CIS2130CP06

Generación de música basada en un género musical

Diego Alejandro Gamboa Pachon
José Nicolas García Pineda
Oscar Giovanni Fonseca Neira

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2021

CIS2030CP06

Generación de música basada en un género

Autor(es):

Diego Alejandro Gamboa Pachon
José Nicolas García Pineda
Oscar Giovanni Fonseca Neira

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR CON UNO DE LOS REQUISITOS DEL GRADO EN INGENIERÍA DE SISTEMAS

Director

PhD. Jaime Andrés Pavlich Mariscal

Jurados del Proyecto Final de Grado

Andrea del Pilar Rueda Olarte

Andrés Darío Moreno Barbosa

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
Mayo, 2021

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENERÍA DE SISTEMAS

Rector de la Pontificia Universidad Javeriana

Jorge Humberto Peláez Piedrahita, S.J.

Decano de la Facultad de Ingeniería

Ing. Lope Hugo Barrero Solano , Sc.D.

Directora de la Carrera de Ingeniería de Sistemas

PhD. Alexandra Pomares Quimbaya

Director del departamento de Ingeniería de Sistemas

PhD. Efraín Ortiz Pabón

Artículo 23 de la Resolución No. 1 de junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Agradecemos a cada una de las personas que fueron participes en el desarrollo de este trabajo de grado, al PhD Jaime Andrés Pavlich que ejerció el rol de Director de Trabajo de grado y quien nos apoyó en la concepción y ejecución del proyecto de grado. A la Pontificia Universidad Javeriana que nos hizo parte de la familia Javeriana y nos permitió formarnos personal y profesionalmente, brindándonos los recursos y guía necesaria para esto.

A cada uno de los profesores que durante el proceso de formación nos nutrieron con el conocimiento que nos permitió llegar a este punto y que fueron ejemplo de esfuerzo y rectitud enseñándonos a conocer nuestras capacidades.

"*la creencia que se convierte en verdad para mi… es aquella que me permite hacer un mejor uso de mi fuerza, el mejor medio de poner en acción mis virtudes.*"

André Gide

Finalmente, agradecemos también a las personas que de forma indirecta fueron parte del proceso, tanto como amigos y familiares quienes nos apoyaron cuando fue necesario.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

ABSTRACT

Thanks to the advances that computing, and software engineering have had, it is possible to automate many tasks that previously could only be done by humans. One of these tasks is music composition. As a graduate project, we made a software system that is capable of composing and generating a complete song of a specific genre without any human intervention in the process. This system is capable of automatically creating the lyrics of the song, the music, melody, and the voice that sings. These elements are integrated into a complete reggaetón song. The song can be generated automatically and with input parameters.

I- INTRODUCCIÓN

En este documento se describe el proceso que se realizó para desarrollar el sistema de generación de música basado en un género. El sistema permite generar una canción de manera automática (sin ningún parámetro de entrada) y semiautomáticamente (con parámetros de entrada como la escala, bpm, estructura de la canción, entre otros), esto se logra desarrollando los distintos componentes que hacen parte de una canción los cuales son: la letra, la música (ritmo, melodía, armonía) de la canción y la voz que canta la letra y va en sincronía con la música. Para este trabajo elegimos el género musical reggaetón debido a que es un género con una complejidad musical y literaria relativamente sencilla en comparación a otros géneros musicales. El sistema es capaz de generar los componentes de una canción de manera independiente y también de integrarlos en una canción completa.

Para el desarrollo de este proyecto, primero se realizó el análisis de la oportunidad, en el cual se expone el contexto en el que tiene cabida este proyecto, se formulan los problemas específicos, se propone la solución y se justifica porque la solución propuesta es adecuada a la oportunidad formulada. Luego, se plantean los objetivos mediante el cual se definió el alcance del proyecto, una vez identificado el contexto del problema y expuesta la solución que se propone, se expone un marco teórico de información relevante al proyecto y un análisis del contexto donde se describen soluciones relacionadas, sus diferencias y similitudes con la solución propuesta. Teniendo claro el contexto del problema y la solución que se propone, se continúa con el resumen de la especificación de los requisitos, en donde se muestra cuáles son las principales funcionalidades y restricciones del sistema.

A partir de esto, se presenta el diseño del sistema en donde se muestran las principales herramientas utilizadas durante el proyecto, la arquitectura general del sistema y la explicación de su funcionamiento. A partir del diseño, se realizó la implementación y despliegue de la solución donde se especifica el proceso que se llevó a cabo para crear la solución. Posteriormente, se presentan los resultados obtenidos tanto de la aplicación de scrum, como del sistema de generación de música. Se continúa presentando los resultados de las pruebas que se realizaron a la herramienta para asegurar que se cumplieron los

requisitos y finalmente, se realiza un análisis de los resultados del proyecto para formular las conclusiones de este.

II- DESCRIPCIÓN GENERAL

En los últimos años ha tomado fuerza el uso de herramientas y tecnologías de automatización gracias al rápido desarrollo tecnológico que se ha estado presentado en esta área. Este desarrollo ha brindado la oportunidad de aplicar estos métodos en diferentes entornos con el fin de automatizar procesos, ya sea a nivel empresarial, investigativo o experimental [1]. En este mismo orden de ideas, la aplicación de estas herramientas y otras tecnologías puede denotarse en áreas como la informática, ciencia, ciberseguridad, educación, finanzas, salud, transporte y muchos campos más de las necesidades que presenta la humanidad [2].

1. Oportunidad, problema

La música es uno de los entornos en donde es muy común aplicar la automatización, la cual ha sido implementada en pequeña escala con técnicas y herramientas que facilitan la mejora de una canción en las áreas de composición, interpretación, teoría musical y procesamiento de sonido digital, en el cual se incluye la síntesis de instrumentos. Estas técnicas facilitan los procesos en la creación de música incluyendo la unificación de todos los aspectos que requiere una composición musical, lo cual disminuye la intervención del humano en estos procesos [3].

Una canción es una pieza musical que se compone de varios elementos que se podrían resumir en 3 conceptos macros: la parte musical, la letra y la voz y, dependiendo del género, la importancia de uno u otro es mayor. Estos tres componentes se han trabajado desde con técnicas que permiten la automatización de procesos: generación de música automática, generación de texto automático y la síntesis de voz [4].

A partir de la investigación de trabajos relacionados, se logró concluir que las soluciones orientadas a la composición específica de canciones de reggaetón en español son limitadas, lo cual abre la oportunidad de crear una herramienta para este nicho.

1.1. Contexto del problema

En el sector musical, el uso de herramientas que facilitan la creación de música es cada vez más frecuente, pero su visión va dirigida a producción de alta calidad y a un público especializado [3]. Para usuarios aficionados, principiantes o que se dediquen ocasionalmente a usar este tipo herramientas no es factible hacer uso de ellas por su complejidad y costo, por lo cual se requiere una herramienta de fácil acceso, sencilla y práctica que apoye sus procesos musicales en el entorno del género del reggaetón.

1.2. Formulación del problema y oportunidad

se desarrolla una herramienta con la integración de los 3 conceptos anteriormente mencionados, con el fin de componer una canción completa, como se describe a continuación:

- **Generación de música semiautomática:** Con el uso de algoritmos de composición musical a partir de cierta información de entrada, es posible generar música que imita estilos existentes de un género en específico. El objetivo es utilizar una serie de algoritmos que, mediante el uso de herramientas y librerías orientadas de síntesis de sonido, sea capaz de generar la composición musical de una canción [5].
- **Generación de texto automático:** Se basa en la obtención de textos cuyo significado se ajuste en gran medida a uno o más datos de entrada que permiten generar estructuras lingüísticas que representan el mensaje a transmitir [6], siendo en este caso un listado de letras de canciones que permiten el patrón de composición literaria.
- **Síntesis de Voz:** La síntesis de voz es el proceso mediante el cual se convierte un lenguaje de texto en voz cantada, La calidad de un sintetizador de voz se juzga por la similitud que tenga con la voz humana y su habilidad para ser entendido con claridad. en el contexto musical la voz debe ser sintetizada en base a una melodía para obtener el efecto del canto [7].

1.3. Propuesta de solución

La solución propuesta consistió en el desarrollo de un software que genera de manera semiautomática una canción mediante la integración de los 3 conceptos anteriormente mencionados, usando diferentes métodos de automatización para cada uno de ellos: música, texto y voz. Para la solución propuesta perteneciente a un trabajo de ingeniería de software, se considera al género reggaetón como caso de estudio.

Por medio de software que permiten la creación y reproducción de audio en tiempo real [8], apoyándose en ritmos euclidianos y la teoría musical propia del género reggaetón se genera una base rítmica y una melódica. La generación de texto se basa en el uso de machine learning, implementando técnicas de generación automática de texto, mediante el análisis de varias letras musicales con el fin de generar una nueva letra de reggaetón.

Por último, para la generación de voz por medio de una melodía previamente creada y un software existente actualmente en la industria que permita la síntesis de esta y adaptarlo para enunciar textos en español que se asemejan al canto.

Figura 1 Diseño herramienta generación de canciones

Por lo tanto, como conclusión, el componente de generación de música es semiautomático ya que se usan patrones predefinidos para la base musical, el componente de generación de letras es totalmente automático y el de generación de voz corresponde a la integración con un sistema de síntesis de voz, los cuales al ser integrados juntos tienen la capacidad de generar una canción completa.

1.4. Justificación de la solución

Actualmente, existen diferentes soluciones que permiten la generación de música, texto (letra) y voz, ya sea por separado o unificadamente, tal como es el caso del proyecto Flow Machines desarrollado por SONY CSL, siendo una de sus primeras creaciones Daddy's Car, una canción basada en los Beatles [9].

El proyecto Flow Machines es una solución altamente robusta y efectiva que no se encuentra disponible para el público, situación que también ocurre con otros proyectos que existen en la actualidad. Es por esta razón que el enfoque de esta tesis es generar el prototipo de una herramienta flexible y disponible para músicos y aficionados, ejerciendo su papel como una herramienta de apoyo a la creación musical, ya sea usando alguno de sus componentes por separado o la solución completa sin tener inconvenientes de derechos de autor.

Para esta tesis se tomó como caso de estudio al género reggaetón, ya que sus canciones son sencillas por su estructura tanto musical como lírica [10]. Adicionalmente, el reggaetón es uno de los géneros musicales más escuchados en la actualidad, con una alta influencia en la sociedad, principalmente en los jóvenes, además es un género es español, lo cual aporta valor a la propuesta ya que otras herramientas tienen una orientación a generar música únicamente de habla inglesa [11].

2. Descripción del proyecto

2.1. Objetivo general

Desarrollar un software que genere una canción de reggaetón semiautomáticamente.

2.2. Objetivos específicos

- Desarrollar un módulo de software para crear la parte instrumental y melódica de una canción de reggaetón.
- Desarrollar un módulo de software para crear la letra de una canción de reggaetón.
- Desarrollar un módulo de software para la síntesis de voz para una canción de reggaetón.

- Implementar un plan de validación para la solución integrada de los componentes.

2.3. Entregables, estándares y justificación

A continuación, en la tabla 1 se describen los entregables que se realizarán durante el desarrollo de la propuesta.

Entregable	Estándares asociados	Justificación
Software Project Management Planning (SPMP)	ISO/IEC/IEEE 16326:2019 Systems and software engineering - Life cycle processes - Project management	Se eligió este estándar en su última versión ya que proporciona el contenido requerido que se establece en un plan de gestión de un proyecto de software.
Software Requirements Specifications (SRS)	ISO/IEC/IEEE 29148:2018 Systems and software engineering - Life cycle processes - Requirement's engineering	Se eligió este estándar debido a que especifica los procesos requeridos que se utilizan en las actividades de ingeniería que resultan en los requerimientos para un producto de ingeniería software.
Backlog	SCRUM	Listado de historias de usuario se definirán junto con el director del trabajo de grado, quien será el product owner. El backlog está ligado a los requerimientos definidos en el SRS [12].
Software Design Description (SDD)	IEEE 1016:2009 IEEE Standard for Information Technology - Systems Design - Software Design Descriptions	Este estándar se escogió ya que contiene la información requerida que debe tener un SDD.
Release	SCRUM	Es la entrega funcional de una versión parcial del software, con el fin de hacer pruebas de uso y si es necesario, realizar ajustes en el desarrollo y definición [12].
	Agile Unified Process (AUP)	Es la solución final de la herramienta, esta se entrega una vez se hayan realizado las pruebas

		de regresión del software desarrollado [13].
Manual de Usuario	ISO/IEC/IEEE 26512:2018 Systems and software engineering - Requirements for acquirers and suppliers of information for users	Se escoge esta norma porque especifica la información coherente, completa, precisa y utilizable que se le proporciona al usuario.
Documento de certificación de calidad	ISO/IEC/IEEE 29119- 3:2013 Software and systems engineering - Software testing - Part3: Test documentation	Se selecciona este estándar ya que especifica las plantillas que pueden ser utilizadas para la documentación de prueba de software.

Tabla 1 Entregables y estándares

III- CONTEXTO DEL PROYECTO

1. Marco teórico

El objetivo principal de la propuesta es la generación de una canción que, como se mencionó en secciones anteriores, consiste en una pieza musical que consta de tres componentes macros: la parte musical, la voz y la letra [4]. Es a partir de estos componentes que surgen los siguientes conceptos fundamentales.

