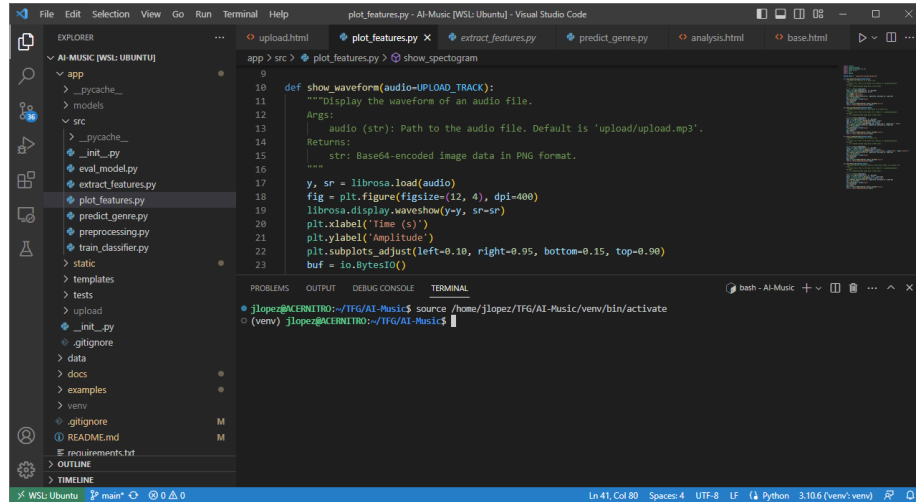


Figura D.2: Visual Studio Code.

- **Jupyter notebooks:** Herramienta de desarrollo de código y visualizaciones. Es una herramienta muy útil para realizar experimentación ya que une la ejecución de código fuente con la visualización de datos y la documentación. No se trata de una herramienta para realizar una versión final del producto, pero es muy útil para probar y generar prototipos de una forma rápida [27].

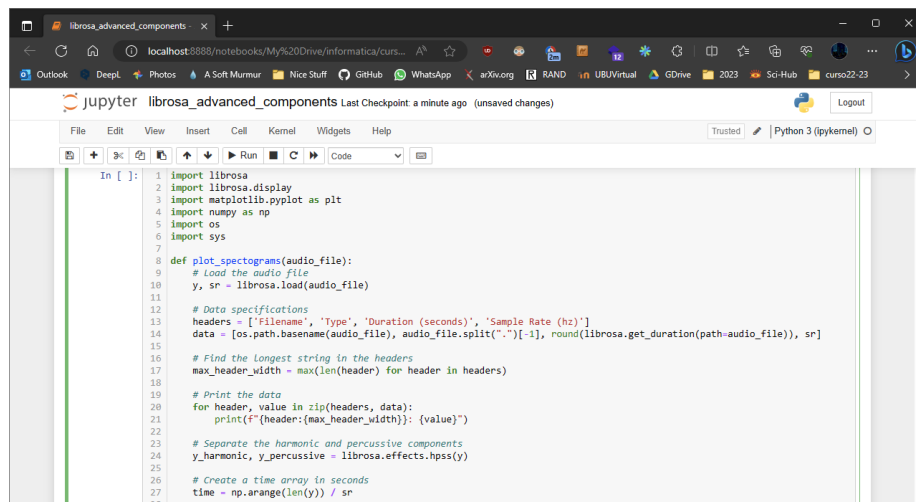


Figura D.3: Jupyter Notebook.

## D.4. Compilación, instalación y ejecución del proyecto

Para obtener el código fuente del proyecto los pasos a seguir son los siguientes:

- **Clonar el repositorio Git**
  - `git clone https://github.com/jle1001/AI-Music.git`
- **Crear un entorno virtual de Python:**
  - `python3 -m venv venv`
- **Activar el entorno virtual:**
  - Linux: `source venv/bin/activate`
  - Windows: `venv\Scripts\activate`
- **Instalación de dependencias**
  - `pip install -r requirements.txt`
- **Iniciar el servidor**
  - `flask run -debug`

Una vez que el servidor está en funcionamiento, se puede acceder a la aplicación web a través de un navegador entrando a la dirección `localhost:5000`.

