Generación Automática de Canciones mediante Modelos de Lenguaje Natural y Redes Neuronales

Alejandro Román González Jurado

Campus Cámara Sevilla

1.Introducción

La música, según la definición tradicional del término, es el arte de crear y organizar sonidos respetando los principios fundamentales de la melodía, armonía y el ritmo, con el objetivo de expresar un sentimiento y generar una emoción en otras personas. Ha evolucionado mucho desde su origen en la antigua Grecia, uniéndose con otras artes como la danza o la poesía. La música se ha extendido en muchos géneros distintos, y de formas en las que cualquier persona o grupo puede realizar siempre que tenga un sentido coherente. Para encontrar ese sentido, es necesario entender cómo funcionan conceptos como la armonía, las melodías o el ritmo, conocimientos de difícil acceso para cualquier persona.

1.1 Problema a resolver

En mi opinión, esta barrera de aprendizaje mencionada hace que gran parte de la sociedad se olvide de intentar mostrar su creatividad y prefiere escuchar las de otras personas. El objetivo perseguido en este proyecto es intentar romper esa barrera de aprendizaje mediante un modelo de inteligencia artificial que pueda generar canciones basándose en unas instrucciones proporcionadas por una persona. El objetivo no es cambiar cómo se crea la música, sino darle voz a todo aquel que no pueda utilizarla para expresarse.

1.2 Contexto del proyecto

La generación de contenido musical automatizado es un desafío en el campo de la inteligencia artificial. Los modelos de generación de lenguaje natural han permitido generar texto coherente, sobre todo tras la creación de los Transformers, tras la publicación del artículo “Attention Is All You Need”[1] por el grupo de científicos de Google. Uno de los desafíos que pueden presentar estos modelos es la ausencia de creatividad, que es algo que no se puede entrenar. De ahí, que los poemas o letras de canciones escritos por IA pueden no ser los mejores. En este campo, el objetivo fijado será entrenar un modelo de generación de texto con gran cantidad de letras musicales para comprobar la calidad de las prestaciones que puede proveer la IA en cuestiones de generación de letras.

Por otro lado, ya se conocen varios modelos cuyo objetivo es generar melodías de música basándose en unas instrucciones introducidas (prompt). Estos modelos entrenados con grandes cantidades de sonido han demostrado que pueden llegar a dar unos resultados excelentes en la creación de música. Sin embargo, no existen muchos modelos de generación de música que puedan proveer al usuario de una letra.

Si el objetivo de la inteligencia artificial es reproducir las tareas humanas de la forma más realista posible, estos dos procesos deberían ir en conjunto, ya que mejora la concordancia musical y los compositores de música siempre lo han realizado de esta forma.

2. Objetivos

En esta sección, se especificarán los objetivos, tanto generales como específicos, que posteriormente se explicarán con más detalle en el marco teórico.

❖ General:

➢ Integración de un modelo NLP de generación de texto y un modelo Text-to-Music mediante el uso de técnicas de fine-tuning para la creación conjunta de música.

❖ Específicos:

➢ Implementar un modelo de generación de texto para la generación de letras de canciones.

➢ Utilizar un modelo de generación de melodías pre entrenado para la composición musical.

➢ Aplicar técnicas de fine-tuning en el modelo de generación de texto con datasets que contengan géneros de música concretos para mejorar su rendimiento.

➢ Desarrollar una interfaz web que facilite la interacción del usuario con la aplicación.

➢ Evaluar la calidad de las letras y melodías generadas.

3. Marco Teórico

En este apartado, se explicarán las bases conceptuales y teóricas en las que se basará el proyecto.

3.1 Conceptos Generales

En la redacción de este proyecto se han repetido en varias ocasiones conceptos como “modelo”, “redes neuronales” o “LLM”, entre otros. Estos términos se van a explicar más profundamente para facilitar el entendimiento de la estructura del proyecto.

- **Modelo:** La definición de un modelo de inteligencia artificial según IBM es “*un programa que ha sido entrenado en un conjunto de datos para reconocer ciertos patrones o tomar ciertas decisiones sin más intervención humana. Los modelos de inteligencia artificial aplican distintos algoritmos a las entradas de datos relevantes para lograr las tareas, u outputs, para los que han sido programados*”. [2]

Existen distintos tipos de modelos en inteligencia artificial; los más modernos son aquellos cuya estructura está basada en redes neuronales, que intentan imitar la estructura de un cerebro humano. También se conoce como *DeepLearning*.

- **LLM (Large Language Model):** Continuando la explicación anterior, un lenguaje de gran tamaño (LLM) es un modelo de aprendizaje profundo (DeepLearning) que ha sido entrenado con inmensas cantidades de datos, permitiendo así que estos modelos puedan comprender y generar lenguaje natural, junto con otros tipos de contenidos para realizar una amplia variedad de tareas. Algunos ejemplos prácticos de aplicaciones que usan estos modelos son ChatGPT, Gemini, Perplexity… [3]

El entrenamiento de estos modelos requiere de unos recursos de hardware muy elevados, lo que supone una gran barrera para la mayoría de los desarrolladores de IA.

En este proyecto, se van a implementar dos modelos; un primero de generación de texto automático (NLP) y otro de Text-to-music.

- **NLP (Natural Language Processing):** El procesamiento del lenguaje natural es una rama de la inteligencia artificial que ayuda a las computadoras a entender, interpretar y manipular el lenguaje humano. NLP toma elementos prestados de muchas disciplinas, incluyendo la ciencia de la computación y la lingüística computacional, en su afán por cerrar la brecha entre la comunicación humana y el entendimiento de las computadoras. [4]

- **Text-to-Music:** Los modelos Text-to-Music tienen como objetivo generar piezas melódicas de gran calidad basándose en unas instrucciones introducidas, tanto en mono como en estéreo, que se evaluarán de forma empírica por parte del usuario. Este tipo de modelos no se entrenan con datos tradicionales, sino que se entrenan con grandes cantidades de audio en los que el modelo puede reconocer patrones para aprender y generar sus propias melodías. [5]

3.2 Conceptos Específicos:

En este subapartado se van a razonar ideas más concretas del proyecto.

- **Implementación de un modelo de generación de texto para la creación de letras de canciones.**

Este objetivo del proyecto consiste en investigar sobre qué distintos modelos NLP existen y cuál puede ser el qué mejor se adecúe a nuestro objetivo. Se conoce que realizar un entrenamiento por completo de un modelo es muy costoso en cuanto a recursos se refiere. Para ello, se aplicarán técnicas que se explicarán más adelante sobre modelos ya entrenados para que puedan ser usados con fines específicos de este proyecto.

Todos los modelos de NLP se basan en la estructura "*Transformers*" [1]. El término "*Transformer*" en IA se refiere a un tipo de arquitectura de red neuronal que utiliza técnicas de aprendizaje profundo y autoatención para manejar secuencias de datos, proporcionando una mejora significativa en tareas como la traducción automática, la generación de texto y la comprensión del lenguaje.

En los últimos años, el campo de la Inteligencia Artificial (IA) ha presenciado una revolución tecnológica que ha cambiado la forma en que las máquinas entienden y procesan el lenguaje humano. Esta revolución viene de la mano de los modelos ya mencionados Transformers, que han establecido nuevos estándares en el Procesamiento del Lenguaje Natural (NLP).

Como se ha mencionado anteriormente, el entrenamiento completo de un modelo de inteligencia artificial con Transformers es muy costoso computacionalmente. Para solucionar este problema y que los modelos de DeepLearning sean utilizables por todo el mundo, la plataforma HuggingFace [6] desarrolló una librería en lenguaje Python [7], apodada como Transformers [8], que permite guardar los modelos ya entrenados para poder utilizarlos sin necesidad de entrenarlos.

Tras realizar una investigación, los modelos NLP que pueden utilizarse en este proyecto son:

- ★ **Llama:** Un modelo NLP open-source desarrollado por Meta AI [9]. Dispone de varias versiones, la mayoría disponibles de forma gratuita. Actualmente se encuentra en su versión 4 , cuyos requisitos para utilizarse son muy elevados y tiene algunas funcionalidades de pago. Otras versiones más antiguas, como la v2 [10] o v3 [11] , tienen aptitudes menores pero pueden estar suficientemente capacitadas para el proyecto.