Inicialmente, se encuentra la generación de la música mediante el uso de un formato llamado MIDI con extensión mid, dicho formato es muy popular y utilizado en programas de edición de audio con fines de composición musical [14]. El formato de archivo MIDI fue desarrollado para la transmisión de datos MIDI de una aplicación a otra por músicos y desarrolladores de audio. Los archivos MIDI son archivo que almacenan la notación, el tono, la velocidad y las señales de control necesarias para varios parámetros, incluidos el volumen, las señales de reloj, entre otros, para sincronizar el tempo entre diferentes dispositivos. El formato MIDI estándar contiene dos variantes: Tipo 0 y Tipo 1. La primera es una pista, mientras que la segunda contiene datos multipista. Asimismo, la creación musical se basa en la teoría musical, que varía según el género y se divide en ritmo, melodía y armonía, tres conceptos básicos de la composición musical. [15].

En primer lugar, el ritmo hace referencia a la combinación de figuras y silencios dentro una canción. Con respecto al proyecto, se utilizó el ritmo conocido como dembow. El dembow es un ritmo musical que tiene su origen de Jamaica y en Puerto Rico [16].

El principal elemento del dembow es que sirve de base para géneros musicales tales como el reggaetón y el dancehall, generalmente estos se diferencian entre sí por sus elementos melódicos particulares y tempos, por ejemplo, el ritmo dembow es más rápido en el reggaetón y es más lento en el dancehall. [16].

En segundo lugar, la melodía es la parte más visible de una canción, la cual denota el factor diferenciador de esta con otra. En palabras simples consiste en la combinación de las notas

musicales de una escala. Por último, se encuentra la armonía que es el acompañamiento de los acordes y se complementa junto a la melodía [15].

Para la construcción de la música se usa FoxDot la cual es una librería que se creada el año 2015 con el propósito de introducir al mundo del live coding (codificación en vivo) para los usuarios nuevos en la programación y que desean crear música de manera rápida y fácil. Esta es una librería para Python que se comunica con el motor de síntesis de sonido SuperCollider para hacer música y crea un entorno de programación interactivo [17].

Como se menciona antes, SuperCollider es una plataforma de composición algorítmica y síntesis de audio que es utilizada por músicos, artistas e investigadores que trabajan con sonido. Es un software de código libre disponible para Windows, macOS y Linux. Uno de los principales componentes de SuperCollider es scsynth, un servidor de audio en tiempo real que forma el núcleo de la plataforma. SuperCollider puede usarse para composición algorítmica y secuenciación y para conectarse a hardware externo, como controladores MIDI o para sus experimentos de programación diarios [8].
Los archivos de música y voz usan el formato WAV, este es un formato de audio que contiene datos de audio digital que se puede reproducir en varios reproductores multimedia y aplicaciones de desarrollo de audio digital. En comparación con el formato MP3 los archivos WAV son más grandes debido a que el contenido de audio digital está sin comprimir [18].

Continuando con los componentes de una canción, también se encuentra la generación de texto, que, en otras palabras, hace referencia a la letra de una canción. Para el proyecto se decidió hacer uso de Machine Learning siendo una técnica de aprendizaje automático que mejora la experiencia, es decir con el entrenamiento. Más explícitamente de un algoritmo llamado LSTM (Long short-term memory), el cual está basado en redes neurales recurrentes (RNN). Este algoritmo se diferencia de las redes neuronales tradicionales, ya que logra predecir y mantener secuencias y patrones en memoria por un mayor tiempo, dando la ventaja de obtener una predicción más exacta al resultado esperado. A grandes rangos el algoritmo LSTM puede hacer un seguimiento de patrones a largo plazo, de esta forma los entrenamientos realizados con una gran cantidad de iteraciones logran mantener un alto

grado de conocimiento sin perdida de patrones en este caso de texto [19]. Para el caso del proyecto se contempla utilizar este algoritmo para la predicción y generación de una letra a partir de un repositorio de letras de canciones existentes del género.

Finalmente, se encuentran los conceptos relacionados con la generación de la voz sintetizada, que hace referencia más exactamente a una tecnología llamada TTS (text to speech), que traducido al español significa texto a voz. En palabras sencillas, TTS es una tecnología que permite la conversión de un texto a una voz artificial o sintética [20]. Para el proyecto se usan librerías que proveen funciones que realizan la generación de la voz de manera sencilla. Uno de estos es espeak el cual es un sintetizador de voz de código abierto que acepta inglés, español y otros idiomas, está disponible para Linux y Windows. eSpeak permite que se proporcionen muchos idiomas en un tamaño pequeño, el habla es clara y se puede utilizar a altas velocidades, pero no es tan natural o suave como otros sintetizadores más robustos que se basan en grabaciones de habla humana existente [21].

El componente de generación de voz hace uso de MuseScore que es un programa de notación musical para Windows, Mac OS y Linux, este programa soporta una amplia variedad de formatos de archivos . Es software libre y de código abierto bajo GNU General Public License. Este programa es complementado por un visor y una aplicación de reproducción de partituras [22]. Este programa es capaz de generar y leer archivos con formato musicxml.

Los archivos MusicXML se han convertido en el estándar para compartir partituras interactivas. Hoy en día, más de 250 aplicaciones incluyen compatibilidad con MusicXML [23].

Adicionalmente el componente de voz usa un sistema de síntesis de canto llamado Sinsy_NG que está basado en un modelo oculto de markov HMM, Los modelos ocultos de Márkov son un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Márkov de parámetros desconocidos , estos modelos son especialmente aplicados a reconocimiento de formas temporales, como reconocimiento del habla, de escritura manual, de gestos, etiquetado gramatical o en bioinformática. En el reconocimiento de voz se emplea para modelar una frase completa, una palabra, un fonema o trifonema en el modelo acústico. Por

ejemplo, la palabra "gato" puede estar formada por dos HMM para los dos trifonemas que la componen /gat/ y /ato/ [24].

2. Análisis de contexto

Se hizo una búsqueda de soluciones relacionadas de las cuales fue posible obtener ideas y encontrar los factores que distinguen la solución propuesta de otras ya existentes. A continuación, se describe 4 de ellas:

- **A New Composition Algorithm for Automatic Generation of Thematic Music from the Existing Music Pieces**

El objetivo de esta investigación es producir música nueva, nunca escuchada antes, el algoritmo desarrollado utiliza técnicas de aprendizaje y probabilidad y análisis estadístico. El algoritmo usa secuencias de notas y otros parámetros musicales como la duración de la nota, tono, alteraciones, modificaciones (intensidad, velocidad) y densidad de repetición de secuencia de notas para la preparación de una tabla de probabilidad que generará nueva música. Usan música temática con piezas (del mismo tema) como música de entrada para el análisis, utilizando aprendizaje seguido de análisis estadístico. Se usa MATLAB para análisis y editor de música MC para su visualización. Este estudio de investigación es el primero de su tipo en crear piezas musicales temáticas de manera efectiva en un entorno basado en computadora. El resultado de esta investigación tiene una amplia gama de usos: música de espera durante llamadas telefónicas automatizadas, música de fondo en aeropuertos, aviones y restaurantes, entre otros [5].

Esta investigación solo se centra en desarrollar piezas musicales sin contemplar las letras y la voz que hacen parte de una canción más completa, por lo que resuelve un problema similar pero el alcance de la solución no alcanza a abarcar todos los elementos del trabajo de grado.

- **Generating lyrics with variational autoencoder and multi-modal artist embeddings**

Este es un sistema de generación de líneas de letras de canciones condicionado al estilo de un artista especificado. El sistema utiliza un codificador automático variacional con

incrustaciones de artistas. Se propone la formación previa de embeddings de artistas con las representaciones aprendidas por un clasificador de CNN, que está capacitado para predecir artistas basándose en espectrogramas MEL de sus clips de canciones. Este trabajo es el primer paso hacia la combinación de audio y texto. modalidades de canciones para generar letras condicionadas al estilo del artista. Los resultados preliminares sugieren que hay un beneficio en inicializar las incrustaciones de los artistas. con las representaciones aprendidas por un clasificador de espectrograma [25]. Este proyecto se centra en la generación de letras en base a las letras existentes de un artista en particular, en ese sentido ofrece una solución similar a nuestro trabajo de grado para la generación de letras, sin embargo, este proyecto no profundiza en la combinación de audio y texto (música y letra).

- **Towards an Automated Composer of Popular Spanish Songs: Integrating a Music Generator and a Song Lyrics Generator**

En este trabajo se presenta la integración de dos sistemas creativos en lo que puede verse como un compositor automatizado para la música popular española. Primero se presenta un sistema que genera melodías con un modelo de Markov aprendido de un corpus de música popular española. Entonces, dada la importancia de la letra en este contexto, la última se integró con un sistema de generación de letras existente, adaptado a este propósito. Varios experimentos fueron llevados a cabo para evaluar la calidad de los resultados. En general las melodías transmiten un sentimiento de música popular española, mientras que el texto de la letra es aceptable para un primer acercamiento [26]. Este trabajo trabaja en 2 de los 3 elementos que desarrollamos en el trabajo de grado, que son la generación automática de música y la generación automática de letras de canciones, sin embargo, no llegan a integrar el componente de canto, y las letras son genéricas y no están hechas con base a un género en específico.

- **Personalized Singing Voice Generation Using WaveRNN**

En este proyecto, se formula un marco de generación de voz de canto personalizado (SVG) utilizando WaveRNN con datos de entrenamiento no paralelos. esta solución desarrolla un

modelo de generación de voces de canto promedio utilizando WaveRNN con las voces de varios cantantes. Para mapear el canto Fonético y características prosódicas del canto en una plantilla a muestras de canto en el dominio del tiempo, se utiliza un i-vector del hablante extraído del habla de destino para controlar la identidad del hablante del canto generado. En tiempo de ejecución, una plantilla de canto y las muestras de voz de destino se utilizan para la generación de voces de canto de destino. Específicamente, el contenido y la identidad del hablante del discurso objetivo no es necesariamente el mismo que el de la plantilla de canto. Los resultados experimentales sugieren que el marco SVG personalizado supera a la tubería tradicional de conversión-codificador de voz en las evaluaciones subjetivas y objetivas [27].

Con respeto a esta solución nuestro trabajo de grado permite hacer síntesis del canto tomando como entradas la letra de la canción y una melodía base por lo que integra los componentes de música y letra para lograr una voz cantada, esta solución solo se centra en generar voces con base datos de entrenamiento por lo que no es fuente para componer una canción completa.

- **Comparación de alternativas**

A partir de lo explicado en las alternativas encontradas y dando uso de las conclusiones dadas a cada una de ellas, se realiza una tabla comparativa denotando los puntos principales a trabajar y de lo cual nos permite distinguir la diferenciación frente a las propuestas existentes respecto a la solución presentada en el trabajo de grado.

	[5]	[25]	[26]	[27]	Nuestra Solución
Genera música automática	+	-	+	-	+
Genera letras de canciones autónomamente	-	+	+	?	+
Sintetiza voz melódicamente	-	-	-	+	+/-
Integra música y letra	-	-	+	-	+
Integra música y voz	-	-	-	-	+
Integra letra y voz	-	-	-	-	+

Tabla 2. Comparación de alternativas

IV- ANÁLISIS DEL PROBLEMA

En esta sección se detallan los requisitos principales, junto a las limitaciones, restricciones y la especificación funcional del problema.

1. Requisitos

En la Tabla 3 se muestran los requisitos funcionales definidos para el problema.

ID	Requisito
R1F01	El sistema debe contar con un repositorio de letras de canciones de reggaetón
R1F02	El sistema debe procesar las letras de las canciones para ser entrenado con el algoritmo LSTM
R1F03	El sistema debe entrenar un algoritmo que genere una letra nueva a partir de las letras de canciones procesadas
R1F04	El sistema debe entregar al usuario la letra de la canción generada en un archivo de formato .txt
R1F05	El sistema debe hacer uso de una herramienta de instrumentos y sonidos para el componente de generación musical
R1F06	El sistema debe generar automáticamente la parte musical de una canción de reggaetón sin la necesidad de recibir parámetros de entrada
R1F07	El sistema debe generar semiautomáticamente la parte musical de una canción de reggaetón mediante el uso de parámetros de entrada definidos por el usuario
R1F08	El sistema debe generar un archivo en formato .wav que contiene la pista de la música
R1F09	El sistema debe hacer uso de un software libre para la generación de voz sintetizada
R1F10	El sistema debe permitir al usuario integrar la generación de música y voz a partir de una letra ingresada en formato txt y una melodía en formato midi
R1F11	El sistema debe permitir al usuario usar los tres componentes por separado o integrados entre ellos para generar música, letra o voz
R1F12	El sistema debe contar con una interfaz web para hacer uso de los componentes de generación de música, letra y/o voz

Tabla 3 Lista de requerimientos

A continuación, se muestra la especificación de los tres requisitos más importantes definidos mediante el uso de una prioridad calculada entre los integrantes del equipo.