## D.5. Pruebas del sistema

### Pruebas unitarias

Las pruebas unitarias son un tipo de pruebas de software en las que se comprueban individualmente las partes más pequeñas del sistema. Estas partes pueden ser funciones, métodos o clases. En este tipo de pruebas no se verifica el correcto funcionamiento del modelo de predicción, sino el funcionamiento de cada parte del código que se necesita para llegar hasta ahí. Por ejemplo:

- **Pruebas unitarias en la generación de predicciones:** verificación de que el uso del modelo entrenado genera una cantidad correcta de predicciones. Por ejemplo, se puede verificar que la capa de salida

del modelo genera una cantidad de clases igual al número de géneros musicales en el conjunto de datos.

```

1  import unittest
2  import librosa
3  import numpy as np
4  import tensorflow as tf
5
6  class TestModelPrediction(unittest.TestCase):
7      TEST_TRACK = './app/tests/test_audio.ogg'
8      def setUp(self):
9          # Load the trained model
10         self.model = tf.keras.models.load_model('app/models/model_8L_Ada.h5')
11         self.target_styles = [38, 15, 12, 10, 17, 25, 21]
12
13     def test_prediction_quantity(self):
14         y, sr = librosa.load(self.TEST_TRACK)
15         y = y[: (25 * sr)]
16         mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=10)
17         mfcc = tf.reshape(mfcc.ravel(), shape=(1, 1077, 10))
18         predictions = self.model.predict(mfcc)[0]
19         # Check that the number of genres predicted is correct.
20         self.assertEqual(len(predictions), max(self.target_styles) + 1)
21

```

Figura D.4: Código del test unitario que comprueba el número correcto de generación de predicciones.

```

1/1 [=====] - 1s 894ms/step
.
-----
Ran 1 test in 3.469s
OK

```

Figura D.5: Resultado de la ejecución del test unitario.

## Pruebas de interfaz de usuario

Las pruebas de interfaz de usuario son aquellas que se utilizan para verificar que la interacción entre el usuario y el software se realiza correctamente. Por ejemplo:

- **Prueba de carga de archivos:** esta prueba verifica que la aplicación puede cargar correctamente los archivos deseados por el usuario.

1. **El usuario carga el archivo:** 06. Give.mp3
2. **Resultado esperado:** se muestra en pantalla información básica sobre el fichero de audio, un menú de selección de modelos para realizar la predicción y un reproductor para escuchar el fichero de audio.

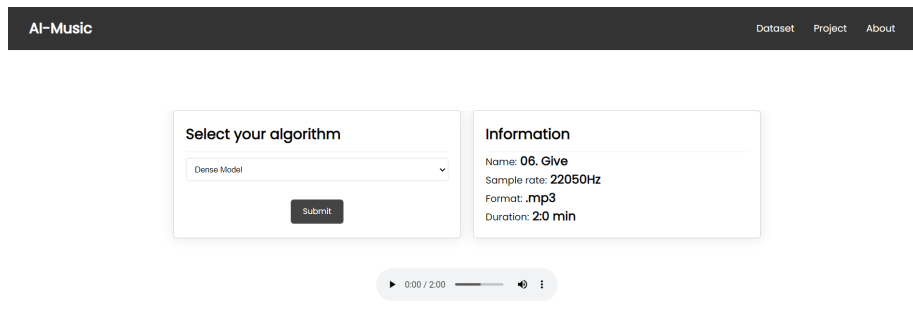


Figura D.6: Resultado obtenido. Es correcto.

- **Prueba de visualización de resultados:** esta prueba verifica que, una vez realizada la predicción, los resultados se muestran correctamente para el usuario.
  1. **El usuario presiona el botón *Submit***
  2. **Resultado esperado:** se muestra en pantalla una lista de tres predicciones con su probabilidad asociada, además de un menú interactivo donde se pueden ver distintas visualizaciones de las características del fichero de audio en concreto