Se han investigado algunos modelos realizados con Llama que tienen un fin similar al objetivo buscado en este proyecto. Estos son:

- **LyricalLlama:** Un modelo basado en LLama v2 [10] diseñado para generar letras en inglés condicionadas por el nombre de una canción y un artista específico. [12]

  Como ventajas de este modelo, se ha encontrado que el modelo funciona con instrucciones estructuradas, es decir, se introduce el nombre del artista y el título de la canción. Asimismo, esta ventaja proporciona otra ventaja, que se trata de la coherencia temática que se lograda debido a vincular títulos con su contenido lírico.

  Por otro lado, una de las desventajas que ha demostrado es que requiere una gran cantidad computacional debido al tamaño del modelo Llama. Además, solo genera letras en inglés y sobre artistas que se encuentren en el dataset.

- **Lorca-LLama3-8B-GGUF:** Adaptación del modelo LLama 3 [11] para generar versos poéticos en español emulando el estilo de Federico García Lorca. [13]

  Uno de los aportes claves de este proyecto es la captura de métricas y simbolismos. El modelo aprende los recursos y patrones lorquianos como las rimas, estrofas o metáforas. Las letras musicales están muy ligadas a la poesía y utilizan muchos recursos de ella, incluso de forma muy concurrente, suelen ser poemas coordinados con melodías musicales. Otra ventaja aportada por el modelo es que tiene un control del estilo de la letra generada mediante tokens. Por ejemplo: [Estilo:Surrealismo]. De esta forma permite mayor libertad al usuario en la generación de letras.

  En cuanto a las desventajas, el modelo está especializado en poesía, lo que implica que si el usuario busca un estilo más moderno, el desempeño del modelo bajará considerablemente. También está especializado en la poesía lorquiana, lo que implica que los estilos son limitados.

★ **GPT:** (Generative Pre-trained Transformer) es una tecnología de inteligencia artificial basada en el aprendizaje profundo, que permite generar textos de forma automática [14]. Desarrollado por OpenAI en 2018, el GPT revolucionó el campo del procesamiento del lenguaje natural. [15] Actualmente, la última versión oficial de GPT es la 4.1, muy potente pero su API es de pago. Existen varias versiones con capacidades muy interesantes pero a partir de la versión 3, no son open-source [16]. La última versión open-source es el GPT-2. [17]

La versión GPT-2 es una versión que está obsoleta en la actualidad para el tipo de servicio que quiere proveer OpenAI, pero para ciertos proyectos especializados es todavía un modelo muy útil. En la plataforma HuggingFace [6], podemos encontrar varios modelos entrenados de GPT-2 utilizables. [18]

Algunos proyectos relacionados con este proyecto que han utilizado GPT-2 son:

- **TheBeatles:** Modelo GPT-2 ajustado para generar letras en el estilo de The Beatles, usando un dataset de 200 canciones de la banda. El objetivo del modelo es muy similar al del proyecto propuesto.

  Una gran ventaja es la optimización de los recursos. El modelo funciona en GPUs más modestas gracias al tamaño reducido del modelo GPT-2. Asimismo, genera las estrofas estructuradas (versos, coros, puente) con temáticas recurrentes, como el amor, revolución…

  Por otro lado, el mayor problema de este modelo reside en que el conjunto de datos utilizado es muy pequeño, por tanto, la diversidad de salidas será muy reducida.

Existen mucho más tipos de modelos, pero no han sido mencionados en esta investigación debido a que están más centrados en otras aplicaciones del NLP, como el análisis de sentimientos o la traducción de idiomas. Algunos son BERT, QWen o XL.

Finalmente, el proyecto estará basado en un modelo GPT-2. Uno de los principales motivos de esta decisión es la capacidad de aportar una alta eficiencia en unos recursos computacionales más reducidos unido a la flexibilidad que aporta el modelo. Existen varias versiones pre entrenadas según las capacidades buscadas. En este proyecto, se utilizará la versión “medium” [19].

- **Uso de un modelo de generación de audio para la creación de melodías.**

En los últimos años, hemos sido testigos de una revolución sin precedentes en el ámbito del audio, impulsada por avances significativos en la tecnología de la inteligencia artificial. El audio digital, en sus inicios, requería de equipos especializados y conocimientos técnicos avanzados para su uso, lo que limitaba su acceso a profesionales y estudios bien equipados. Al igual que con los modelos NLP, existen distintos modelos entrenados de distintas formas para la generación de audio, y cada uno de ellos están enfocados en distintos objetivos, como la replicación de voces o la generación de melodías.

Se ha realizado una investigación sobre los modelos más conocidos para la generación de audio:

- **WaveNet:** Desarrollado por DeepMind, WaveNet es una red neuronal profunda que genera audio a nivel de muestra modelando directamente las formas de onda, permitiendo la síntesis de voces humanas y música con una naturalidad sin precedentes, superando a los sistemas de texto a voz tradicionales [20]. Este modelo supuso una gran revolución en cuanto a la generación de audio, debido a que otros modelos de generación de audio se basaron en él para llevar a cabo otros proyectos.

  Este modelo tiene una arquitectura autoregresiva, es decir, predice cada muestra de audio basándose en las anteriores, resultando en una calidad de sonido altamente realista [21].

- **NSynth:** Un modelo que utiliza redes neuronales profundas para generar sonidos a nivel de muestras individuales. Al aprender directamente de los datos, NSynth ofrece a los artistas un control intuitivo del timbre y la dinámica, así como la capacidad de explorar nuevos sonidos que serían difíciles de producir con un sintetizador afinado manualmente [22].

  NSynth nace de Magenta, un proyecto de investigación de Google que explora la forma en la que la inteligencia artificial puede ayudar al arte. Las cualidades acústicas del instrumento aprendido dependen tanto del modelo aprendido como de los datos de entrenamiento disponibles. NSynth ha mejorado todo lo posible estas dependencias, creando un conjunto de datos que contiene una gran colección de aproximadamente 300000 notas musicales muestreadas de aproximadamente 1000 instrumentos, un orden de magnitud mucho mayor que otros conjuntos de datos disponible

públicamente. Para mejorar el modelo, han creado un nuevo modelo de autocodificador basado en WaveNet que aprende códigos que representan de forma significativa el espacio de los sonidos de los instrumentos.[23]

- **MusicGen:** Un potente modelo desarrollado por Meta que redefine los límites de la generación musical condicional creando música de alta calidad basándose en descripciones de texto o melodías [24]. Se trata de un modelo muy útil que simplifica el uso de la generación de música con IA.

  Se compone de un transformador de una sola etapa junto con patrones eficientes de entrelazado de tokens, eliminando la necesidad de conectar múltiples modelos en cascada, de forma jerárquica o mediante sobremuestreo. Establece la evaluación empírica como predeterminada, es decir, el humano será el que decida si la música es de su gusto o no [5].

  A su vez, Meta desarrolló una librería de código abierto conocida como Audiocraft diseñada para el procesamiento y generación de audio con inteligencia artificial [25].

  Se trata de una librería muy revolucionaria debido a que engloba 3 modelos de inteligencia artificial que trabajan en conjunto para producir música y sonidos de alta calidad a través de textos. Estos modelos son MusicGen, AudioGen y Encodec. Tanto MusicGen como AudioGen se focalizan en capturar patrones de sonido y generar audio de calidad, mientras que Encodec convierte la onda sonora en tokens de audio, permitiendo controlar la generación de audio aplicar modelos de condicionamiento para adaptar el resultado final a las necesidades específicas, como aplicaciones de texto a audio. [26]

  El hecho de que esté desarrollado como una librería y que proporcione códigos de entrenamiento compatibles con las librerías principales de desarrollo de inteligencia artificial como PyTorch hace que los usuarios puedan desarrollar sus propios modelos de audio generativos, siguiendo las pautas y principios de diseño de Audiocraft.

Como conclusión, el proyecto utilizará Audiocraft para la parte de generación de audio.

- **Aplicación de técnicas de fine-tuning en el modelo de generación de texto con datasets que contengan géneros musicales concretos para mejorar su rendimiento.**

Durante la redacción del proyecto, se ha reiterado lo computacionalmente costoso que resulta entrenar un modelo por completo. La importancia de poder usar modelos pre entrenados libera al usuario de la necesidad de tener unas condiciones computacionales altísimas. El fine-tuning es el siguiente complemento disponible para los usuarios para facilitar el uso de grandes modelos de IA en tareas más específicas.

