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 — 26 de junio de 2023
Tutores: César Ignacio García Osorio, Alicia Olivares Gil

Índice general

Índice de figuras

Índice de tablas

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)

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

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

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

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

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

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

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

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

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

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

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

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

Sprint 14 (23/06/2023 - 30/06/2023)

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

Esta sección detalla los costes económicos a los que hay que hacer frente para diseñar, implementar y desplegar el software.

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 teníendo 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 o Microsoft Azure. El coste puede variar según el uso necesario. En este caso ficticio se estima que el gasto será de 100€.

Concepto	**Coste**
Windows 11 Home	145€
Computación en la nube	100€
Total	**245€**

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€
Total (mensual)	**435€**
Total (6 meses)	**2610€**

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	35€
Otors costes	2610€
Total	**2695€**

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 ofrecer 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 económica 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.

INSERTAR TABLA CON LICENCIAS DE CADA PARTE DEL SOFTWARE.

La licencia utilizada se corresponde con la licencia más general que sea compatible con el resto.

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. Catalogo 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. A continuación se listan los requisitos funcionales extraidos 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. A continuación se listan los requisistos 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 este 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

Insertar diagrama general de casos de uso.

Casos de uso

CU-1	**Carga de la canción**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-1.1, RF-3.1
Descripción	El usuario utiliza la función de carga de archivo para seleccionar y cargar una canción en la aplicación.
Precondiciones	1. La aplicación se encuentra en funcionamiento. 2. El usuario tiene acceso a los archivos de música que desea cargar.
Acciones	1. El usuario selecciona la opción para cargar una canción pulsando el botón *Upload File*. 2. El usuario busca y selecciona la canción que desea cargar desde su almacenamiento local. 3. La canción se carga en la aplicación.
Postcondiciones	1. La canción se encuentra cargada en la aplicación. 2. La aplicación está lista para extraer características y realizar el proceso de predicción de la canción cargada.
Excepciones	1. El archivo cargado no es un fichero de audio. 2. La canción tiene un formato incompatible.
Importancia	Alta

Tabla B.1: CU-1 Carga de la canción

CU-2	**Extracción de características de audio.**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-1.1
Descripción	La aplicación extrae las características relevantes del audio para su posterior análisis.
Precondiciones	1. La canción se ha cargado correctamente en la aplicación.
Acciones	1. La aplicación inicia el proceso de extracción de características del archivo de audio cargado.
Postcondiciones	1. Las características relevantes de la canción se han extraído correctamente. 2. La aplicación está lista para utilizar las características extraídas para realizar la predicción del estilo musical.
Excepciones	1. Ocurrió un error durante la extracción de las características del audio.
Importancia	Alta

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

CU-3	**Ejecución del modelo de machine learning**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-1.2
Descripción	La aplicación ejecuta el modelo de machine learning para realizar la predicción del estilo musical según las caracterísiticas extraídas.
Precondiciones	1. Las características de audio de la canción se han extraído correctamente. 2. Las características de audio de la canción son suficientes para realizar el proceso de predicción.
Acciones	1. El modelo procesa las características extraídas de la canción. 2. Se obtiene el resultado de la predicción, el cual consiste en un vector de probabilidades entre diferentes estilos musicales.
Postcondiciones	1. La aplicación ha generado una predicción sobre el estilo musical de la canción. 2. La aplicación está lista para mostrar los detalles de la predicción al usuario.
Excepciones	1. El modelo no puede procesar las características extraídas.
Importancia	Alta

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

CU-4	**Visualización de la predicción**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-2.1, RF-2.2
Descripción	Tras la ejecución del modelo de machine learning, la aplicación muestra la predicción del estilo musical al usuario.
Precondiciones	1. La aplicación ha generado una predicción.
Acciones	1. La aplicación muestra la predicción del estilo musical en la interfaz. 2. El usuario puede cambiar entre distintas visualizaciones de forma dinámica. 3. El usuario visualiza la predicción.
Postcondiciones	1. El usuario ha obtenido la predicción del estilo musical de la canción.
Excepciones	1. Se produce un error al intentar mostrar la predicción.
Importancia	Alta

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

CU-5	**Reproducción de la canción**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-1.2, RF-1.3
Descripción	La aplicación permite al usuario reproducir la canción cargada.
Precondiciones	1. La canción ha sido cargada correctamente en la aplicación.
Acciones	1. El usuario selecciona la opción de reproducir la canción mediante el botón *play* del reproductor. 2. La aplicación reproduce la canción cargada.
Postcondiciones	1. La canción está siendo reproducida en la aplicación.
Excepciones	1. Se produce un error en la reproducción de la canción.
Importancia	Media

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

CU-6	**Carga de una nueva canción**
Versión	1.0
Autor	José Ángel López
Requisitos asociados	RF-1.1, RF-1.3
Descripción	El usuario puede cargar una nueva canción sin tener que cerrar o reiniciar la aplicación.
Precondiciones	1. La aplicación ha realizado la predicción de una canción.
Acciones	1. El usuario hace selecciona el nombre de la aplicación, en la zona superior izquierda de la pantalla, para volver a la página inicial. 2. El usuario pulsa el botón *Upload File*. 3. El usuario busca y selecciona la nueva canción que desea cargar desde su almacenamiento.
Postcondiciones	1. La nueva canción se carga en la aplicación.
Excepciones	1. El archivo cargado no es un fichero de audio. 2. La canción tiene un formato incompatible.
Importancia	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.

C.2. Diseño de datos

La aplicación trabaja con distintos tipos de datos.

- **Archivos CSV**:
- **Archivos de audio**:
- **Archivos pickle**:

INSERTAR DIAGRAMA DE CLASES, RELACIONES, E/R, ETC.

C.3. Diseño procedimental

DISEÑO PROCEDIMENTAL SIGNIFICA CÓMO SE HAN DISEÑADO LOS PROCEDIMIENTOS DE LA APLICACIÓN (CÓMO LOS MÉTODOS, FUNCIONES, ETC.) MOSTRAR CADA PROCEDIMIENTO EN PSEUDOCODIGO O LENGUAJE NATURAL. diagramas de interacción, secuencia, etc.

C.4. Diseño arquitectónico

MODELO ARQUITECTONICO PREVISTO, PATRONES DE DISEÑO. Por ejemplo decir si hemos usado patrón vista controlador, patrón MVM, etc. herencias, genericidad, clases, etc.

EXPLICAR LA FORMA GENERAL DE LA ARQUITECTURA DE LA APLICACIÓN (CON DIAGRAMAS, FLECHAS, ETC.)

INSERTAR NUEVA SECCIÓN CON DISEÑO DE INTERFACES SI PROCEDE.

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 LaTeX.
- `/examples/`: Ejemplos de uso de las librerías.

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.

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 `localhost:5000`.

D.5. Pruebas del sistema

Docuemntar tests realizados, captura de pantalla con los tests en verde, captura del buen funcionamiento de los componentes, etc.

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**:

- **2GB de almacenamiento libre**: Esta cantidad de almacenamiento libre es la mínima necesaria para instalar la aplicación y sus librerías 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

Como instalar el programa en el pc o como acceder a la página web etc.

E.4. Manual del usuario

Descripción de como usar diferentes funcionalidades de la aplicación.

Como subir canciones. Como analizar canciones. Ver características. Cambiar algoritmo ETC.