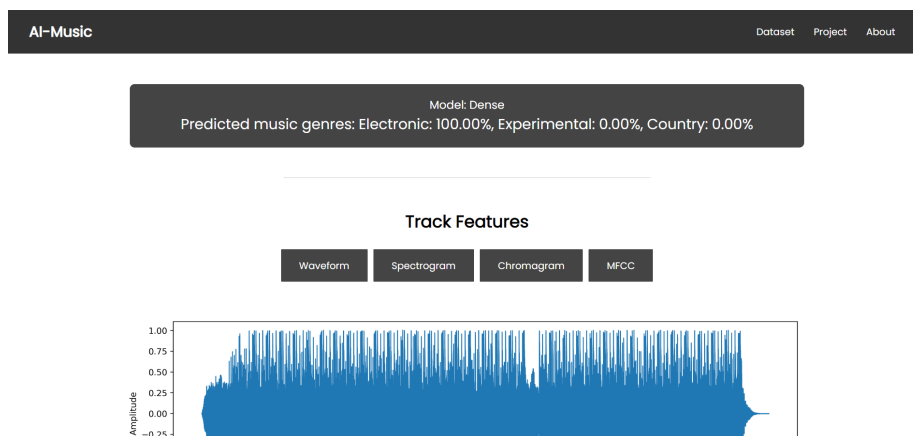


Figura D.7: Resultado obtenido. Es correcto.





## Apéndice *E*

---

# Documentación de usuario

---

### E.1. Introducción

Esta sección proporciona instrucciones de uso detalladas sobre cómo utilizar la aplicación, desde el proceso de instalación hasta el proceso de subida de archivos.

### E.2. Requisitos de usuarios

#### Entorno local

Requisito	Especificación
CPU	2.0 GHz o superior
RAM	4GB o superior
Disco duro	2GB de almacenamiento libre
Sistema Operativo	Compatible con Python 3.10 o superior
Python	Versión 3.10 o superior
Bibliotecas de Python	Especificadas en la sección de instalación

Tabla E.1: Requisitos para el funcionamiento local del sistema.

- **CPU 2.0 GHz o superior:** Un procesador de 2 GHz es capaz de manejar las instrucciones por segundo necesarias para realizar las tareas requeridas en esta aplicación. Como el procesamiento de audio y la ejecución de la predicción.

- **Memoria RAM de 4GB o superior:** Los 4GB de RAM son necesarios para que la aplicación funcione de manera eficiente sin ralentizaciones o bloqueos.
- **2GB de almacenamiento libre:** Esta cantidad de almacenamiento libre es la mínima necesaria para instalar la aplicación y sus bibliotecas necesarias.
- **Sistema operativo compatible con Python 3.10 o superior:** La aplicación está escrita en Python, por lo que se requiere un sistema operativo que pueda soportar la versión de Python 3.10 o superior.

### E.3. Instalación y ejecución

Para utilizar la aplicación en línea, se puede acceder al siguiente enlace: <https://jle1001.pythonanywhere.com>

Si se desea utilizar la aplicación localmente, se deben seguir los pasos indicados en el Apéndice D.4. Compilación, instalación y ejecución del proyecto.

### E.4. Manual del usuario

El objetivo de este manual es proporcionar una guía paso a paso para que el usuario pueda utilizar las funciones de la aplicación.

#### Subida de canciones

- En la ventana principal de la aplicación hay que pulsar el botón *Upload File*.
- Elegir el fichero de audio en el almacenamiento.
- Una vez subido el fichero de audio se darán opciones como: reproducir la canción, visualizar datos básicos y elegir el modelo para la predicción.

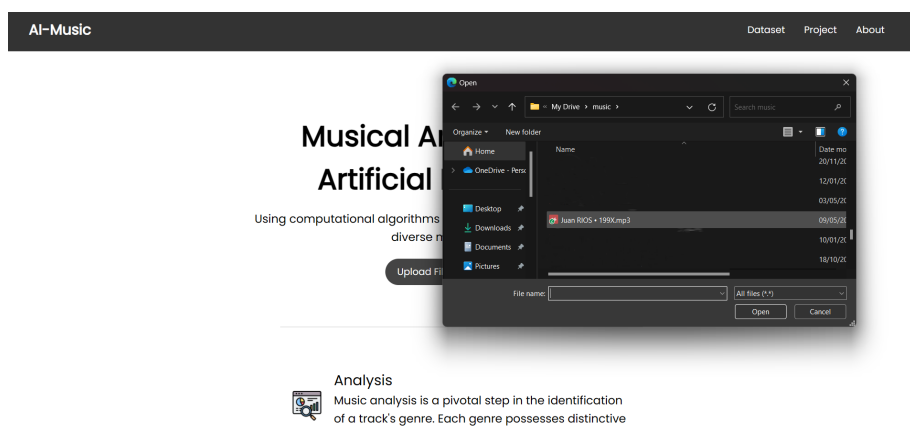


Figura E.1: Subida de canciones.