El fine-tuning es el proceso de adaptar un modelo previamente entrenado para tareas o casos específicos. Se trata de una técnica de transfer-learning (aprendizaje por transferencia), es decir, la práctica de aprovechar el conocimiento que un modelo existente ya ha aprendido como punto de partida para aprender nuevas tareas [27].

Esto junto con la aparición de comunidades online de IA como HuggingFace o Kaggle, ha permitido que cualquiera pueda realizar y publicar un modelo para tareas específicas basado en un modelo de aprendizaje anteriormente entrenado.

Generalmente, el fine-tuning trata de un segundo entrenamiento de un modelo, previamente entrenado con grandes cantidades de datos más generales, con un conjunto de datos más pequeño de ejemplos que reflejan de manera más directa las tareas y los casos de uso específicos para los que se utilizará el modelo.

Por tanto, los resultados de una técnica de fine-tuning dependen casi exclusivamente de la calidad del conjunto de datos que se va a aplicar. En consecuencia, para que este proyecto aumente sus prestaciones, se necesitará de un buen conjunto de datos que contenga letras de canciones para que el modelo GPT-2 escogido previamente pueda especializarse en la tarea buscada.

Tras realizar una búsqueda de datos sobre canciones, se ha encontrado un conjunto de datos estructurados público con el título de más de 5 millones de canciones, sus letras, su género, el nombre del artista o grupo que la compuso, el año de publicación, el número de visualizaciones y el idioma en el que están escritas [28]. Este conjunto de datos está sacado de Genius, una página web que contiene las letras de millones de canciones, además de datos sobre ellas que pueden ser escritos y confirmados por la comunidad [29].

Para resumir y concluir este apartado, se realizará un preprocesamiento al conjunto de datos para aplicar técnicas de fine-tuning y obtener uno o varios modelos especializados en la generación de letras.

- **Desarrollo de una interfaz web que facilite la interacción del usuario con la aplicación.**

Todas las aplicaciones o dispositivos disponen de una parte técnica que solo el personal con el conocimiento sabe controlar, pero el usuario final no tiene porqué conocer el funcionamiento interno de una aplicación, solamente necesita saber cómo utilizarla.

La parte interna de una aplicación, aquella que solo el desarrollador conoce y tiene acceso, es denominada como **BackEnd**. Por otro lado, la interfaz que el usuario ve e interactúa cuando utiliza la aplicación es conocida como **FrontEnd**. En este proyecto, todo lo mencionado anteriormente, el modelo de generación de texto o de melodías, constituirá el BackEnd, y en este apartado será necesario desarrollar una interfaz que funcione como FrontEnd, es decir, que haga de nexo de unión entre el usuario y el código final, simplificando al máximo todo lo necesario para que el usuario pueda utilizar la aplicación.

Los modelos de IA se van a desarrollar todos en lenguaje Python, un lenguaje moderno y muy versátil. Para continuar con el mismo lenguaje en el desarrollo del FrontEnd, se utilizará Flask [30].

Flask es un framework (marco de trabajo) que proporciona una base y un conjunto de herramientas para el desarrollo de proyectos web en lenguaje Python. Se caracteriza por su sencillez para el desarrollo de webs, su estructura flexible y el soporte para extensiones y complementos que hacen que Flask pueda ser una increíble herramienta para desarrollar una aplicación web [31].

El funcionamiento de Flask está basado en el uso de rutas que dispondrán de diferentes funciones que se encargarán de ejecutarlas. Además, una de las mayores virtudes de Flask es que se centra en resolver las solicitudes HTTP de una manera ágil para que el usuario pueda navegar de forma rápida y sencilla por la aplicación. Serán necesarios algunos conocimientos de programación normales de HTML y CSS, el lenguaje estándar para diseñar páginas web y el lenguaje que permite animarlas, respectivamente.

- **Evaluación de las letras y melodías generadas.**

Una de las mayores facultades que tiene el arte es que no dispone de una métrica que pueda medir la calidad de un poema, un cuadro, una canción o una escultura. Se trata de algo completamente subjetivo. Gran parte de las elecciones realizadas durante las investigaciones de modelos, tanto de melodías como de letras, era que la comprobación de la calidad fuese de forma empírica. En otras palabras, el usuario será el que decida si le gusta la letra y la melodía. Para ello, se pondrá a disposición del usuario la posibilidad de repetir la generación tanto de la letra como de la melodía hasta que encuentre una que sea de su agrado.

4. Marco Metodológico

En esta sección se detallarán los métodos, procedimientos, herramientas y técnicas empleadas para llevar a la práctica toda la investigación realizada previamente en el marco teórico.

4.1 Entorno de desarrollo

Primero, se expondrán todos los detalles del entorno, tanto software como hardware, en el que se ha realizado el proyecto.

4.1.1 Hardware

El equipo disponible para desarrollar el proyecto tiene las siguientes características:

- **Procesador:** AMD Ryzen 7 5800X (8 núcleos / 16 hilos)
- **Memoria RAM:** 16 GB DDR4
- **Tarjeta gráfica:** NVIDIA GeForce RTX 3060 (12 GB VRAM)
- **Almacenamiento:** SSD 1TB
- **Sistema operativo:** Windows 11 (64 Bits)

Este hardware ha sido fundamental para llevar a cabo ciertas tareas, como el uso de redes neuronales profundas que requieren de grandes cantidades de recursos computacionales. Hay que recalcar la importancia de la tarjeta gráfica (GPU), debido a que los procesos de entrenamiento de modelos de IA, aunque se trate de fine-tuning, requieren de una gran cantidad de memoria gráfica.

4.1.2 Software

El software utilizado también tiene un gran peso en el desarrollo de este proyecto. En esta sección se explicará en qué consiste cada herramienta y tecnología utilizada en el desarrollo de este proyecto.

- **Lenguaje de programación:** Python (versión 3.10). [7]

Python es un lenguaje de programación de alto nivel, orientado a objetos, caracterizado por su flexibilidad de uso, debido a que puede utilizarse casi para cualquier actividad de programación. Su sintaxis es muy sencilla y cuenta con una gran cantidad de extensiones y librerías que hacen que Python funcione en cualquier campo de la programación. No hay que pasar por alto la importancia de las librerías, puesto que facilitan enormemente el trabajo del programador. [32]

En cuanto a inteligencia artificial se refiere, Python es el lenguaje de programación por excelencia. Por la flexibilidad que proporciona, la gran comunidad que tiene, la compatibilidad e incluso el uso de cuadernos Jupyter, que permiten desarrollar códigos por partes sin necesidad de ejecutar todo un algoritmo cada vez que se quiera modificar una parte del código.

En este proyecto, también se ha utilizado Python para el desarrollo de la aplicación web con la librería Flask. Se utilizó la versión 3.10 de Python debido a la compatibilidad entre versiones de las librerías.

En resumen, todo gira en torno al lenguaje Python en el backend de este proyecto.

- **Entorno de desarrollo:** Visual Studio Code [33]

Visual Studio Code es un popular editor de código fuente gratuito, multiplataforma, desarrollado por Microsoft. Se trata de una herramienta ligera que permite escribir, depurar, implementar y colaborar en proyectos de desarrollo de software. Integra control de versiones con Git de forma nativa y demuestra una gran compatibilidad con cualquier lenguaje de programación. Su extensa comunidad y las numerosas extensiones permiten adaptar VS Code a cualquier tarea, simplificando el desarrollo de código y concediendo al programador muchas otras herramientas, como la colaboración en tiempo real o el trabajo en remoto [34].

- **Entorno virtual:** Conda [35]

Anteriormente se ha mencionado que se trabajará con Python y sus librerías. Conda es la aplicación encargada de que todas funcionen correctamente sin molestarse unas a otras. Conda es un sistema de gestión de paquetes y entornos de código abierto que permite instalar múltiples versiones de paquetes de software y sus dependencias, alternando fácilmente entre ellas. De esta forma, se evitarán incompatibilidades y se asegurará que el proyecto pueda llevarse a cabo sin problemas de software.