# Requisito	R1F11	***Tipo de requisito***	Funcional	***Casos de uso asociados***	
Descripción	El sistema debe permitir al usuario usar los tres componentes por separado o integrados entre ellos para generar música, letra o voz				
Razón	El usuario tiene la libertad de escoger los componentes a utilizar				
Autor	Scrum Team				
Criterio de medición	Los componentes especificados por el usuario se generaron correctamente				
Prioridad	4.9				
Versión	1	***Fecha***		22/11/2020	

Tabla 4 Requerimiento principal 1

# Requisito	R1F06	***Tipo de requisito***	Funcional	***Casos de uso asociados***	CU-01
Descripción	El sistema debe generar automáticamente la parte musical de una canción de reggaetón sin la necesidad de recibir parámetros de entrada				
Razón	Se debe generar una melodía de forma automática para que el usuario pueda escucharla y usarla				
Autor	Scrum Team				
Criterio de medición	Archivo en formato WAV con el contenido de la melodía				
Prioridad	4.8				
Versión	1	***Fecha***		22/11/2020	

Tabla 5 Requerimiento principal 2

# Requisito	R1F10	***Tipo de requisito***	Funcional	***Casos de uso asociados***	
Descripción	El sistema debe permitir al usuario integrar la generación de música y voz a partir de una letra ingresada en formato txt y una melodía en formato midi				
Razón	En el caso de que el usuario no requiera usar el componente de generación de letra				
Autor	Scrum Team				
Criterio de medición	La música y la síntesis de voz se generaron correctamente teniendo concordancia con la letra ingresada				

Prioridad	4.7		
Versión	1	***Fecha***	22/11/2020

Tabla 6 Requerimiento principal 3

2. Restricciones

A continuación, se especifican las características o restricciones del sistema.

Restricciones Generales

- El sistema estará en idioma español
- Se podrá acceder al sistema mediante una página web o usando solicitudes HTTP RESTFUL
- La herramienta sólo será capaz de producir música reggaetón

Restricciones de Software

- El sistema debe utilizar el protocolo de red TCP/IP
- El sistema debe utilizar el protocolo HTTP
- Se utilizará un software libre para la síntesis de voz
- Se usarán librerías para los algoritmos de generación de texto
- El componente de generación de voz debe estar en sistema operativo Linux
- Se requiere que el usuario utilice un navegador web preferiblemente Google Chrome

Restricciones de Hardware

- La interfaz gráfica del sistema debe poder ser visualizada en computadores con pantallas de resolución mínima 1024X768 mp
- Se requiere un servidor web (proporcionado por la Universidad Javeriana) y un servidor de respaldo, teniendo en cuenta que las máquinas personales no cuentan con la capacidad para hacer un entrenamiento del algoritmo LSTM con un repositorio de letras extenso.
- Computador:
    - Procesador: 1.0 GHz
    - Memoria: 256 MB de RAM

- Disco Duro: 600 MB de espacio libre
- Se necesita conexión a internet de mínimo 5 mbps

3. Especificación funcional

La figura 2 muestra el diagrama de casos de uso que representa a los requisitos funcionales, dentro del diagrama se cuenta con un actor Músico que interactúa con el sistema de generación de música, el cual se encarga de ejecutar funciones automáticamente. Adicionalmente, el sistema hace uso de un sistema externo para llevar cabo la síntesis de voz.

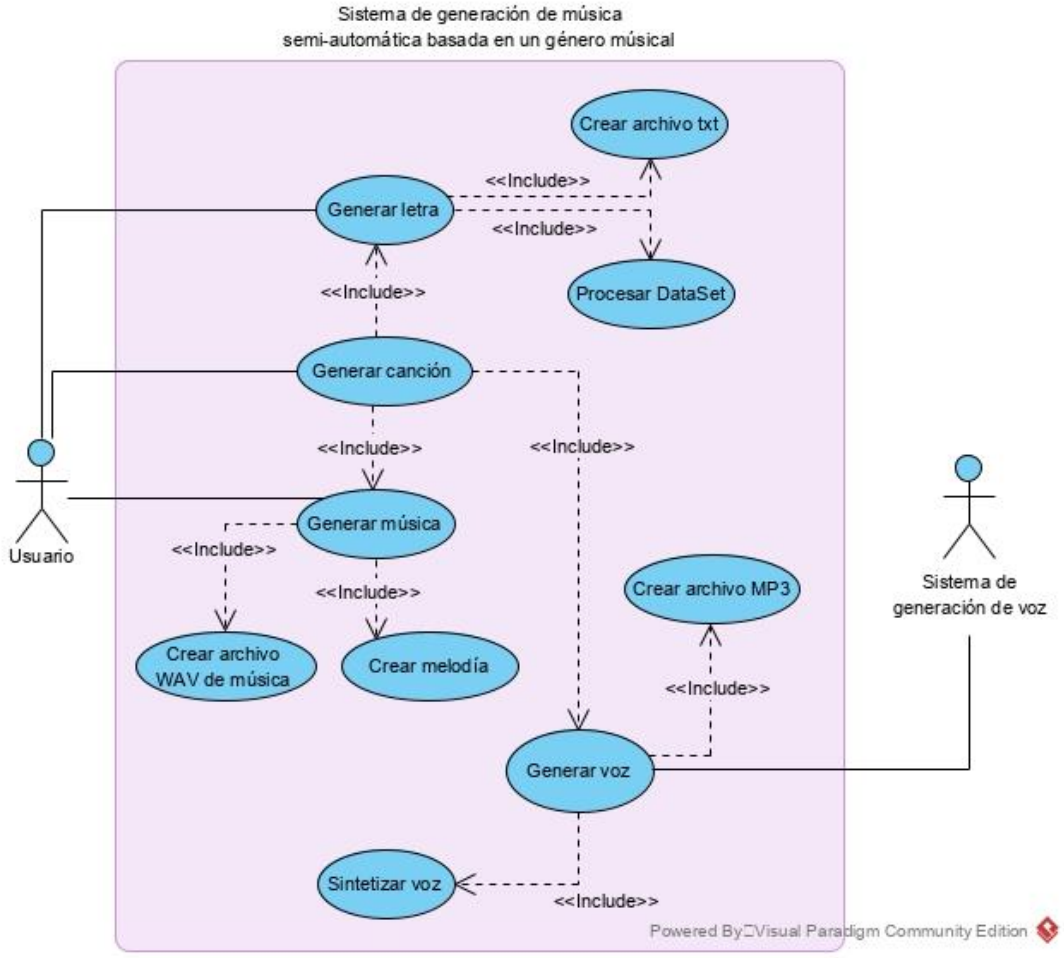

Figura 2 Diagrama casos de uso

V- DISEÑO DE LA SOLUCIÓN

En esta sección se describen los principales artefactos de diseño planteados para el desarrollo del trabajo de grado, haciendo uso del diagrama de componentes y el diagrama de despliegue, además se ofrece una explicación de las herramientas y tecnologías seleccionadas.

1. Funcionamiento del sistema

En esta sección se describe el funcionamiento general del sistema, la figura 3 es una representación del flujo que se ejecuta en el sistema en el momento que se crea una solicitud de canción.

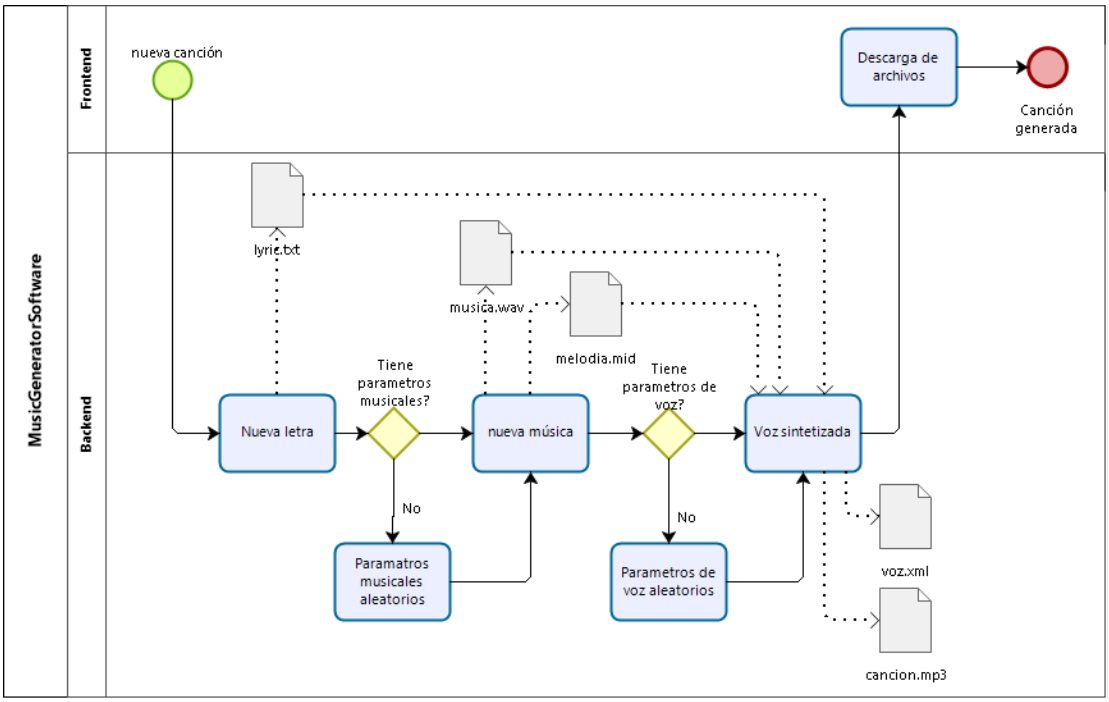

Figura 3 BPMN funcionamiento del sistema

Por medio de la página web, se envía una solicitud de nueva canción. Primero el sistema genera un archivo con la letra de una canción de reggaetón posteriormente el sistema valida si el usuario ingresó parámetros para la generación de la música y en el caso negativo el sistema es quien genera los parámetros de forma aleatoria, posteriormente se generan

archivos de melodía y música, con estos archivos y el anterior archivo de letra se realiza la síntesis de voz y la generación de la canción, esto se puede evidenciar en los archivos entregados de voz y canción. El usuario final podrá visualizar todos los archivos antes mencionados cuando termine la ejecución del sistema.

2. Herramientas y tecnologías

La tabla 7 muestra las principales herramientas utilizadas, cada una con su respectiva descripción, versión utilizada y justificación.

Nombre	Descripción	Versión	Justificación
Python	Es un lenguaje de programación interpretado, de alto nivel y de propósito general con un enfoque orientado a objetos que ayuda a los programadores a escribir código claro y lógico. [28]	3.6.9	Python es uno de los lenguajes de programación más populares actualmente, esto hace que su soporte sea muy bueno, adicionalmente, es de los más usados para temas de inteligencia artificial. Es por esto por lo que se escogió ya que cuenta con una extensa cantidad de frameworks y librerías que facilitaron el desarrollo del proyecto. Entre las librerías que se utilizaron están TensorFlow, Flask y FoxDot, siendo esta última primordial para utilizar SuperCollider como sintetizador de audio.
Docker Engine	Es un software de TI que permite el uso de contenedores como máquinas virtuales livianas y flexibles, permitiendo crearlos, implementarlos, copiarlos y moverlos de un entorno a otro. [29]	20.10.6	Al seleccionarse una arquitectura de microservicios, Docker es la mejor solución ya que permite que cada uno de los servicios o componentes que se desarrollaron, cuenten con su respectiva configuración independiente para cada uno, además de su fácil escalabilidad, asignación de recursos, entre otras características que ofrece

Angular	Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones SPA [30]	11.0.3	Se escogió Angular por su gran facilidad de uso mediante la creación y distribución de componentes, clases y servicios, permitiendo ahorrar tanto código como tiempo.
SuperCollider	Es una plataforma para síntesis de audio y composición algorítmica. Es un software gratuito y de código abierto disponible para Windows, macOS y Linux [8].	3.11.2	El principal motivo por el que se escogió SuperCollider fue debido a la librería de Python llamada FoxDot, permitiendo generar sonidos musicales de manera rápida y sencilla mediante un lenguaje de programación. Adicionalmente, SuperCollider cuenta con grabación de audio. Este software se escogió entre diferentes opciones como se muestra en la sección herramientas para generación de audio, herramientas para generación de audio, luego de haber realizado una comparación de las características que ofrecen diferentes programas similares.
OpenStack	Es un sistema operativo en la nube que controla grandes grupos de recursos informáticos, almacenamiento y redes en todo un centro de datos, todo administrado a través de un panel de control [31]		Es un sistema de fácil uso que a partir de un dashboard se puede configurar fácilmente los recursos asignados a la instancia que se está utilizando. Además, es el servicio que ofrece la universidad y se encuentra dentro de la red interna de la misma por lo cual el acceso es restringido y mejora la seguridad.
DigitalOcean	DigitalOcean maneja el concepto de droplet para designar a cada uno de los servidores virtuales, los cuales ofrecen en alquiler. No interviene en nada en su instalación y manejo, limitándose a ofrecer imágenes de los principales sistemas		Es un sistema de cloud computing que permite la creación de instancias de sistemas operativos, tal como lo ofrece OpenStack. El ideal de su uso es contar con un servidor de soporte a fallos del servidor principal. Replicando así todo lo necesario para funcionar correctamente en caso de ser requerido.

	operativos junto con sus repositorios de manera local [32].		
PuTTY	PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre licenciado bajo la Licencia MIT [33].	0.75	Este es uno de los programas más usados para acceder de forma remota a otros equipos o servidores, de igual forma de manera directa por comunicación serial. Al ser uno de los más usados demuestra la confiabilidad que ofrece a los usuarios.
MuseScore	es un programa de notación musical para Microsoft Windows, Mac OS X y Linux, que soporta una amplia variedad de formatos de archivo.[22]		Este programa permite convertir un archivo midi a formato musicXml con letra y de esta manera obtener la partitura de voz con letra y notas que posteriormente puede ser usado por el software de síntesis de canto. Para más información de la elección de esta herramienta ver la sección herramientas para la síntesis de voz
ffmpeg	FFmpeg es una colección de software libre que puede grabar, convertir (transcodificar) Incluye libavcodec, una biblioteca de códecs [34].		Este software es necesario para hacer la mezcla de archivos de dos archivos de audio (música y voz) en un único archivo de audio
Sinsy NG	Sinsy es un sistema de síntesis de voz para cantar basado en HMM (*Hidden Markov Model*) [24].		Este sistema se usa para lograr una síntesis de voz para cantar
Tensorflow	Es una librería de código abierto desarrollado por Google para el aprendizaje automático a través de una serie de tareas para sistemas capaces de construir y entrenar redes neuronales. Es capaz de detectar y descifrar patrones y correlaciones, análogos al aprendizaje y		Se eligió usar esta librería debido a su gran comunidad, su capacidad de entrenamiento y manipulación. Además, es una herramienta de código libre. Se conoce que su curva de aprendizaje es más larga frente a otras herramientas como PyTorch pero su capacidad de ajuste y mejora en el rendimiento frente a frameworks como theano y keras permitió tomar dicha herramienta como la más adecuada [37].

	razonamiento usados por los humanos [36].		