## Análisis y predicción

- En la ventana principal de la aplicación, pulsar el botón *Upload File*.
- Elegir el fichero de audio en el almacenamiento.
- Una vez subido el fichero de audio se darán opciones como: reproducir la canción, visualizar datos básicos y elegir el modelo para la predicción.
- Para realizar la predicción del género musical se debe elegir un modelo a utilizar y pulsar el botón *Predict*.

AI-Music


DatasetProjectAbout

Select your algorithm

Initial Model

Submit

Information


Name: Juan RIOS  cosmos


Sample rate: 22050Hz


Format: .mp3


Duration: 3:14 min

0:00 / 3:14











### Analysis

Music analysis is a pivotal step in the identification of a track's genre. Each genre possesses distinctive characteristics and features that set it apart.



### Classification

Artificial Intelligence algorithm steps in to categorize the song into its respective musical style based on the analyzed features.




### Insights

Data and graphical representations that portray the underlying structure and characteristics of the music.

## FMA Dataset

A publicly accessible, large-scale dataset tailored for evaluating various tasks in Music Information Retrieval. The model training leverages the extensive resources of the FMA dataset.

Learn More



2023 - AI-Music - José Ángel López  
LinkedIn - GitHub - Email

Figura E.2: Información sobre la canción subida.

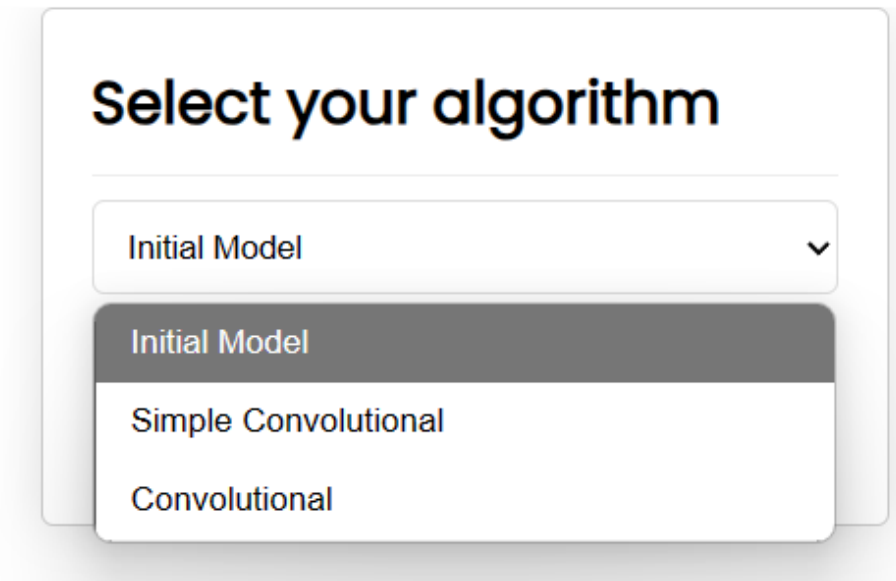


Figura E.3: Selección de algoritmo.

## Visualización del resultado

- En la ventana principal de la aplicación, pulsar el botón *Upload File*.
- Elegir el fichero de audio en el almacenamiento.
- Una vez subido el fichero de audio se darán opciones como: reproducir la canción, visualizar datos básicos y elegir el modelo para la predicción.
- Para realizar la predicción del género musical se debe elegir un modelo a utilizar y pulsar el botón *Predict*.
- Una vez el modelo haya terminado de predecir los estilos musicales, estos se mostrarán en la parte superior de la pantalla.
- En la parte central de la pantalla se podrán ver cuatro gráficos con diferentes características musicales.