- **Software de computación:** CUDA es una plataforma de computación paralela y un modelo de programación desarrollado por NVIDIA para realizar computación general en unidades de procesamiento gráfico (GPUs). En otras palabras, permite a los desarrolladores acelerar aplicaciones aprovechando la potencia de las GPU [36].
- **Frameworks y librerías:** A continuación, se explicarán los frameworks y librerías de Python utilizadas.

    - **Hugging Face Transformers:** Librería desarrollada por la plataforma Hugging Face que permite importar y exportar modelos de redes neuronales profundas de manera sencilla. Esta librería se caracteriza por permitir cargar los modelos ya entrenados, eliminando los requisitos computacionales que se necesitan para entrenar un modelo [8]. En este proyecto, se utilizará para cargar el modelo GPT-2 pre entrenado.

    - **Hugging Face Datasets:** Esta biblioteca facilita la importación y exportación de conjuntos de datos para cualquier tarea que requiera de manipulación de datos. Contiene unos potentes algoritmos para la carga de datos en una sola línea de código, procesando grandes conjuntos de datos sin restricciones de memoria para una velocidad y eficiencia óptimas. Además, tiene integración directa con la plataforma Hugging Face, permitiendo que se pueda trabajar con cualquier conjunto de datos disponible en la plataforma [37]. Esta librería se usará para descargar y manipular el dataset con las letras de canciones [28].

- **PyTorch:** [38] PyTorch es un framework open-source de aprendizaje profundo que se utiliza para crear modelos basados en redes neuronales. Su flexibilidad y facilidad de uso lo han convertido en el framework líder de construcción de modelos con IA. PyTorch admite una amplia variedad de arquitecturas de red neuronal, desde algoritmos de regresión lineal simples hasta redes neuronales convolucionales complejas y modelos de Transformers generativos utilizados para tareas como visión por ordenador o procesamiento de lenguaje natural (NLP) [39]. En este proyecto, PyTorch se utilizará para realizar el fine-tuning y entrenamiento del modelo que generará las letras.

- **AudioCraft:** [25] AudioCraft es un framework open-source de generación de música y audio compuesto por varios modelos de inteligencia artificial en conjunto: MusicGen para componer música, AudioGen para crear efectos de sonido y EnCodec para ayudar con la compresión de audio. Su objetivo es generar música basándose en unas instrucciones introducidas por el usuario [26]. En este proyecto, funcionará como modelo de generación de melodías. Para obtener más detalles, se recomienda revisar el marco teórico.

- **Flask:** [30] Un framework ligero de desarrollo de aplicaciones web en Python. Sus principales características son la simplicidad de desarrollo, la agilidad para realizar respuestas HTTP y su flexibilidad para todo tipo de aplicaciones [31]. En este proyecto, su función será mantener una aplicación web que funcione como soporte de los modelos. Para obtener más detalles, se recomienda revisar el marco teórico.

- **HTML, CSS:** HTML es el lenguaje de marcas con el que se desarrollan la inmensa mayoría de páginas web accesibles desde Internet. Diseña la estructura de una página web, es decir, títulos, párrafos, saltos de línea, imágenes… Por otro lado, CSS es el lenguaje de marcado que determina el aspecto con el que se visualiza el contenido de una página web. Utiliza elementos como el color, estilos de letra, estilos de imágenes… Su código se conjuga con el del HTML para crear páginas web completas [40]. Su función en este proyecto será el diseño y animación de la interfaz de la aplicación web que utilizará el usuario.

- **FFmpeg:** [41] FFmpeg es una completa solución multi-plataforma para grabar, convertir y hacer streaming de audio y vídeo a otras fuentes. Es utilizado en en proyectos enormes como Google Chrome, VLC, o MPlayer entre otros [42]. En este proyecto, se encargará de la conversión y manipulación del audio generado por el modelo de generación de melodías.

- **Jupyter Notebook:** [43] Jupyter Notebook es una aplicación web de código abierto que permite a los usuarios crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Esta herramienta es especialmente popular entre los científicos de datos, analistas y desarrolladores por su capacidad para facilitar el desarrollo, la documentación y la presentación de proyectos tanto de aprendizaje como profesionales. Soporta múltiples lenguajes de programación y encaja a la perfección con los entornos virtuales de Conda, debido a que permite cambiar muy sencillamente entre ellos [44]. En este caso, se instalará directamente desde el panel de extensiones de VS Code para trabajar con Notebooks en el entorno. Se utilizará para realizar pruebas y tests sobre las distintas partes del proyecto.

4.2 Arquitectura General del Sistema

El sistema se divide en dos grandes módulos principales:

1. **Módulo de generación de letras (NLP):**

   Utiliza un modelo GPT-2 entrenado con letras de canciones para generar nuevas composiciones líricas a partir de unas instrucciones dadas por el usuario, que contendrá el título de la canción, el artista en la que está basada y el inicio de la letra.

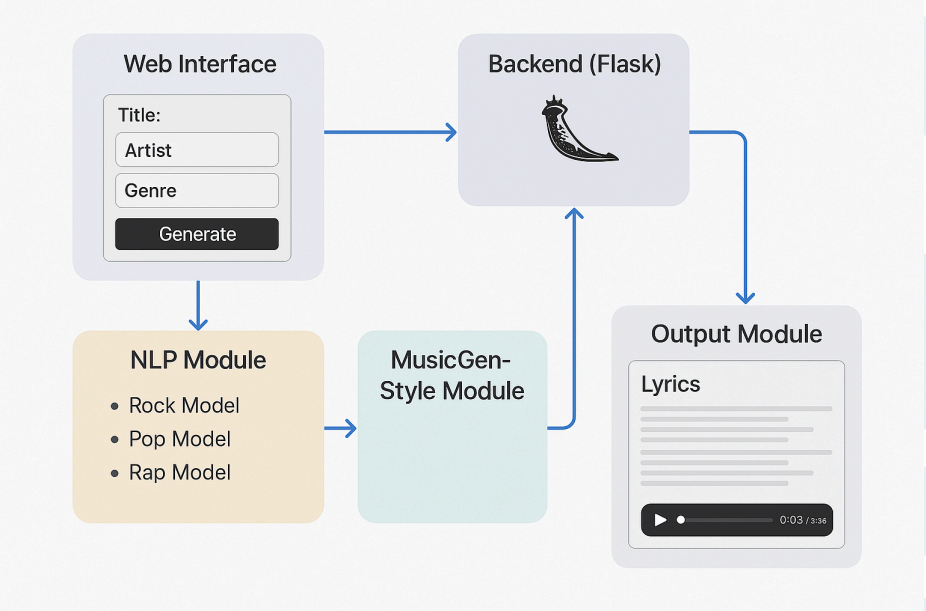
2. **Módulo de generación musical (Audio):**

   Utiliza AudioCraft para sintetizar una melodía basada en el texto generado previamente. Este módulo produce un archivo de audio en formato MP3.

Ambos módulos se integran en una interfaz web, dónde el usuario puede:

- Seleccionar un género para generar, tanto la letra como la música.
- Introducir un título, artista y una línea de inicio.
- Obtener una letra completa generada automáticamente.Tiene la opción de repetir el proceso hasta que la letra sea de su gusto.
- Escuchar una melodía generada a partir de dicha letra.
- Descargar el archivo resultante.

Visualizandolo en un esquema, el funcionamiento del proyecto sería tal que así.



4.3 Metodología del desarrollo

En esta sección, se explicarán paso a paso todas las técnicas realizadas y decisiones tomadas durante la realización del proyecto.

1. Preparación del entorno

Antes de comenzar a desarrollar el proyecto, siempre es necesario poner a punto el entorno en el que se trabajará para no encontrar problemas ajenos al proyecto.

El hardware estaba a disposición previamente, no se tuvo que realizar ninguna modificación. En la parte de software, se comenzó instalando Conda [35] para crear entornos virtuales en los que trabajar con las diferentes librerías y frameworks de Python, sin alterar nada del equipo principal. Una vez instalado, se instala Python con todas las librerías necesarias para trabajar. Conda se encargará de escoger e instalar las dependencias y versiones concretas de cada una de las librerías para no hallar problemas de compatibilidad.

Tras tener el entorno virtual instalado, se procedió a instalar el software CUDA [36], de NVIDIA, necesario para poder realizar el fine-tuning de los modelos NLP. Su instalación, de forma muy resumida, consistía en descargar una serie de archivos, instalarlos de forma sencilla y agregarlos a las variables de entorno del equipo. Se puede encontrar la guía de instalación oficial en la página web de NVIDIA [45].