Tabla 7 Herramientas y tecnologías

1.1. Herramienta para generación de audio

En la siguiente tabla se explican los criterios tenidos en cuenta para elegir la herramienta para crear sonidos musicales y se asigna una importancia del criterio en base a que tan relevante es para la elección de la herramienta de creación de música. (donde 10 es la mayor importancia y 1 es la menor importancia)

Criterio	Explicación	importancia
Funciona mediante programación	La herramienta permite crear sonidos de manera programática haciendo uso de estructuras de datos y elementos de programación para la creación de algoritmos	10
Funciona mediante interfaz gráfica	La herramienta permite crear sonidos mediante una interfaz gráfica donde el usuario no puede crear sonidos usando un lenguaje de programación	1
Live coding	La herramienta es capaz de crear y reproducir sonidos en tiempo real	5
Facilidad percibida de uso	Representa la complejidad de uso de la herramienta	8
Síntesis de sonidos	La herramienta es capaz de reproducir sonidos de instrumentos musicales	10
Importar sonidos externos	Es posible agregar sonidos externos que no se encuentran dentro de la herramienta	6
Ejecutar varios sonidos simultáneamente	Capacidad de la herramienta para reproducir varios sonidos e instrumentos musicales al mismo tiempo	9
Modificar sonidos en tiempo real	La herramienta es capaz de hacer modificaciones a los sonidos tales como efectos de audio y volumen en tiempo real	7
Señales de control programables	La herramienta es capaz de modificar señales de control de instrumentos mediante lenguaje de programación	5
Definir tempo, velocidad de la música	La herramienta permite asignar un tempo o bpm a los sonidos que genera	9
Crear melodías	La herramienta permite crear melodías monofónicas	10

Crear percusiones	La herramienta cuanta con sonidos de percusiones y permite crear ritmos	10
Efectos en los sonidos	La herramienta permite agregar efectos de audio a los sonidos tales como delays, eco, distorsión, entre otros	7
Grabar	La herramienta permite guardar los sonidos generados en archivos de audios (WAV, mp3)	10

Tabla 8 Descripción de criterios para la generación de audio

En la siguiente tabla se realiza una comparación entre diferentes herramientas que permiten crear sonidos musicales. En base a unos criterios listados en la misma tabla se seleccionó la herramienta que mejor se adecuo a los requerimientos del proyecto.

Criterios	SuperCollider	FoxDot	Pure Data	Stochas	Tone.js	ToneTransfer	Otomata	L2Ork	Earslap	Sonic Pi
Funciona mediante programación										
Funciona mediante interfaz gráfica										
Live coding										
Facilidad percibida de uso										
Síntesis de sonidos										
Importar sonidos externos										
Ejecutar varios sonidos simultáneamente										
Modificar sonidos en tiempo real										
Señales de control programables										
Definir tempo, velocidad de la música										
Crear melodías										
Crear percusiones										
Efectos en los sonidos										
Grabar										

Tabla 9 Estimación herramientas de audio

En base a la tabla 9 estimación de herramientas de audio y la tabla 8 de descripción de criterios para la generación de audio la herramienta seleccionada fue FoxDot ya que cumple con los criterios más importantes y se concluye que es la herramienta que mejor se adopta a la solución propuesta.

1.2. Herramientas para la síntesis de voz

En la tabla 10 se explican los criterios que se tuvieron en cuenta para elegir la herramienta de síntesis de voz. Se asigna una importancia del criterio en base a que tan relevante es para la elección de la herramienta, donde 10 es la mayor importancia y 1 es la menor importancia.

Criterio	Explicación	importancia
Habla español	El sistema es capaz de sintetizar voz en idioma español	10
Software gratuito	La herramienta es gratuita, esto significa que no se requiere un pago o una suscripción para usarlo	6
Software libre	La herramienta ofrece la posibilidad de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el Software	8
Interoperabilidad con lenguajes de programación	El sistema de síntesis de voz se puede integrar de manera sencilla con otros programas computaciones	7
Síntesis de voz	La herramienta sintetiza voz en base a un texto de entrada	5
Síntesis de canto en base a una melodía	La herramienta hace una sinterización de canto	10
Varias opciones de voces	La herramienta tiene varias voces diferentes para hacer una síntesis de voz	6

Tabla 10 Criterios de elección herramienta de síntesis de voz

En la tabla 11 se realiza una comparación entre diferentes herramientas y sistemas de síntesis de voz y canto. En base a los criterios listados en la misma tabla se seleccionó la herramienta que mejor se adecuo a los requerimientos del proyecto.

criterios	vocaloid	espeak	amazon polly	ispeach	sinsy ng	tacotron 2	sinsy web
Habla español							
Software gratuito							
Software libre							
Interoperabilidad con lenguajes de programación							
Síntesis de voz							
Síntesis de canto en base a una melodía							
Varias opciones de voces							

Tabla 11 Estimación de herramientas de síntesis de voz

En base a la comparación de herramientas y sistemas de síntesis de voz la herramienta seleccionada para la síntesis de canto fue Sinsy NG, la cual cumple con todos los criterios definidos y aunque no es la herramienta más moderna y con mayor calidad de síntesis de canto, se destaca principalmente por ser en español, por permitir síntesis de canto y por ser software gratuito y libre, criterios en donde las demás herramientas no destacan.

Además, utilizamos Musecore como programa de notación musical de partituras, esto fue necesario debido a que Sinsy NG requiere de un archivo musicxml como entrada. El programa Musecore es el encargado de transformar el archivo de letra y el archivo midi con la melodía en un archivo musicxml para lograr de esta manera la sinterización de canto. También es un programa fácil integrar con SinsyNG, aunque existen programas con características similares no se tuvieron en cuenta criterios adicionales para elegir uno u otro.

3. Diagrama de flujo de datos

Se usa un diagrama de flujo de datos (DFD) ya que permite demostrar el flujo de la información desde que se recibe la solicitud por la página web hasta que se descargan los archivos.

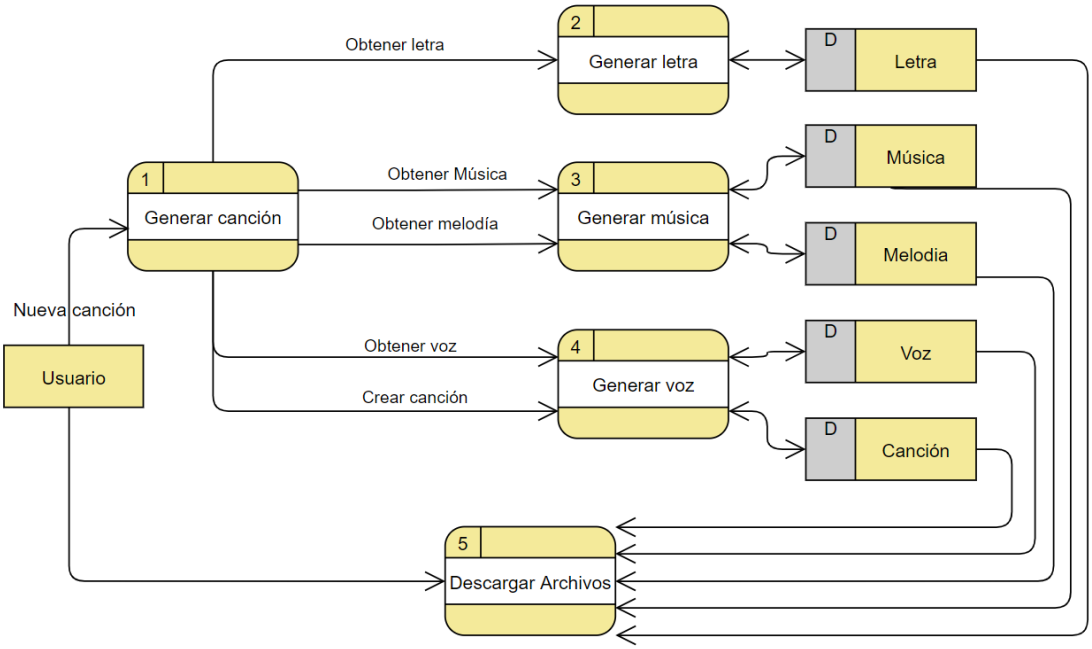

Figura 4 Diagrama de flujo de datos (DFD)

Como se muestra en la figura 4, el primer paso es el de generar una canción, el cual es activado por una página web enviando una solicitud de tipo REST. Esta función realiza 3 integraciones principales: generar letra, música y voz.

El segundo paso siendo la integración con el componente de generación de letra se usa a partir de un servicio web, de igual forma de tipo REST, el cual no requiere parámetros, su implementación es basada en el algoritmo LSTM el cual se desarrolló por medio de la librería Tensorflow y funciona en 2 etapas:

- Entrenamiento:

  Para esto se requirió contar con un conjunto de datos (dataset) de letras de canciones de reggaetón, para esto se realizó una búsqueda de dichas canciones y parte de ellas fueron ajustadas en su formato para el desarrollo, la otra parte se tomó a partir de un dataset existente obteniendo así aproximado de 3150 letras de canciones.

  Luego, con estos datos se hizo un entrenamiento de dicho algoritmo el cual a un nivel general funciona de la siguiente forma:

1. Hace una lectura de archivo de letras, inicialmente de forma parcial y posteriormente se incluyó la totalidad con el avance del análisis de los resultados obtenidos.
2. Cada una de las diferentes palabras encontradas se transforma a una representación numérica, y este número es el que recibe el algoritmo, lo cual lo hace funcional para diferentes soluciones, convirtiendo el dataset de entrada en el formato requerido.
3. Se configuran los parámetros de entrada del algoritmo a partir de prueba y error, analizando los resultados con el fin de encontrar las predicciones más coherentes. Para esto los parámetros que fueron modificados son:

parámetro	definición
input_shape	Define la longitud de los datos de entrada, limitando la cantidad de caracteres de las palabras de entrada [36], logrando así que el sintetizador de voz no fuera afectado.
Dropout	Es un numero en el rango entre 0 y 1, tomando los valores decimales, siendo así el cero el valor más pequeño de perdida de neuronas o patrones definidos, que en el momento de ser menor la generalización de los datos es más exacta, evitando crear múltiples caminos de predicción [36].
Dense	Esto permite definir capas de salida de un entrenamiento, es decir, cada capa cuenta con una configuración y una longitud de datos de salida lo cual es tomado a partir de la diversidad de la información procesada [36], en este caso los diferentes caracteres encontrados en el dataset.
save_best_only	La función de este parámetro es definir si se almacenan todos los archivos binarios de entrenamiento, o de lo contrario solo de se almacena el mejor de ellos.
batch_size	Define la cantidad de muestras a través de las cuales trabajar antes de actualizar los parámetros internos del modelo [36], por ejemplo, para este trabajo de grado, cada 128 muestras actualizan el modelo con las nuevas posibles predicciones
epochs	Hace referencia a la cantidad de veces que hace un análisis completo de todo el dataset [36].
loss	Calcula la medida de perdida que un modelo puede alcanzar al realizar una predicción, con el fin de optimizar su rendimiento [36].

Tabla 12 Parámetros de algoritmo LSTM

- Generación:

  Para la etapa de generación se hace uso de 2 archivos: dataset de letras de canciones y archivo binario de entrenamiento. Tomando el archivo de entrenamiento, se procesa y nos permite generar predicciones, con las cuales se hace una transformación usando dataset para así tener como resultado una palabra, de esta forma se genera palabra tras palabra hasta obtener un texto formado en párrafos. Pero adicionalmente se valida la longitud mínima y máxima de palabras para cada renglón y se ajusta de ser necesario. Finalmente, este resultado se almacena en un archivo de texto.

Una vez se cuenta con el archivo de letra, este podrá ser descargado por el componente de integración, lo que permite continuar con el paso 3, que según se especifica en el diagrama, consiste en generar la parte musical de la canción.

La generación de música inicialmente recibe de una serie de parámetros de entrada, los cuales pueden ser especificados y enviados por el usuario desde la página web, de no ser así se crearán aleatoriamente, tomando en cuenta una serie de criterios y rangos disponibles. Los parámetros ser especifican en la tabla 13, junto a los valores posibles que pueden tener.