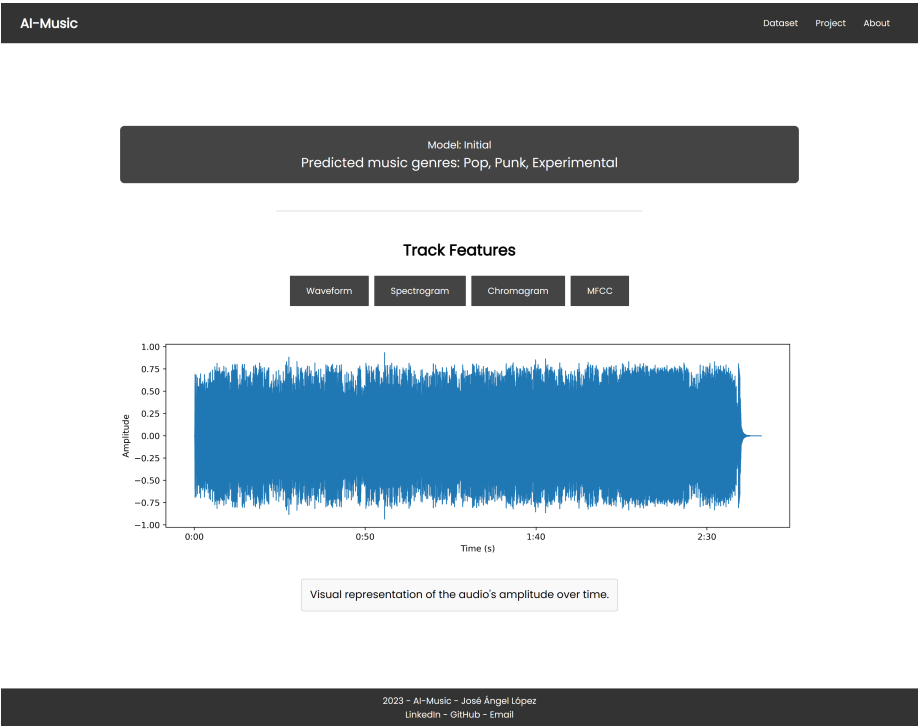


Figura E.4: Visualización del resultado: Onda de audio.

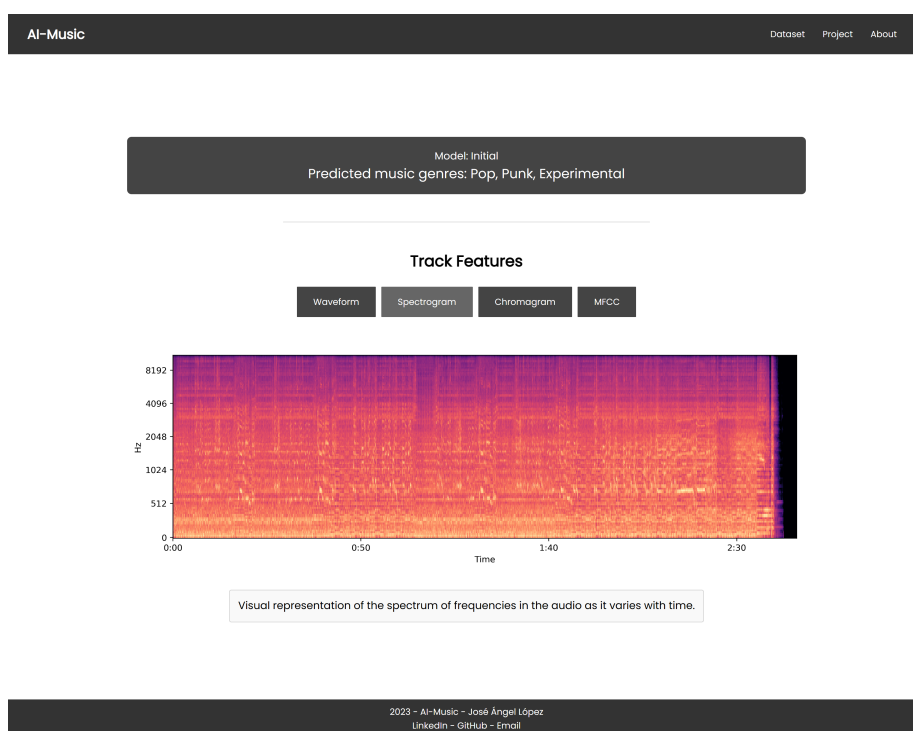


Figura E.5: Visualización del resultado: Espectrograma.