Finalmente, se instaló el entorno de desarrollo VS Code. La instalación se realiza desde la página web oficial y es muy sencilla [46]. Se instalaron algunas extensiones para facilitar el desarrollo de código en Python y se configuró a gusto del programador para poder comenzar a trabajar.

Como conclusión, estos tres pasos seguidos pueden realizarse en cualquier orden puesto que son independientes entre ellos. Se ejecutó en este orden debido a que es más sencillo encontrar errores entre las librerías que en las instalaciones posteriores.

2. Desarrollo de un modelo NLP inicial

Tras la puesta a punto del entorno, el primer objetivo propuesto era entrenar un modelo inicial para evaluar su funcionamiento y a partir de ahí tomar decisiones sobre cómo mejorar sus prestaciones. Todas estas pruebas se llevaron a cabo en Notebooks de Jupyter, que facilitan mucho la ejecución de código por partes.

En todos los archivos Python, no importa que sean Notebooks u otros, se deben importar las librerías con las que se va a trabajar de forma previa. En este Notebook no comenzó de manera diferente. Se importó la librería “torch” para trabajar con redes neuronales, la librería “dataset” para cargar el dataset de letras y la librería transformers para importar el modelo GPT-2 con su tokenizador correspondiente. El tokenizador es una herramienta propia de cada modelo de NLP, que permite a los modelos de IA procesar grandes cantidades de datos de manera más eficiente.

Una vez importadas las librerías, comenzó el proceso de fine-tuning. El primer fine-tuning que se realizó no llevaría un preprocesamiento del dataset. En este primer modelo, se entrenaría con todas las canciones cuyo idioma principal fuese el inglés. El modelo GPT-2 es muy potente a la vez que ligero, pero se trata de un modelo que dispone sólo de vocabulario en inglés. Por tanto, el preprocesamiento de los datos consta de eliminar todos los registros que no sean en inglés. Se continuó dividiendo el dataset entre los datos que se utilizarán para el entrenamiento y los datos para el test.

Posteriormente, se cargó el tokenizador pre-entrenado y se le añadieron unos tokens especiales para que el modelo aprenda cuándo acaba una letra. Después de esto, se desarrollaron dos funciones para establecer un mismo formato a las letras y su posterior tokenización para que el modelo pueda leerlas de la manera más limpia posible.

Con todo esto, terminaría el preprocesamiento y comenzaría la configuración del entrenamiento. Se cargó el modelo GPT-2 y se definieron los parámetros del entrenamiento. Los parámetros se establecieron de forma que el entrenamiento fuese ligero para tener unos primeros resultados con los que tener una noción inicial. El parámetro principal era el número de épocas. Estas son un hiper parámetro que define el número de veces que el algoritmo trabajará a través del conjunto de datos, permitiendo que el modelo aprenda de cada ejemplo de los datos varias veces [47]. Se estableció un número de épocas inicial de 1. Finalmente, el entrenamiento comenzó a realizarse.

El modelo tardó aproximadamente 6 horas en realizarse. Una vez finalizado, se creó una función de inferencia que trasladase las instrucciones de un usuario al modelo entrenado. Las instrucciones se compondrían de el nombre de una canción, el artista en la que se basará y un inicio de la canción.

Como evaluación de este modelo, se introdujeron distintas cantidades de instrucciones que presentaron las siguientes conclusiones. El modelo funcionaba decentemente para ser un modelo inicial, pero se encontró que al escribir instrucciones sobre ciertos géneros musicales con menor número de canciones el modelo disminuía sus prestaciones considerablemente. Esto se debe a un problema de desbalance de clases del conjunto de datos. Existían un gran número de registros sobre géneros como rap, rock o pop, pero otros géneros como el country o el rythm & blues no tenían mucha diversidad.

La conclusión que se obtuvo de este modelo es que puede funcionar correctamente pero hay que arreglar este desbalance de datos para que los resultados mejoren.

3. Desarrollo de un modelo para cada género

En el primer modelo se encontró un desbalance de datos que hacía que el modelo no aprendiese bien de todos los géneros. Las técnicas más comunes para arreglar el desbalance de datos suelen pasar por eliminar registros de la clase más voluminosa hasta que esté al nivel de los demás, o la creación de registros sintéticos para aumentar los registros de la clase con menor observaciones. La primera técnica tiene una desventaja en la pérdida de información importante al eliminar registros, y la segunda suele obtener resultados sobreajustados, en otras palabras, más perfectos de lo que pueden ser, debido a que estos registros se han generado en base a parámetros como la media, la mediana o la moda. Además, puede funcionar de forma más óptima en conjuntos de datos numéricos, pero en conjuntos de datos que contienen muchas cadenas de caracteres no se pueden generar adecuadamente.

Por tanto, la decisión tomada fue eliminar estos géneros que no tuviesen un número considerable de observaciones. En el dataset existen 6 géneros: rap, pop, rock, misc, r&b y country. Se determinó que se trabajará con 3 géneros solamente; rap, pop y rock. De esa forma, el modelo podría centrarse en aprender patrones solamente de estos géneros devolviendo unos resultados mucho mejores.

Tras continuar reflexionando cómo se podría mejorar aún más el modelo NLP, surgió la hipótesis de independizar los géneros en cada modelo. Debido a la decisión de reducir la cantidad de géneros a 3, un número muy reducido, existe la posibilidad de entrenar un modelo para cada uno de estos géneros. De esta forma, el modelo aprendería patrones únicamente entre los de su género, sin tener que discernir entre varios géneros, lo que debería mejorar los resultados de manera considerable. Además, de esta forma se reduciría la importancia del desbalance de clases, debido a que cada modelo aprenderá de su género y no tendrá que aprender a distinguir entre ellos.

Así comenzó la segunda fase de desarrollo de modelos NLP. Comenzaría con el preprocesamiento, que sería muy similar al anterior. Partiendo del dataset original, se volverían a eliminar todas las canciones que no estuviesen en inglés. Después de este filtrado, se dividiría el dataset en 3: uno por cada género.Cada dataset tenía el siguiente número de canciones:

Rock: 633308 canciones,Pop: 1393559 canciones, Rap: 964605 canciones.

Cada género tiene un número suficientemente alto de canciones para poder proporcionar unos buenos resultados.

A partir de aquí, el preprocesamiento continuaría de la misma manera en los 3 datasets. Se prosiguió con la carga del tokenizador.Se añadieron los tokens especiales de final de letra y se aplicaron las mismas funciones de formateo de letra y tokenización de texto. Se prosiguió dividiendo los datasets entre entrenamiento y test con una distribución de 90%/10% respectivamente. Es decir, el 90% de las canciones se utilizarán para entrenar el modelo y el 10% restante para los tests. La distribución quedaría tal que así:

Rock - ***Train***: 569977, ***Test***: 63331

Pop - ***Train***: 1254203, ***Test***: 139356

Rap - ***Train***: 868144, ***Test***: 96461

Para finalizar los preparativos del fine-tuning, se cargó el modelo GPT-2 pre-entrenado y se modificaron los hiperparámetros de entrenamiento para que esta vez fuese de la mayor calidad posible. A continuación, se listarán los parámetros más importantes:

- **Número total de épocas:** Número de veces que el modelo trabajará a través del dataset en el entrenamiento, permitiendo que el modelo aprenda de cada ejemplo de los datos varias veces [47]. Se estableció un total de 2, debido a que se tratan de datasets con más de 600k registros.
- **Tamaño del batch de entrenamiento y de test:** El tamaño del batch,o tamaño del lote, es un número que determina cuántas muestras se procesan simultáneamente en cada iteración del algoritmo de optimización. Influye en la velocidad y la estabilidad del entrenamiento, así como en el rendimiento final del modelo [48]. Se estableció un número de 4 en ambos, para no saturar la tarjeta gráfica.