Parámetro	Descripción	Valores posibles
bpm	Es el tempo de la canción «Beat per Minute» que define la velocidad de la canción.	Entre 85 bpm a 95 bpm
root	Es la nota principal en la que está la canción, se conoce como nota tónica.	C(Do), D(Re), E(Mi), F(Fa), G(Sol), A(La) y B(Si)
scale	Es la escala musical de la canción	Mayor o menor
n_chords	Cantidad de acordes que tendrá de la canción	Entre 2 a 4 acordes
progression	Es la progresión de los índices de los acordes y se representa como un arreglo. El tamaño de la progresión debe ser igual al valor del parámetro n_chords.	Cada índice tiene como valor un número entre 1 a 7. Se puede definir cualquier combinación de estos acordes.
n_beats	Número de percusiones diferentes de la canción	Entre 1 a 3 beats

structure	Es la estructura de la canción y se representa como un arreglo. Define los momentos en los que los instrumentos serán ejecutados o silenciados.	Los posibles valores son: intro, pre_chorus, chorus, verse y outro. Se puede definir cualquier combinación de estos valores.

Tabla 13 Parámetros para la generación de música

Una vez se definen los valores de los parámetros inicia la funcionalidad de generar música, la cual se divide en dos fases, la primera consiste en construir la parte instrumental de la canción y la segunda en crear una melodía acorde a la parte instrumental, que será utilizada posteriormente para la síntesis de voz.

En la primera fase se genera inicialmente la progresión de los acordes, la cual se representa como un arreglo de tamaño 4 y sus valores (acordes) se definirán tomando en cuenta los parámetros de entrada y la tabla teórica de la progresión.

Clave Mayor	I	II	III	IV	V	VI	VII
A	A	Bm	C#m	D	E	F#m	G#dim
B	B	C#M	D#m	E	F#	G#m	A#dim
C	C	Dm	Em	F	G	Am	Bdim
D	D	Em	G#m	G	A	Bm	C#dim
E	E	F#m	F#m	A	B	C#m	D#dim
F	F	Gm	Am	Bb	C	Dm	Edim
G	G	Am	Bm	C	D	Em	F#dim
Clave Menor	I	II	III	IV	V	VI	VII
Am	Am	Bdim	C	Dm	Em	F	G
Bm	Bm	C#dim	D	Em	F#m	G	A
Cm	Cm	Ddim	Eb	Fm	Gm	Ab	Bb
Dm	Dm	Edim	F#m	Gm	Am	Bb	C
Em	Em	F#dim	G	Am	Bm	C	D
Fm	Fm	Gdim	Ab	Bbm	Cm	Db	Eb
Gm	Gm	Adim	Bb	Cm	Dm	Eb	F

Tabla 14 Circulo armónico

En la tabla 14 se muestran los 7 acordes para cada una de las escalas, tanto mayores como menores, cada acorde representa un conjunto de 3 notas y cada nota representa una posición

numérica de la escala musical, es importante mencionar que los acordes tienen diferentes variaciones y, por lo tanto, los números (notas) no siempre serán los mismo. En resumidas palabras, la progresión consiste en un arreglo de 4 acordes, y cada acorde consiste en un arreglo de 3 números o notas, dando un conjunto total de 12 notas, las cuales serán la base principal de la composición musical.

Una vez se obtiene el conjunto de notas, estas serán utilizadas, mediante la librería FoxDot, para la generación de los sonidos musicales de la canción, entre los que se incluyen acordes, bajos, acompañamientos y la percusión. Cada uno de estos sonidos cuentan con varias alternativas, lo que permite obtener diferentes combinaciones musicales en cada solicitud. Una vez se tengan los sonidos configurados, se procede a ejecutar la música teniendo en cuenta principalmente la estructura recibida como entrada, la cual definirá los momentos exactos en los que los sonidos serán ejecutados o no.

Adicionalmente, la librería de FoxDot permite establecer la conexión con el servicio de SuperCollider mediante el protocolo de sonido OSC. Dentro del servicio se tiene la configuración para realizar la grabación final de la música en un archivo en formato wav.

Al igual que en la primera fase, en la segunda fase también se utilizada el conjunto de notas junto a la estructura, para configurar la secuencia y los tiempos que tendrá cada una de las notas, que serán escritas dentro un archivo en formato mid mediante el uso de la librería midiutil para python . Finalmente, se envían los archivos resultantes de las dos fases, la música en archivo wav y la melodía en archivo mid.

Para la próxima etapa de sintesis de voz, se hace uso de los 3 archivos con los archivos generados hasta el momento (txt, wav y mid) para poder hacer uso de dicho componente, que en el diagrama de flujo de datos puede verse como el paso 4. Este componente funciona tomando el archivo de letra y al archivo de la melodía de voz y usando el programa MuseScore genera un archivo Musicxml que contiene la partitura del canto en donde se le asigna una silaba del texto a cada nota del archivo midi de melodía, esto se hace separando la letra en silabas y asignado estas particiones a las notas del archivo midi, cada línea del archivo de letra corresponde a 2 de los 4 tiempos de un compás del archivo midi. Una vez generado el archivo

musicxml se usa el software Sinsy-NG para sintetizar la voz tomando como entrada el archivo musicxml, el bpm del archivo midi de melodía y el lenguaje, Sinsy NG lo que hace es sintetizar la voz tomando cada silaba y cada nota y sintetizando ese sonido en el tono, con la letra en el lenguaje especificado. Sinsy-NG soporta múltiples idiomas incluido español e inglés ya que hace uso del software de TTS (text to speach) espeak como motor de síntesis de voz.

Como último paso de este componente el archivo de voz generado y el archivo de música, son combinados en un único archivo de audio MP3 haciendo uso del software ffmpeg para este propósito.

Finalmente, se dispondrán los 6 archivos generados durante todo el proceso a usuario para que sean descargados:

1. Lyrics.txt
2. Melodia.mid
3. Música.wav
4. Partitura.xml
5. Canción.mp3
6. Voice.wav

4. Diagrama de componentes

El diagrama de componentes permite demostrar más fácilmente la interconexión que existe entre los diferentes componentes, y facilitar el entendimiento del proceso descrito en la sección 3. Diagrama de flujo de datos.

Se planteó una arquitectura de microservicios, interpretando cada componente como un servicio de los cuales 4 de ellos son destinadas para la parte del backend y uno para la creación de una página web como se explica a continuación:

- MusicGenerator: Se encarga de crear un archivo el cual contiene la música o instrumental de la canción, a partir de ciertos parámetros de entrada y en caso de no hacerlo, se asignarán aleatoriamente, así mismo, se genera un archivo en formato midi el cual contiene la melodía que se le asignará a la voz. Este proceso es realizado

a través de la librería FoxDox para Python que se comunica con la herramienta supercollider.

- LyricsGenerator: Este componente esta desarrollado usando el algoritmo LSTM el cual usando un data set de letras de canciones de reggaetón genera un archivo txt con la letra de la canción.
- VoiceGenerator: Usando los archivos descritos anteriormente (midi, wav y txt), se genera una voz y una canción en formato mp3.
- IntegrationSong: Como su nombre lo indica, haciendo uso de los tres componentes antes mencionados, el componente orquesta la integración con cada uno de estos para entregar como resultado al cliente cada uno de los archivos generados en el proceso.
- SongGeneratorWeb: Es una interfaz gráfica la cual permite hacer uso del sistema de forma sencilla teniendo solo conexión con el componente de integración.

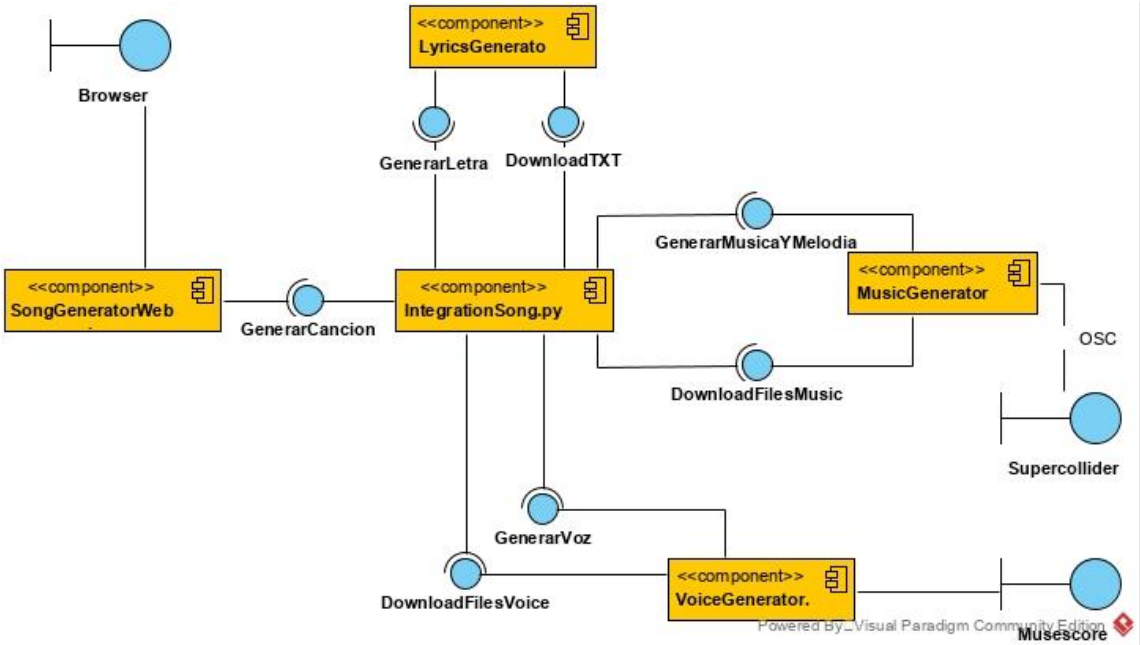

Figura 5 Diagrama de componentes

5. Diagrama de despliegue

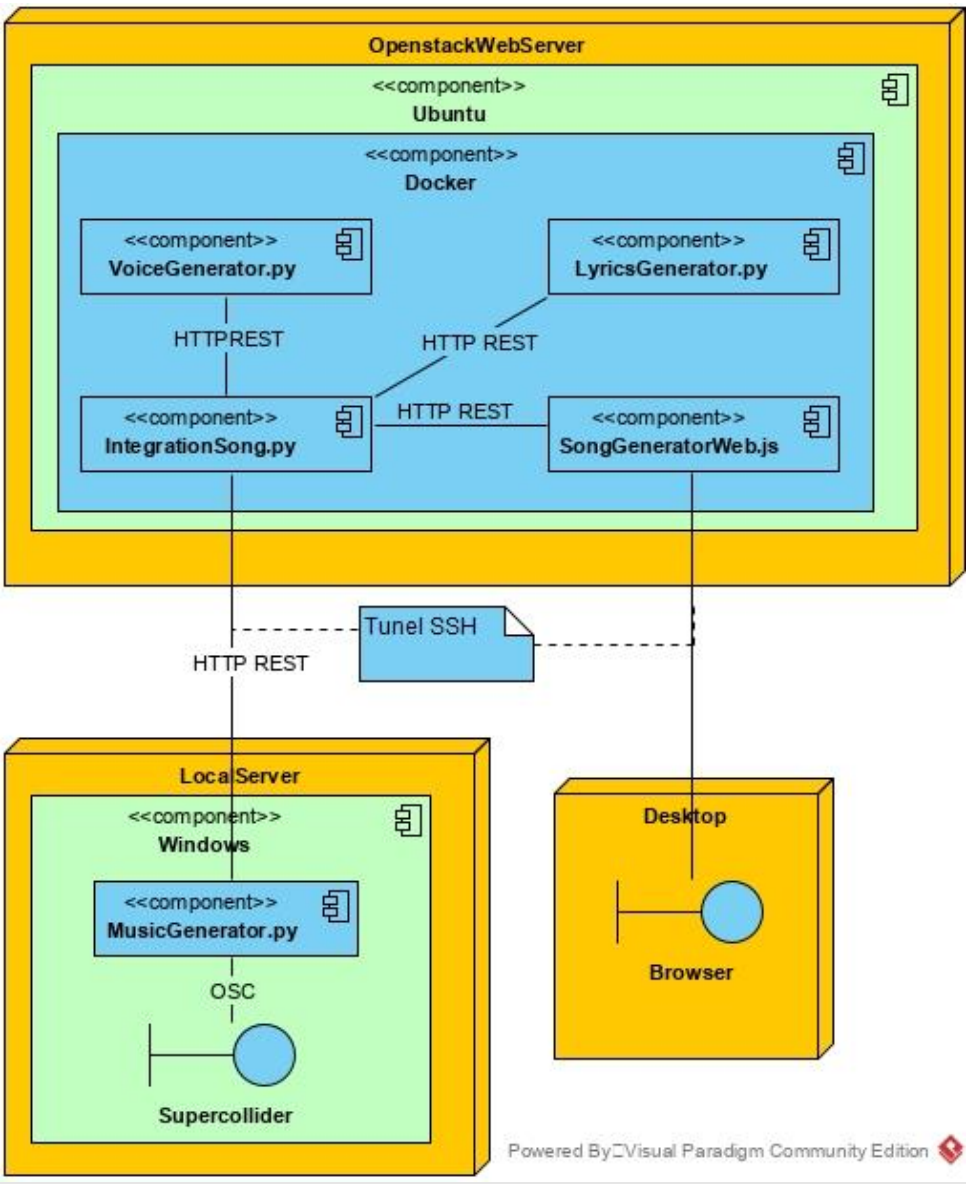

Figura 6 Diagrama de despliegue

Para modelar el estado actual de la herramienta respecto a los ambientes en los cuales se encuentra funcionando, por medio del diagrama de despliegue se muestra su estructura (ver la figura 7). Se cuenta con dos servidores en los cuales están desplegados los diferentes componentes:

- OpenstackWebServer:

La plataforma de computación en la nube OpenStack es ofrecida por la universidad Javeriana, la cual nos permite crear instancias de un sistema operativo con ciertas características de software, para este caso se cuenta con una instancia de Ubuntu 18.04, la cual cuenta con la instalación de Docker. Esto se debe a que como se mencionó en la sección anterior, se implementa una arquitectura de microservicios y a partir de Docker es posible optimizar dicho proceso, como se puede ver en la figura 7. Los componentes de generación de letras, voz, integración y pagina web se encuentran configurados en contenedores Docker comunicándose por medio del protocolo HTTP REST.