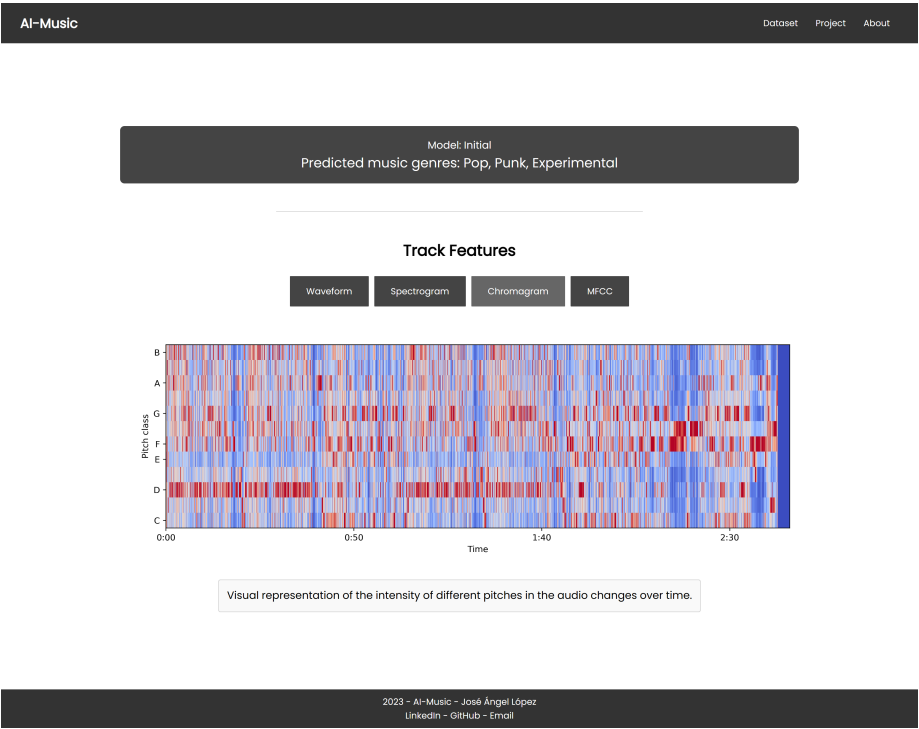


Figura E.6: Visualización del resultado: Cromograma



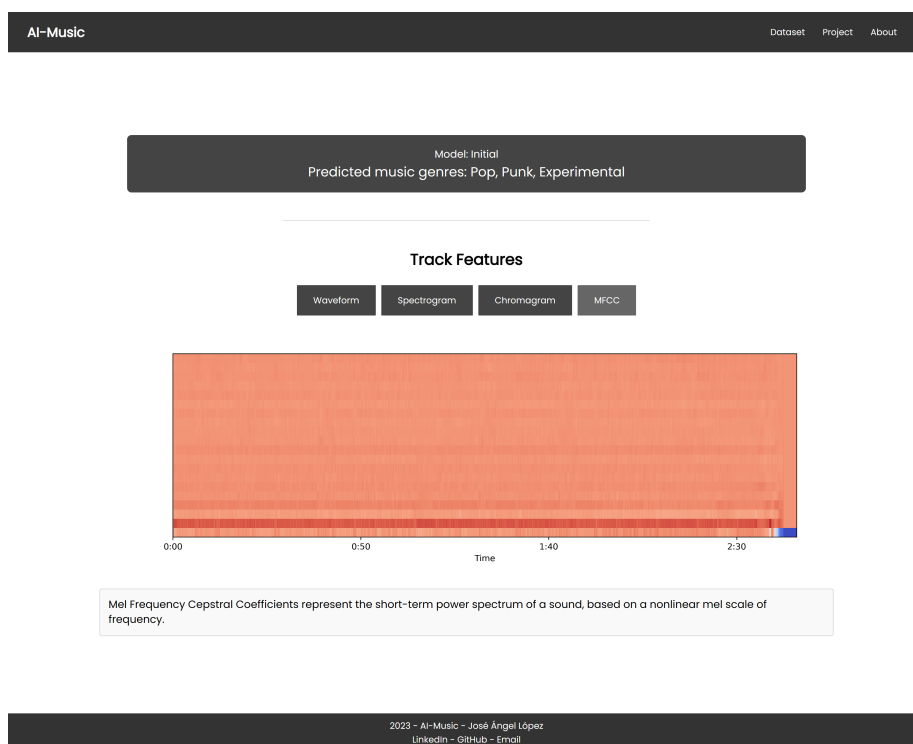


Figura E.7: Visualización del resultado: MFCC.