- **Tasa de aprendizaje:** Este parámetro rige cuánto ajusta sus parámetros un modelo en cada paso de su algoritmo de optimización. El algoritmo de optimización de una red neuronal tiene como objetivo minimizar la función de pérdidas que mide la diferencia entre las predicciones del modelo y los datos reales [49]. Por tanto, este parámetro es importante para que el modelo ofrezca un rendimiento óptimo. Se estableció un valor de 3e-5 debido a que los rangos de aprendizaje recomendados en modelos de NLP son 5e-5, 3e-5 y 2e-5 [50], por lo que la decisión se decantó por el valor intermedio.
- **Estrategia de evaluación:** Indica qué plan se utilizará para evaluar el modelo. Se estableció el parámetro “steps”, qué hace que el modelo realice una evaluación cada cierto número de pasos.
- **Número de pasos para evaluar:** Esto indica cada cuántos pasos se realizará una evaluación. Se estableció un valor de 500, debido a que posteriormente, se establecería el número de pasos máximos de 1000. De esta forma, habría 2 evaluaciones por entrenamiento.
- **Número de pasos máximos:** Como se ha mencionado anteriormente, establece el número de pasos máximos de entrenamiento. Si se alcanza este número, el entrenamiento se detiene independientemente de que no haya alcanzado el número de épocas máximo.
- **Pasos de logs:** Establece cada cuantos pasos se guarda información, cómo la pérdida o la métrica de evaluación, en los logs. Se estableció un número de 100 pasos, para obtener 5 logs de información antes de la primera evaluación.
- **Pasos de acumulación de gradiente:** El gradiente en IA se utiliza para encontrar el mínimo de una función de pérdida al ajustar los parámetros de un modelo [51]. Estos parámetros en este modelo son los pesos de una red neuronal. Los pesos son valores numéricos que determinan la fuerza de las conexiones entre los nodos de una red neuronal [52]. Es decir, este parámetro acumula gradientes durante un número de pasos determinados antes de hacer una actualización de los pesos. Se estableció un número de 4 pasos para que los pesos se actualicen de forma constante.

Una vez se definieron todos los hiperparámetros, comenzó el entrenamiento de los 3 modelos. El modelo de rock tardó en entrenarse 24 horas, obteniendo un valor de la función de pérdida final de entrenamiento de 1.409100 y de test 1.398691. El modelo de pop tardó en entrenarse 45 horas, obteniendo un valor de la función de pérdida de entrenamiento de 1.441900 y de test de 1.358317. El modelo de rap tardó en entrenarse 25 horas y obtuvo unos resultados de la función de pérdida de entrenamiento y test de 2.726600 y 3.171878, respectivamente. Este último modelo tuvo unos resultados un poco más bajos, pero realizaríamos la comprobación de forma empírica.

Se desarrolló una función de inferencia para introducir instrucciones a los modelos. Esta función es importante debido a que es la que establece cómo funcionará el modelo en base a unos parámetros. Los parámetros más importantes son:

- ➔ **Longitud máxima:** Establece la longitud máxima del texto generado. Inicialmente, se establecieron un total de 150 palabras.
- ➔ **Temperatura:** Controla la aleatoriedad en la selección de la siguiente palabra. Si los valores son más bajos (<1) hacen que el texto sea más determinista. Por el contrario, si son más altos (>1) hacen que el texto sea más creativo. Por defecto, se estableció en 1.
- ➔ **Top_K:** Top_K hace que de entre todas las palabras posibles, solamente se consideran un valor establecido de las más probables. En este caso, solo las 50 más probables, reduciendo opciones poco probables y haciendo el texto más coherente.
- ➔ **Top_P:** Selecciona palabras hasta acumular un porcentaje establecido de la probabilidad total. El valor introducido fue de un 95%, por tanto, se ignorarán las palabras de menor probabilidad fuera de ese 95%

A continuación, se mostrará una prueba del modelo de rock:

```
[Artista: Muse]

[Canción: Unintended]

You could be my unintended baby

I've learned to keep my distance

But I see you're still there


You'll be my unexpected baby

And you've done it before

We all know that it can't last forever


You'll be my unexpected baby

And you've done it before

We all know that it can't last forever
```

I can't say it

So I will just say it

I can't say it

So I will just say it

I can't say it

So I will just say it

I'm still going to hurt you

I am, I am

And I'll stay that way

But we'll get back to the truth

(And I'm still going to hurt you)

And you'll find a way to turn that around

So I will just

Los resultados son bastante buenos. Se puede ajustar un poco más la función de inferencia modificando algunos parámetros pero el desempeño del modelo es más que decente.

4. Test inicial del modelo de generación de melodías

El siguiente paso de la metodología del método pasa por configurar el modelo de melodías. Al igual que en el modelo de generación de letras, el modelo de melodías se testeará primero en un Jupyter Notebook para comprobar su funcionamiento. Este modelo no requiere de ninguna modificación ni entrenamiento, facilitando su uso.

El testeo inicial comienza cargando la librería Audiocraft, cargando el modelo MusicGen y la función “audio_write”. Una vez importadas las librerías, el primer paso será cargar el modelo pre-entrenado de MusicGen, que al igual que GPT-2, dispone de varias versiones para utilizar, de la cuál se utilizará la versión llamada “melody”, una versión estable al igual que potente para la generación de melodías.

Tras cargar el modelo, se introdujo una instrucción genérica y se estableció una duración de la melodía creada de 10 segundos, para comenzar con un valor bajo. Este proceso tardó aproximadamente 5 minutos.

Finalmente, se utilizó la función *audio_write* para guardar y reproducir el audio en un archivo .mp3 en una ruta concreta. Finalmente, reproducimos el archivo generado y se pudieron sacar algunas conclusiones. El archivo generado era breve pero mostraba unos sonidos interesantes aunque solamente se tratasen de 10 segundos. Sin embargo, si aumentamos la duración del sonido, el modelo generará una melodía en la que ya se puedan distinguir versos, puentes y estribillos. Por otro lado, estos procesos tardarían mucho en generarse, requiriendo hasta 1 hora para audios de 30 segundos. Estas duraciones se deben a los recursos disponibles. Una mejor tarjeta gráfica o un equipo virtual en la nube mejorarían las prestaciones considerablemente, pero se escapa de presupuesto actualmente.

Como conclusión, el modelo puede utilizarse correctamente pero los tiempos de respuesta serán lentos debido a los recursos computacionales.

5. Desarrollo de la aplicación web que integre ambos modelos y facilite la interacción del usuario

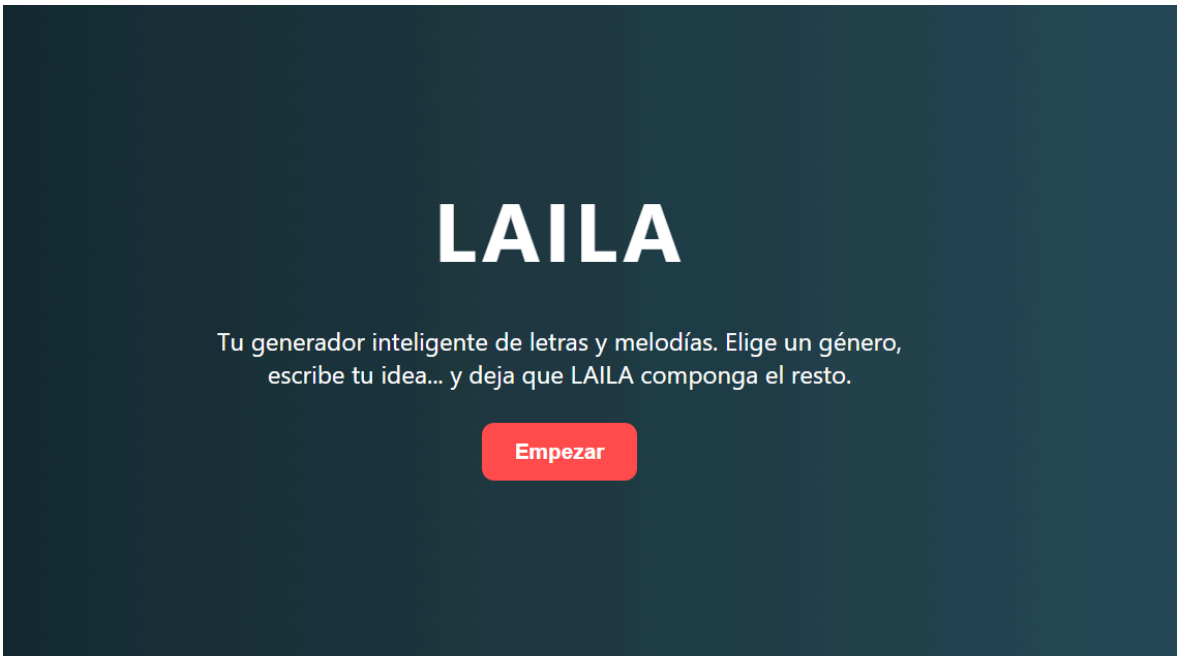
La siguiente parte se compondrá de la aplicación web que integrará ambos módulos. El objetivo principal de la aplicación es permitir al usuario poder utilizar el algoritmo de ambos modelos de la manera más sencilla posible. Para que esto sea posible, se desarrollará y se animará una interfaz con lenguaje HTML y CSS. En cambio, el funcionamiento y mantenimiento de la aplicación se realizará con Flask. Se disponían de conocimientos de trabajo con Flask, HTML y CSS previos al desarrollo del trabajo, aunque se ha necesitado revisar la documentación puntualmente [30].