- LocalServer:

Este servidor hace referencia a un computador con características básicas con sistema operativo Windows, esto se debe a que el uso de Supercollider está limitado a este sistema operativo, por lo tanto, se decidió que por medio de un túnel SSH con la herramienta PuTTY, fuera posible la conexión y envío de peticiones REST entre componentes, por consiguiente, entre los servidores.

VI- DESARROLLO DE SOLUCIONES

El proceso que usamos para crear la solución está basado en la metodología scrum que es una metodología ágil basada en el estándar Agile Unified Process (AUP), el cual permite adaptarse a los cambios durante el proceso, con el fin mejorar el resultado final enfatizando el trabajo colaborativo y las buenas prácticas [13]. Esta metodología consiste en 4 etapas como se puede ver en la figura 7.

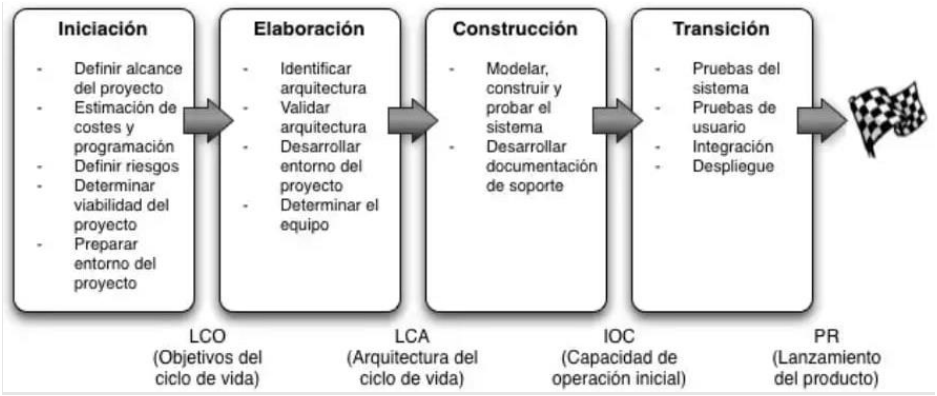

Figura 7 Agile Unified Process

La razón principal de su elección se debe a que es una metodología ágil para el desarrollo de proyectos que permite refinar los objetivos planteados durante el proceso de desarrollo, así como redefinirse para mejorar o facilitar el software, problema con el que cuentan las metodologías tradicionales.

1. Fases metodológicas

1.1. Fase metodológica: concepción

En esta fase identificamos la oportunidad de crear una herramienta de generación de música para el trabajo de grado. Adicionalmente, en esta fase se definieron la propuesta, el alcance y los parámetros para el desarrollo de las próximas fases, permitiendo así la creación de un plan de trabajo [13].

Método: Durante esta fase establecimos el alcance, calendario, riesgos y factibilidad del proyecto.

Actividades

- Exploración del problema
- Definición de la propuesta del proyecto final
- Planteamiento de la solución
- Desarrollo del plan de proyecto
- Especificación de requerimientos

Resultados obtenidos: Como resultado de esta fase, se obtuvieron los siguientes entregables:

- Documento del plan de administración de proyecto de software (SPMP).
- Documento de la especificación de requerimientos de software (SRS).

1.2. Fase metodológica: Elaboración

En esta fase se desarrolló el diseño de la solución y la definición de las pruebas que se realizaron para validar el correcto funcionamiento del sistema [13].

Método: Durante esta fase desarrollamos el diseño de la propuesta a nivel técnico y el diseño de las pruebas.

Actividades

- Elaboración de la arquitectura del software
- Diseño detallado
- Elaboración del plan de pruebas

Resultados obtenidos: Como resultado de esta fase, se obtuvieron los siguientes documentos:

- Documento de la descripción del diseño del software (SDD)
- Plan de pruebas

1.3. Fase metodológica: Construcción

En esta fase realizamos la implementación de la solución a partir del diseño presentado en la fase de elaboración, así como su correspondiente verificación del funcionamiento a partir de lo descrito en el plan de pruebas [13].

Método: Mediante el uso de la metodología Scrum, definimos los requerimientos como historias de usuarios, distribuidas en los diferentes Sprints mediante el orden de priorización definidos por el equipo y avaladas por el product owner que es el responsable de la gestión y maximización del valor del producto, entregando avances al finalizar cada Sprint y realizando una retroalimentación de este (Sprint retrospective) [12].

Para las historias de usuario utilizamos la técnica de planning poker, asignando un peso a la historia de acuerdo con el esfuerzo que se requiere para su desarrollo, por medio de la serie de fibonacci [35]. También se hicieron cambios en la arquitectura y en algunos requerimientos. En total hicimos 5 sprints durante esta fase como se ve en la siguiente grafica.

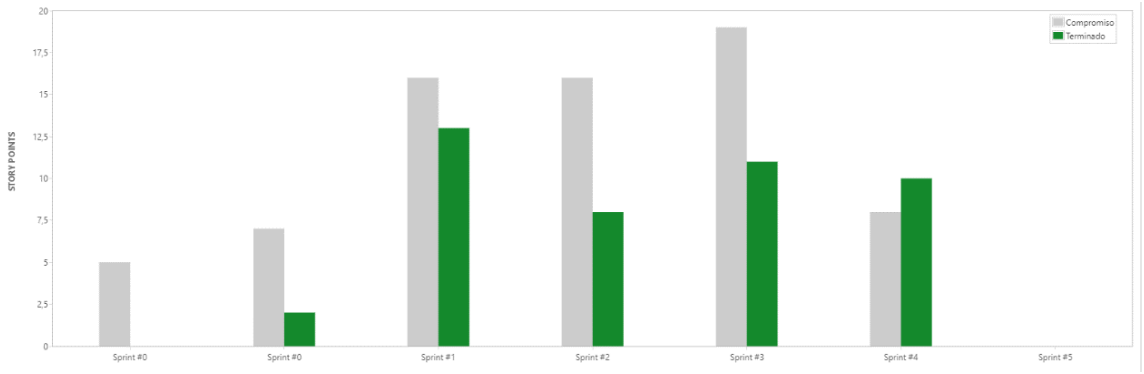

Figura 8 Rendimiento sprints

Actividades

- Sprint planning: Se definieron las historias de usuario a las que se compromete el equipo a desarrollar durante el sprint.
- Sprint: Son iteraciones que constan de 2 semanas donde el equipo desarrollo las historias de usuario.
- Daily: Se realizaron 2 veces a la semana, una los lunes y la otra los viernes.

- Sprint review: Se presentan al product owner los resultados de las historias de usuario comprometidas en el sprint. Esta reunión se hizo una vez por semana.
- Sprint retrospective: Al finalizar cada sprint se realizó una sesión para evaluar los resultados, el estado y satisfacción de los integrantes del equipo, planteamiento de procesos a mejorar.
- Refinamiento: Sesiones donde a partir de los comentarios del product owner ajustamos las historias de usuario y objetivos del sprint.

Resultados obtenidos: Al término de esta fase, el equipo obtuvo los siguientes resultados dados como entregables:

- Código fuente
- Documentación de los componentes desarrollados

1.4. Fase metodológica: Transición

En esta última fase realizamos un proceso de pruebas sobre del prototipo, además se elaborará un informe con los resultados [13] el cual describirá los pasos de la validación de datos de entrada y de salida.

Método: En esta etapa se construye una serie entregables finales que muestran los resultados finales del proyecto

Actividades

- Pruebas de flujo completo del prototipo
- Documentación de resultados en pruebas
- Creación de manuales
- Entrega del proyecto

Resultados obtenidos: Al finalizar esta fase, se obtuvieron los siguientes entregables:

- Prototipo de la solución
- Manual de usuario
- Documento de certificación de calidad
- Memoria del trabajo de grado

2. Prototipo

A partir de lo presentado en el documento, se desarrolló un prototipo funcional que cumple con lo planteado en la sección de objetivo general y objetivos específicos. La herramienta cuenta con las funcionalidades de generar letra, música y síntesis de voz, las cuales pueden ser accedidas por medio de una página web, que permite al usuario interactuar con ellos mediante 3 funcionalidades principales: generación de letra, generación de instrumental y generación de canción.

- **Generar letra:**

Se genera una letra de reggaetón. Este por su parte no requiere parámetros de entrada por lo que la generación de texto (letras de canciones) es totalmente automática. El primer paso es acceder a la página web y seleccionar la opción de generar letra, lo cual acciona el sistema, haciendo el llamado al componente de generación de letra.

Figura 9 Interfaz generar letra

Una vez finalizada la ejecución del sistema, se genera un archivo de texto (txt) con la letra generada y el usuario puede usar la opción de descargar archivo entregando así un el archivo generado y nombrado con un identificador, como se muestra en la figura 10.

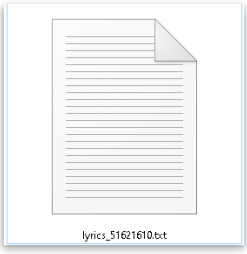

Figura 10 letra en archivo de texto

El contenido de este archivo es el siguiente texto o letra más específicamente.

te nota que te estas viniendo
y que tetra de perderte yo tu
se que que voy me album me
paso e olvidarse hermoso jajajaja
de mas girl que somos robe a
esto soy mismo
a gran se porno solo aprietan
ese animales y hay voy yo poder
y y dale soy
siempre yeah oh el mami de la
lo que puede perdoname me cuerpo im
a prendemo pero buenos enamoro tu
loco dejaron su tiempo
veinte malo el bailando las resuelve
si un voy
nota bailando como el noche literal ca
que es todavia
mejor que va
al iba yo quiero ni calle baby
ozuna canse a usted
a apiade los conmigo
tu maroma con mentir
diamante y tras soy
comeria y yo te acompa ao
y por va
puedo marquesina bebe
and a le mas que solo
ignores en la
la pregunto a carita
los rica jajaja que te motel

- **Generar música:**

Se genera la parte instrumental de una canción de reggaetón ya sea enviando los parámetros Bpm, Root, Scale, n_Scale o Progression. Estos parámetros están descritos en la tabla 8**.** A continuación, se muestra la pantalla que permite accionar esta funcionalidad:

Figura 11 Interfaz generar música

El diligenciamiento del formulario es opcional, si no se especifican los parámetros el sistema asigna parámetros aleatorios entre unas opciones predeterminadas). Cuando el usuario hace click en el botón confirmar el sistema genera un archivo de audio con extensión WAV y un archivo mid con la melodía principal que el usuario puede descargar en la opción descargar música.

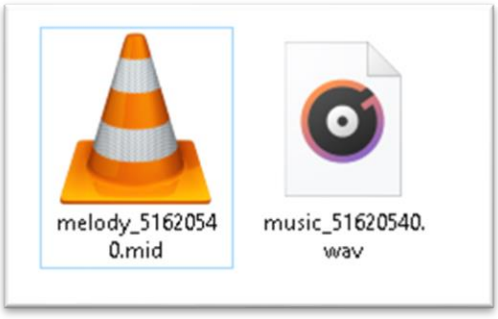

Figura 12 Archivos de música generados

El archivo en formato WAV puede ser abierto por medio de un reproductor de archivos de audio, para este caso se usó uno con la capacidad de mostrar la onda de audio y demostrar como dicha música cuenta con una instrumentación variada.