## Vídeos explicativos

Demostración de uso de la aplicación: [AI-Music - Demostración de funcionamiento](#)



---

## Bibliografía

---

- [1] “Amazon web services.” [Online]. Available: <https://aws.amazon.com/>
- [2] “Azure documentation.” [Online]. Available: <https://learn.microsoft.com/en-us/azure/?product=popular>
- [3] “Windows 11 home.” [Online]. Available: <https://www.microsoft.com/es-es/d/windows-11-home/dg7gmgf0krt0?rtc=1>
- [4] “Creative commons licenses,” 2023. [Online]. Available: <https://creativecommons.org/licenses/>
- [5] “The mit license,” Jun 2023. [Online]. Available: <https://opensource.org/license/mit/>
- [6] “Python 3.10 documentation,” 2023. [Online]. Available: <https://docs.python.org/3.10/>
- [7] “librosa: Audio and music signal analysis,” 2023. [Online]. Available: <https://librosa.org/doc/latest/index.html>
- [8] “Tensorflow: An end-to-end open source platform for machine learning,” 2023. [Online]. Available: <https://www.tensorflow.org/guide?hl=es-419>
- [9] “Scikit-learn: Machine learning in python,” 2023. [Online]. Available: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
- [10] “Welcome to flask - flask documentation (2.3.x),” 2023. [Online]. Available: <https://flask.palletsprojects.com/en/2.3.x/>
- [11] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A dataset for music analysis,” in *18th International Society for Music*

- Information Retrieval Conference (ISMIR)*, 2017. [Online]. Available: <https://arxiv.org/abs/1612.01840>
- [12] “Matplotlib 3.7.1 user’s guide,” 2023. [Online]. Available: <https://matplotlib.org/stable/users/index.html>
- [13] “Pandas 2.0.3 documentation.” [Online]. Available: <https://pandas.pydata.org/docs/>
- [14] “Numpy v1.25 manual.” [Online]. Available: <https://numpy.org/doc/1.25/>
- [15] R. Blog, “Requerimientos funcionales y no funcionales, ejemplos y tips,” Nov 2018. [Online]. Available: <https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a>
- [16] “csv — csv file reading and writing,” 2023. [Online]. Available: <https://docs.python.org/3/library/csv.html>
- [17] R. Graves, “Understanding the mp3 format.” [Online]. Available: <https://www.crutchfield.com/S-6bObTNjHeb6/learn/mp3-format-explained.html>
- [18] N. Selvaraj, “Python pickle tutorial: Object serialization,” Feb 2023. [Online]. Available: <https://www.datacamp.com/tutorial/pickle-python-tutorial>
- [19] O. Blancarte, “Arquitectura en capas.” [Online]. Available: <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas>
- [20] “Html tutorial,” 2023. [Online]. Available: <https://www.w3schools.com/html/default.asp>
- [21] MozDevNet, “Html: Lenguaje de etiquetas de hipertexto.” [Online]. Available: <https://developer.mozilla.org/es/docs/Web/HTML>
- [22] “Flaticon: Free vector icons,” 2023. [Online]. Available: <https://www.flaticon.es/>
- [23] “Poppins font,” 2023. [Online]. Available: <https://fonts.google.com/specimen/Poppins>
- [24] “Git: Distributed version control system,” 2023. [Online]. Available: <https://git-scm.com/>

- [25] “Github: Where the world builds software,” 2023. [Online]. Available: <https://github.com/>
- [26] Microsoft, “Documentation for visual studio code,” Nov 2021. [Online]. Available: <https://code.visualstudio.com/docs>
- [27] “Jupyter documentation.” [Online]. Available: <https://docs.jupyter.org/en/latest/>