El desarrollo comenzó por la interfaz, debido a que su codificación es la parte más sencilla de la aplicación y se tenía un concepto sobre cómo se podía realizar previamente. La interfaz se iba a componer de varias páginas distintas, concretamente de 6:

1. **Introducción:** Será la portada de la aplicación. Esta se compondrá del título del proyecto, una breve descripción sobre su funcionamiento y un botón de comenzar.
2. **Elección de modelo:** Tras pulsar el botón de comenzar, el usuario será dirigido a una pantalla en la que encontrará 3 botones con cada uno de los géneros musicales.
3. **Generador de letras:** Una vez elegido el género, el usuario se encontrará un formulario en el que tendrá que introducir el título de la canción que quiere crear, el artista en el que quiere que se base y el comienzo de la letra. Finalmente, podrá pulsar en generar para que comience el proceso de generación de letras. Tras unos segundos, se generará una letra y se expondrá en la pantalla. Junto con la letra aparecerán 3 botones: uno de repetir generación, otro de nueva generación y otro de continuar con la música. El primero repite la generación de la letra con las mismas instrucciones generadas. El segundo borra la letra y el formulario aparecerá en blanco para introducir nuevas instrucciones. El tercer y último botón pasará a la pantalla de generación de música. Esta página será exactamente igual para todos los géneros.
4. **Generación de música:** Esta página mostrará el audio generado en el centro de la pantalla, con la opción de reproducirlo, descargarlo o crearlo de nuevo. También habrá un botón de vuelta al inicio.

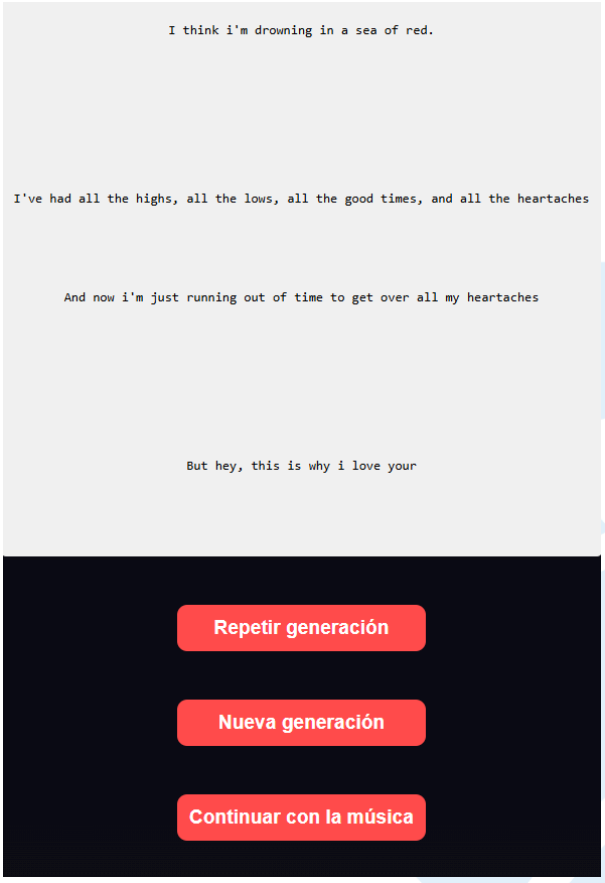
Con estas pautas, se comenzó a diseñar las interfaces. Se buscaron algunos diseños por internet para coger ideas, aunque el objetivo principal era un diseño minimalista. Los diseños quedaron tal que así:

1. **Introducción**



2. **Elección de modelo**

3. Generación de letras



Las imágenes de la pantalla de elección de género han sido recogidas de las siguientes páginas web con licencia gratuita [53] [54] [55].

Una vez finalizados todos los diseños, el siguiente paso es acoplar los algoritmos a los distintos botones para que funcionen correctamente. Esta tarea comenzó creando un archivo Python llamado "app.py". A partir de ahora, no se trabajará con Notebooks, si no con archivos Python. También se creará otro archivo Python llamado "inference.py". Este archivo contendrá dos funciones; una primera función de carga de modelos y tokenizadores para cargar cada uno de los modelos entrenados, y una segunda función cuyo trabajo será introducir los parámetros e instrucciones a los modelos y generar las letras. Estas funciones podrían introducirse en el módulo principal, pero de esta forma se mantiene un mejor orden para trabajar.

Al igual que en los Notebooks, se comienza importando todas las librerías y módulos con los que se va a trabajar. Concretamente, se importará la librería Flask y varios módulos de esta. Además, se importarán las funciones implementadas en el archivo "inference.py" para poder utilizarlas en este módulo. Antes de comenzar a programar con Flask, se deben cargar los 3 modelos para que estén disponibles en el preciso instante en el que se inicia la aplicación.

Flask trabaja con rutas, que contienen funciones concretas que pueden intercambiar instrucciones entre los distintos formularios y el módulo principal. Las primeras rutas, solamente funcionarán como redirección. Carga la interfaz y al hacer click en los botones se redirige a la siguiente pantalla.

Esto cambiará en las páginas de generación de letras. En estas, se comenzará a generar una función que obtenga el contenido del formulario, en otras palabras, el título, el nombre del artista y el inicio de la letra introducidos en la página web. Estos datos recogidos se introducen en la función de generación de letras, se activa y los resultados generados (la letra) se envían a la página web para mostrarlos por pantalla. Los botones de acción posteriores aparecerán tras obtener una letra generada. El botón de repetir generación simplemente repite el proceso con los datos que se obtuvieron previamente. El botón de nueva generación redirige hacia la página vacía del formulario. Y el botón de continuar con la música introduce la letra en el modelo de generación de música y redirige a la última página de música. Esta función completa se repetirá 3 veces, una vez para la ruta de cada modelo.

Finalmente, se codificó una ruta que obtuviese el audio generado por el modelo y lo expusiese en la página web. Aquí se encontrarán los botones de repetir generación, descargar el audio y volver al inicio.

De esta forma, concluye el desarrollo de la aplicación web y la finalización de la metodología. Cómo maqueta inicial, funciona muy correctamente, pero para explotarlo como servicio sería necesario algunas inversiones para aumentar el rendimiento.

5. Problemas encontrados

Durante el desarrollo del proyecto no se han encontrado muchos dilemas. Las investigaciones se han realizado de una forma adecuada, impulsando que las decisiones tomadas hayan sido, como mínimo,correctas. Puede que otros modelos sean mejores, pero estos tienen unos resultados muy decentes.

El mayor problema encontrado son los recursos computacionales disponibles. El mayor problema que tienen los modelos de redes neuronales es que para poder experimentar con ellos se necesitan unos recursos muy altos, y cuanto más ambicioso sean los proyectos y los usos, más altos son estos recursos. El equipo propio disponible era de altas capacidades, pero el proyecto necesitaba de unos recursos mejores. De esta forma, el entrenamiento de modelos de NLP serían de mejor calidad y se reduciría el tiempo de entrenamiento, al igual que el modelo de audio reduciría sus tiempos de generación.

Otro problema que se encontró fue con la instalación de librerías. Inicialmente, Conda mostraba un error de compatibilidad entre algunas librerías que se necesitaban en diferentes versiones en algunas librerías. Este error se solucionó tras actualizar Conda y algunas librerías que estaban obsoletas.