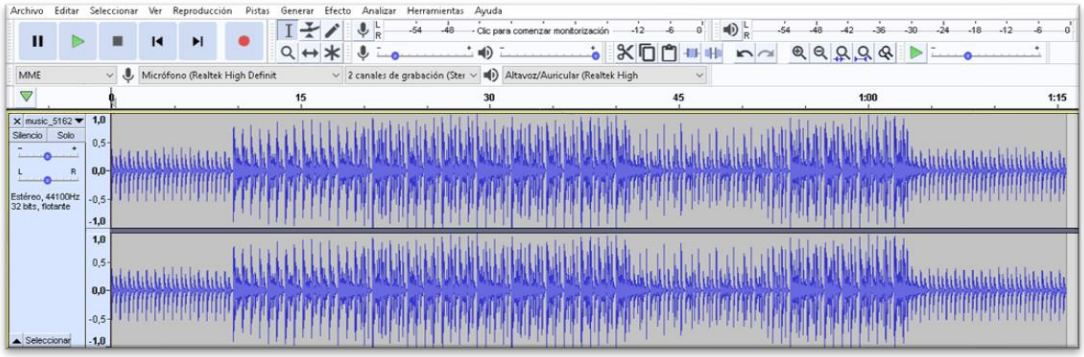

Figura 13 Archivo wav

De igual forma, se muestra el archivo de midi que contiene la melodía principal de la canción:

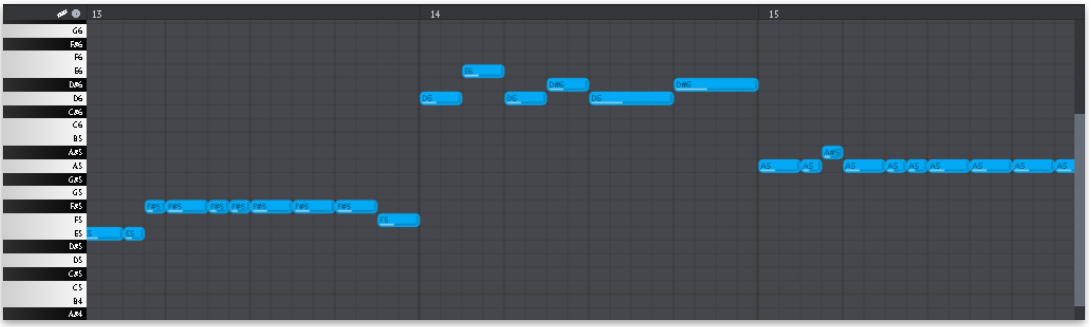

Figura 14 Archivo midi

- **Generar canción:**

Esta funcionalidad es la que cumple con el objetivo general, donde se genera una canción completa de reggaetón haciendo uso de los 3 componentes (música, letra y voz). Su usabilidad consiste en que por medio de la interfaz de la página web, se seleccione la opción de Generar Letra ya sea adjuntando los parámetros disponibles en el formulario, o enviándolos vacíos como se describe en la tabla 8.

Figura 15 Interfaz web

Una vez se ha ejecutado todo el sistema, se contarán con 6 archivos archivos diferentes que fueron necesarios durante el procedimiento y que pueden ser usados por el usuario:

- Lyrics: archivo txt con la letra generada
- Melody: archivo midi con la melodía de la voz
- Music: archivo wav con la música
- Partitura: archivo xml con la partitura de la voz
- Song: archivo mp3 de la canción completa
- Voice: Archivo wav con la voz cantada

Como se puede ver en la figura 16, todos los archivos cuentan con un serial que identifica la generación realizada.

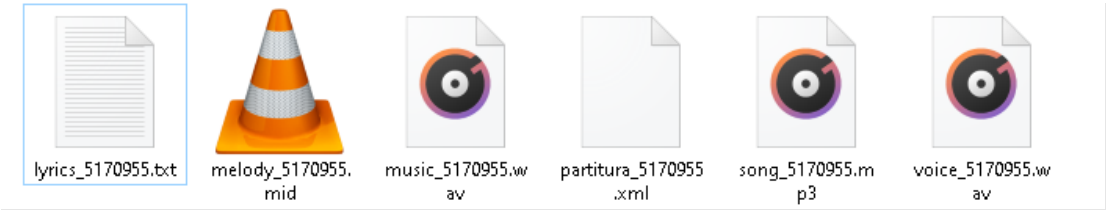

Figura 16 Archivos generados

Para el archivo de texto se puede ver un ejemplo en la sección del prototipó que describe la generación de la letra, mientras que para los archivos de melodía y música se pueden ver en

el apartado de generación de música. Finalmente se muestra el contenido de los archivos no vistos en las secciones anteriores:

- La partitura de la canción muestra como cada letra es representada en una nota musical y de esta forma poder ser cantada por el componente de síntesis de voz.

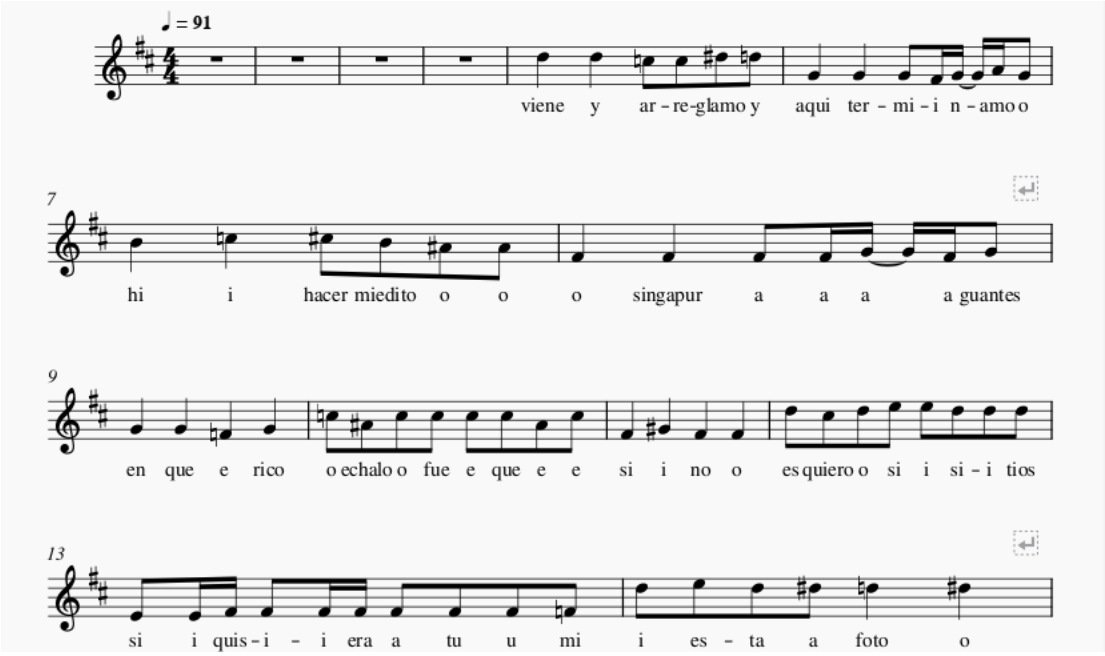

Figura 17 Partitura para voz generado

- Además, se cuenta con el archivo de voz generado a partir de la melodía dada y la partitura

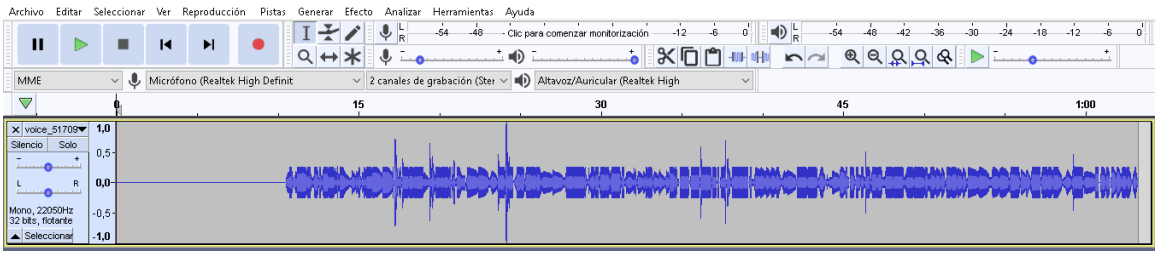

Figura 18 Voz generada

- Finalmente, se cuenta con el archivo de la canción en formato en mp3, el cual integra los resultados de todos los otros archivos.

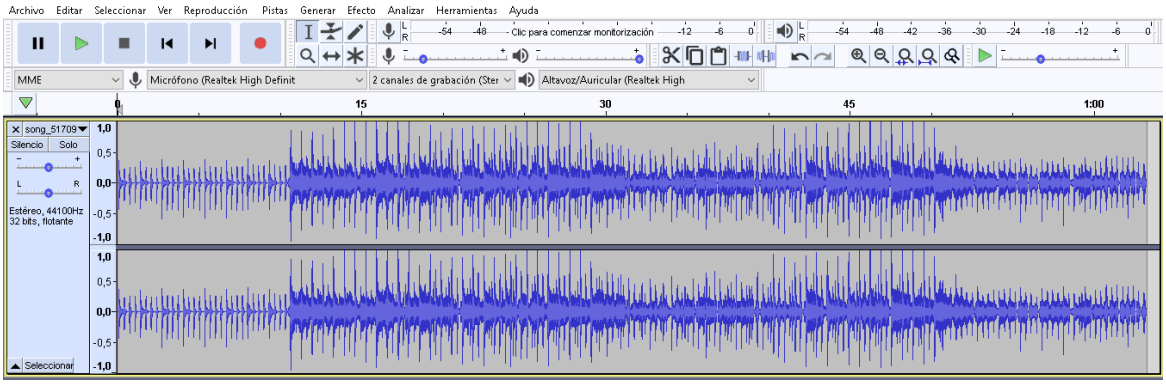

Figura 19 Canción generada

VII- RESULTADOS

Durante la etapa de desarrollo se realizaron una serie de pruebas para asegurar el proceso de control de calidad, estas pruebas se dividen en 4, pruebas funcionales, pruebas unitarias, pruebas de integración de los componentes y pruebas de aceptación. Adicionalmente, se realizó una configuración en el repositorio para tener un control de la calidad del código fuente a nivel de desarrollo seguro.

1.1. Pruebas unitarias

Se realizaron pruebas unitarias manuales para la validación de cada uno de los componentes a nivel de desarrollo. Estas pruebas se realizaron para avanzar y cumplir con las funcionalidades básicas de cada componente, las cuáles se iban presentando en el transcurso de las reuniones con el product owner, donde se obtenía la respectiva retroalimentación de los ajustes que se requerían de ser necesarios.

En este tipo de pruebas no se generó ningún tipo de documento ya que, los avances se entregan semanalmente y su función era mejorar la calidad del software que se estaba desarrollando.

1.2. Pruebas Funcionales

Las pruebas funcionales se realizaron al momento de completar el desarrollo inicial de los componentes. El propósito de estas pruebas era garantizar la generación y el envío de los archivos por cada uno de los componentes, a través del envío de peticiones REST a los servicios, con el fin de garantizar las definiciones de negocio dadas para cada uno de estos.

El proceso consistía en validar como reaccionaba los servicios frente a diferentes datos de entrada que se ajustaban por cada petición y, asimismo ajustar los bugs presentados. En la sección prototipo se puede evidenciar la ejecución del sistema y los resultados obtenidos cumpliendo así los requerimientos descritos.

1.3. Pruebas de Integración

El propósito de las pruebas de integración era validar la correcta comunicación entre los contenedores localizados en el servidor, que contienen el frontend y los componentes de letra y voz, así como el computador local que contiene el componente de música. Estas pruebas se realizaron a partir del envío de peticiones REST al servicio de integración de 2 formas:

- Postman: Validación de peticiones REST en sus datos de entrada y de salida.
- Frontend: A partir de una aplicación web que facilitaba su uso.

Además, se realizó un documento de certificación de pruebas de los componentes integrados de los cuales se definieron unos parámetros de aceptación para cada escenario propuesto como se puede ver en la tabla 15.

Criterio de aceptación
Ejecución exitosa
Response esperado
Uso del request
Descarga de todos los archivos
Duración
Letra
Voz sintetizada corresponde a la letra
¿La voz usa la melodía?
Ritmo del reggaetón (dembow)

Tabla 15 Criterios de aceptación

A continuación, se muestra un ejemplo de los resultados obtenidos.

Escenario de prueba

Los 4 componentes se encuentran activos, 3 de ellos se encuentran activos sobre el servidor Openstack, al cual se accede por medio de VPN, y uno en un equipo de cómputo local. La interconexión de estos se realiza usando un túnel SSH.

La prueba se realiza a partir de Postman, realizando una solicitud REST.

- Tiene parámetros para generación de música.
- No tiene estructura por defecto.

Request

- Url: http://127.0.0.1:8000/song
- Tipo de servicio: [GET]
- Body:


```
{
   "music": {
   "bpm": 90,
   "root": "A",
   "scale": "minor",
   "n_chords": 4,
   "progression": [ 3, 4, 5, 7 ],
   "n_beats": 2
   }
}
```


Response


```
{
  "lyric": "lyrics_51622437.txt",
  "voice": "voice_51622437.vaw",
  "melody": "melody_51622437.mid",
  "music": "music_51622437.vaw",
  "song": "song_51622437.mp3",
  "voicexml": "partiture_51622437.xml"
}
```


- Archivos generados: /resultados/51622437.rar

Análisis

Criterio	Resultado
Ejecución exitosa	Si
Response esperado	Si
Uso del request	Si, para la música. No se envía parámetros de la estructura, por lo que el sistema los genera de forma aleatoria y correcta.
Descarga de todos los archivos	Si. Todos abren correctamente
Duración	3:23 Menor al tiempo promedio de una canción
Letra	• 69 líneas de texto. Longitud correcta • 6 palabras máximo por línea. Longitud correcta
Voz sintetizada corresponde a la letra	Si

¿La voz usa la melodía?	Si
Ritmo del reggaetón (dembow)	Si

Tabla 16 Resultado de certificación de calidad

Los demás escenarios de pruebas se describen en el documento de certificación de calidad, adjunto y los resultados de cada uno de estos se encuentran dentro de la carpeta anexa certificación de calidad.