En resumen, no se ha encontrado un problema que impidiese el proyecto, más allá de realizar un proyecto demasiado ambicioso para unos recursos que estaban limitados.

6. Conclusión

Este proyecto ha demostrado la viabilidad de integrar técnicas avanzadas de inteligencia artificial para la generación automática de contenido musical, combinando modelos de lenguaje natural para la creación de letras con modelos generativos de audio para la composición de melodías. La implementación de modelos pre-entrenados, junto con estrategias de fine-tuning con datasets específicos por género, ha permitido obtener resultados coherentes.

Uno de los aportes principales ha sido el desarrollo de una arquitectura modular capaz de vincular entradas del usuario con modelos especializados, permitiendo generar composiciones personalizadas a partir de títulos o fragmentos de letras. Además, se evaluó la calidad del contenido generado de forma empírica, confirmando que el sistema puede producir resultados creativos y musicalmente aceptables a gusto del usuario.

A pesar de ciertos desafíos técnicos, se logró un entorno funcional que puede ser escalado o refinado en futuros desarrollos. Esto abre oportunidades para su aplicación en ámbitos como la asistencia a compositores, la generación de contenido para videojuegos o el desarrollo de herramientas creativas interactivas.

En resumen, este trabajo sienta una base sólida para continuar explorando la intersección entre inteligencia artificial y arte musical, promoviendo nuevas formas de colaboración hombre-máquina en el proceso creativo. Con este proyecto, no se pretende eliminar el carácter humano de la música, sino facilitar la expresión musical para aquellos que no disponen de instrumentos o conocimientos. Por ejemplo, un equipo pequeño que quiere animar un corto con música, personas que se dediquen a la animación de videos de forma personal o profesional, para la creación de anuncios y miles de posibilidades más.

7. Referencias

[1] A. Vaswani *et al.*, «Attention Is All You Need», 2017, *arXiv*. doi: 10.48550/ARXIV.1706.03762.

[2] «¿Qué es un modelo de IA? | IBM». Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/ai-model
[3] «¿Qué es un LLM? - Explicación de los modelos de lenguaje grandes - AWS», Amazon Web Services, Inc. Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/large-language-model/
[4] «Procesamiento del Lenguaje Natural (PLN): Qué es y por qué es importante». Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://www.sas.com/es_es/insights/analytics/what-is-natural-language-processing-nlp.html
[5] J. Copet *et al.*, «Simple and Controllable Music Generation», 2023, *arXiv*. doi: 10.48550/ARXIV.2306.05284.
[6] «Hugging Face – The AI community building the future.» Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/
[7] «Welcome to Python.org», Python.org. Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://www.python.org/
[8] «🤗 Transformers». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/docs/transformers/es/index
[9] «Llama», Llama. Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://www.llama.com/
[10] L. M. Ulizarna, «LLaMA-2: NLP open-source de la mano de Meta AI», Adictos al trabajo. Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://adictosaltrabajo.com/2023/08/08/llama-2-nlp-open-source-meta-ai/
[11] «Introducing Meta Llama 3: The most capable openly available LLM to date», Meta AI. Accedido: 9 de mayo de 2025. [En línea]. Disponible en: https://ai.meta.com/blog/meta-llama-3/
[12] «grantsl/LyricaLlama · Hugging Face». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/grantsl/LyricaLlama
[13] «xaviviro/Lorca-LLama3-8B-GGUF · Hugging Face». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/xaviviro/Lorca-LLama3-8B-GGUF
[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, y I. Sutskever, «Language Models are Unsupervised Multitask Learners».
[15] «Qué es GPT: definición», Salesforce. Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://www.salesforce.com/es/resources/definition/gpt/
[16] «Models - OpenAI API». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://platform.openai.com
[17] «GPT-2: 1.5B release». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://openai.com/index/gpt-2-1-5b-release/
[18] «openai-community/gpt2 · Hugging Face». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/openai-community/gpt2
[19] «openai-community/gpt2-medium · Hugging Face». Accedido: 12 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/openai-community/gpt2-medium
[20] «WaveNet», Google DeepMind. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://deepmind.google/research/breakthroughs/wavenet/
[21] A. van den Oord *et al.*, «WaveNet: A Generative Model for Raw Audio», 2016, *arXiv*. doi: 10.48550/ARXIV.1609.03499.
[22] «NSynth: Neural Audio Synthesis», Magenta. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://magenta.tensorflow.org/nsynth
[23] J. Engel *et al.*, «Neural Audio Synthesis of Musical Notes with WaveNet

Autoencoders», 2017, *arXiv*. doi: 10.48550/ARXIV.1704.01279.
[24] «MusicGen - Generación de música con IA avanzada». Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://musicgen.com/
[25] «AudioCraft». Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://audiocraft.metademolab.com/
[26] DimensionIA, «AudioCraft - Generación y Procesamiento de Audio con IA», DimensionIA. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://www.dimensionia.com/audiocraft-generacion-procesamiento-audio-con-ia/
[27] «¿Qué es el fine-tuning? | IBM». Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/fine-tuning
[28] «sebastiandizon/genius-song-lyrics · Datasets at Hugging Face». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/datasets/sebastiandizon/genius-song-lyrics
[29] «Genius | Song Lyrics & Knowledge», Genius. Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://genius.com/
[30] «Welcome to Flask — Flask Documentation (3.1.x)». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://flask.palletsprojects.com/en/stable/
[31] «¿Qué es Flask (Python) y cuáles son sus ventajas? | Blog de Arsys», Arsys. Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://www.arsys.es/blog/que-es-flask-python-y-cuales-son-sus-ventajas
[32] «Python: qué es, para qué sirve y cómo se programa | Informática Industrial». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://www.cursosaula21.com/que-es-python/
[33] «Documentation for Visual Studio Code». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://code.visualstudio.com/docs
[34] «¿Qué es Visual Studio Code y cuáles son sus ventajas? | Blog de Arsys», Arsys. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas
[35] «Conda | Anaconda.org». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://anaconda.org/anaconda/conda
[36] «CUDA Zone - Library of Resources», NVIDIA Developer. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://developer.nvidia.com/cuda-zone
[37] «Datasets». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://huggingface.co/docs/datasets/index
[38] «PyTorch», PyTorch. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://pytorch.org/
[39] «¿Qué es PyTorch? | IBM». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/pytorch
[40] Roberto, «Qué es HTML y CSS: los básicos del desarrollo web |». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://esdima.com/que-es-html-y-css/
[41] «Documentation». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://ffmpeg.org/documentation.html
[42] «FFMPEG: qué es y cómo instalar el codec multimedia en Windows», SoftZone. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://www.softzone.es/programas/video/ffmpeg/
[43] «Project Jupyter Documentation — Jupyter Documentation 4.1.1 alpha documentation». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://docs.jupyter.org/en/latest/
[44] «Jupyter Notebook: Herramienta esencial para científicos de datos |

OpenWebinars», OpenWebinars.net. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://openwebinars.net/blog/jupyter-notebook/
[45] «1. Introduction — Installation Guide Windows 12.9 documentation». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html
[46] «Download Visual Studio Code - Mac, Linux, Windows». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://code.visualstudio.com/Download
[47] «Época en el aprendizaje automático (ML) | Ultralytics». Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://www.ultralytics.com/es/glossary/epoch
[48] «Tamaño del lote en el aprendizaje profundo | Ultralytics». Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://www.ultralytics.com/es/glossary/batch-size
[49] «¿Qué es la tasa de aprendizaje en machine learning? | IBM». Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/learning-rate
[50] «Tutorial: Perfeccionamiento de BERT para el análisis de sentimientos - por Skim AI». Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://skimai.com/es/ajuste-de-bert-para-el-analisis-de-sentimientos/
[51] «Gradiente Descendente», FOQUM. Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://foqum.io/blog/termino/gradiente-descendente/
[52] «Peso», FOQUM. Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://foqum.io/blog/termino/peso/
[53] magazinecrew, «Historia de la música: el nacimiento del rap (I)», UNDERGROUNDLAB. Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://undergroundlab.es/2020/08/rap/
[54] «¿Qué es el rock?», We Are Rock. Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://wearerock.wordpress.com/2013/11/24/que-es-el-rock/
[55] «Views Of Michael Jackson Concert During by New York Daily News Archive - Online Store». Accedido: 16 de mayo de 2025. [En línea]. Disponible en: https://camoloveet.click/product_details/73571095.html