1.4. Desarrollo Seguro

Es importante recalcar el uso de la herramienta de CI GitHub Actions junto a SonarCloud, que permitieron automatizar los procesos de configuración y control de calidad, garantizando las buenas prácticas y desarrollo seguro en el código fuente cada vez se ejecutaba un "push" en los repositorios. En la figura 20 se puede ver uno de los resultados obtenidos en el proceso de escaneo con sonar cloud.

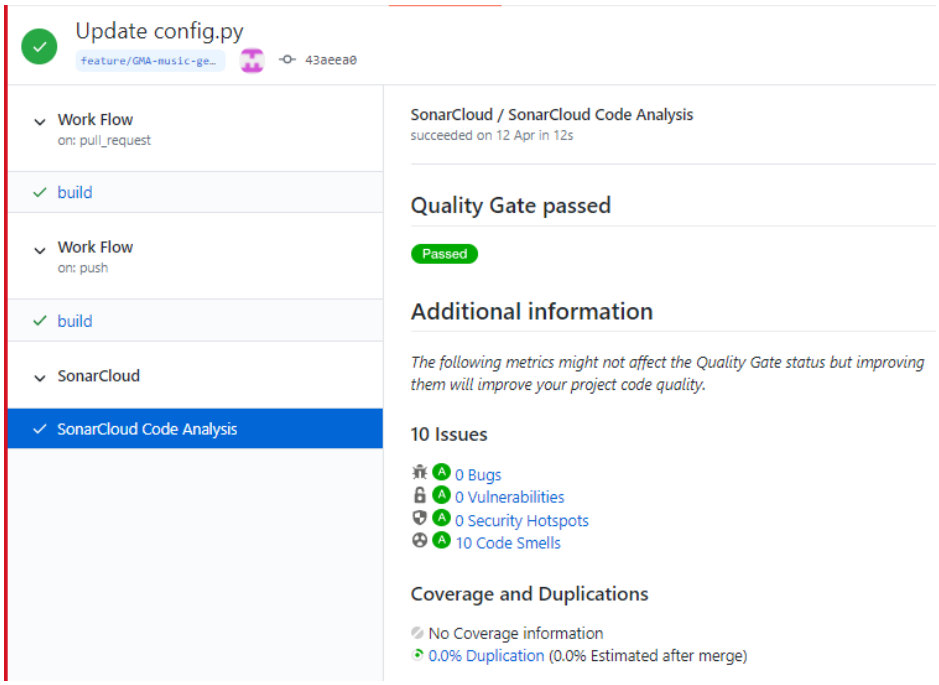

Figura 20 Resultados SonarCloud

Finalmente, para la aprobación de calidad en tanto al desarrollo seguro y buenas prácticas, cada uno de los componentes genera un resultado y aprobación desde la página web de Sonar Cloud como se puede ver en la figura 21 y figura 22.

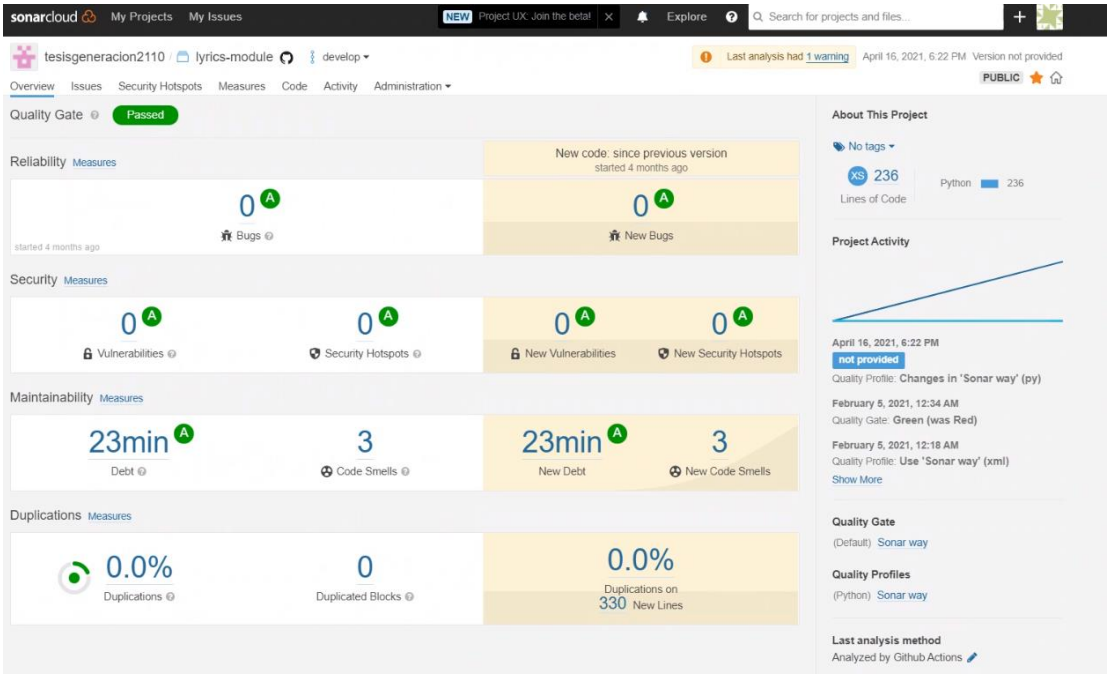

Figura 21 Resultados Sonar Cloud - Lyrics Module

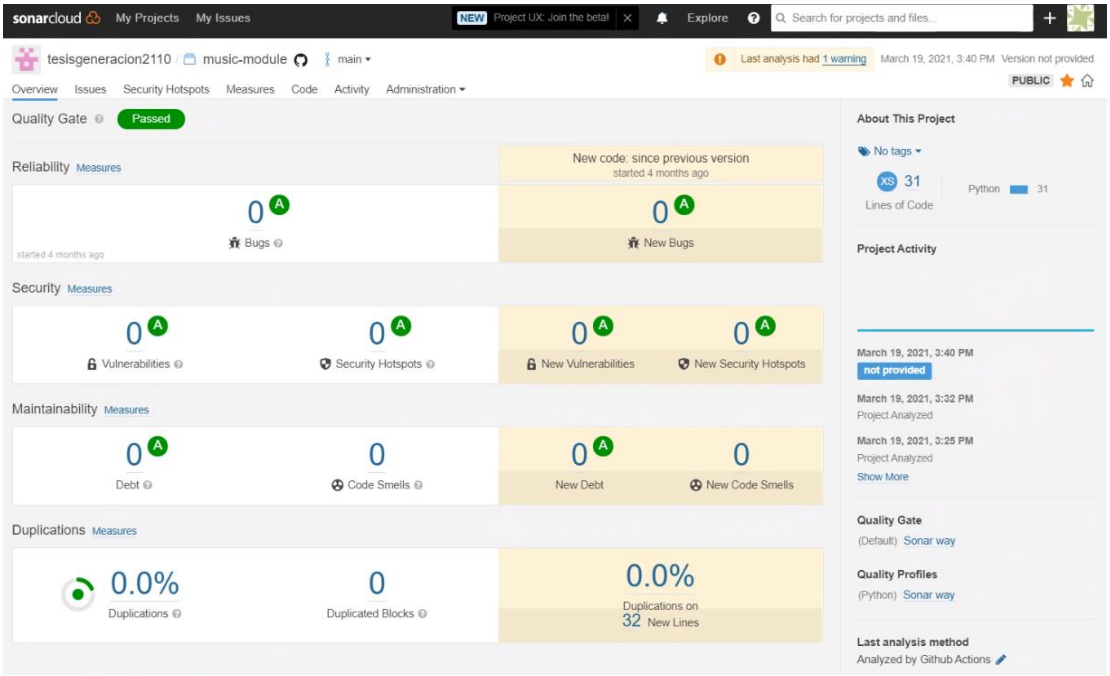

Figura 22 Resultados Sonar Cloud - Music Module

VIII- CONCLUSIONES

1. Análisis de impacto del proyecto

Al iniciar la planeación del proyecto, se realizó una búsqueda sobre la actualidad de la música junto con la automatización, donde se encontraron diversos proyectos que buscaban la automatización e incluso, software muy completo desarrollado por importantes industrias del sector de la tecnología. A partir de esto, fue posible definir una serie de características comunes entre ellos, no solo a nivel tecnológico, sino también a nivel de mercado y usabilidad de los usuarios. Es por esto por lo que podemos definir el impacto del proyecto realizado en diferentes instancias más allá del área de estudio técnica a la que se enfoca el trabajo de grado.

Además, se buscó implementar tecnologías actuales que facilitaron y mejoraron el proceso de desarrollo tanto a nivel de software como a nivel de equipo de desarrollo, ejecutando así temas como SCRUM, Docker, microservicios y técnicas de Deep Learning.

1.1. Impacto a corto plazo

Es importante destacar que el proyecto realizado es un prototipo, por lo tanto, cuenta con diferentes factores que pueden ser optimizados o cambiados dependiendo las definiciones que se le quiera otorgar a la herramienta. Al ser una herramienta nueva y en una versión preliminar, se espera que su uso este dado para pruebas y optimización de este.

1.2. Impacto a mediano plazo

Se espera que la herramienta pueda ser utilizada por músicos que investiguen, trabajen o simplemente sean entusiastas del reggaetón, para un apoyo en su desarrollo y composición musical, teniendo la capacidad de usar los archivos generados por parte de los 3 componentes. De igual forma, se espera que nuevos trabajos de grado o proyectos universitarios mejoren la implementación actual del software, con el fin de optimizar los resultados obtenidos, e incluso aumentar la capacidad de la síntesis de voz y de los géneros que abarca.

1.3. Impacto a largo plazo

A largo plazo, se espera que la herramienta esté disponible a gran escala y que apoye procesos formativos en el ámbito musical. De igual forma, se espera que sea usado en el ámbito empresarial automatizando los procesos musicales, siendo su objetivo apoyar a esta industria en mejorar la calidad de la música que se crea.

1.4. Impacto en la ingeniería de sistemas

Tomando un punto de vista desde la ingeniería, se espera que, como se mencionó en las secciones anteriores, inspire nuevos temas de investigación e implementación, como de realizar mejoras al prototipo creado. Además, que este puede ser abarcado dentro de otras ramas de la ingeniería como lo puede ser la ingeniería de sonido y en la ingeniería electrónica, implementándose en distintos microcontroladores de capacidad limitada en cuando a memoria.

2. Conclusiones

Durante la planeación de este proyecto se identificaron varios trabajos de generación musical con características similares al nuestro. Pero no se identificó ninguno que cumpliera con todos los requisitos propuestos, como se describe en la sección de análisis de contexto, es por esta razón que decidimos llevar a cabo este proyecto para presentarlo como nuestra propuesta de trabajo de grado. Es importante mencionar también que fue necesario realizar una previa investigación sobre los diferentes conceptos de teoría musical, tales como la melodía, ritmos, escalas, entre otros.

Una vez planteada la propuesta, se definieron un conjunto de objetivos específicos que se deberían cumplir para lograr el objetivo general. El primer objetivo específico se relaciona con el componente de generación de música, que hoy en día es capaz de crear una pieza musical que contiene ritmo, melodía y armonía propios del género de reggaetón, además de que es capaz de generar una melodía para la voz en formato midi.

Por otra parte, el componente de generación de letra es capaz de crear una nueva letra a partir del entrenamiento del algoritmo LSTM, utilizando un data set de letras de canciones de reggaetón, cumpliendo con el segundo objetivo específico.

Continuando con el tercer objetivo específico, el componente de generación de voz es capaz de generar una partitura de la voz en formato musicxml, a partir de una letra en formato txt y una melodía en formato midi. Con el uso de la partitura permite generar un nuevo archivo de audio en formato wav con la voz cantada. Adicionalmente, este componente es capaz de unificar los archivos de música y de voz en uno solo, formando la canción en formato mp3.

Finalmente se encuentran las pruebas realizadas en cada uno de los componentes, así como en el componente de integración. Estas pruebas arrojaron los resultados esperados, permitiendo cumplir con el último objetivo específico y al mismo tiempo, con el objetivo general del proyecto. El cual consistía en generar una canción de reggaetón de manera semiautomática implementando los 3 componentes, y que pudieran cooperan entre si mediante un componente de integración. Actualmente, el software desarrollado es capaz de generar una canción de reggaetón acorde a las características definidas de este género.

Del desarrollo del proyecto se obtienen varias lecciones aprendidas, por un lado, es muy importante contar con un buen dataset de datos al momento de trabajar con redes neuronales, más específicamente, con el algoritmo LSTM. Ya que utilizamos un dataset muy extenso de 3150 letras de canciones, sin embargo, se presentaron algunos errores en la generación de letra debido a fallos en la ortografía y sintaxis de estas letras.

Por otro lado, se encuentra el equipo de cómputo, al momento de probar fue evidente que para realizar un entrenamiento extenso de machine learning, se requiere una máquina con altas capacidades, principalmente de memoria y procesamiento. Es por esta razón se debió cambiar de una máquina de escritorio convencional a un servidor con más poder de cómputo, que permitió ejecutar entrenamientos con mayores iteraciones y al mismo tiempo obtener mejores resultados de generación.

Con respecto a la síntesis de voz, fue complicado encontrar herramientas que trabajaran con voces en idioma español, ya que los sistemas más avanzados de este concepto funcionan principalmente en idioma inglés, japones y chino.

Finalmente, al momento de desplegar un software en un servidor, es importante verificar si todas las herramientas y tecnologías pueden ser ejecutadas con éxito en ese servidor. En el

proyecto se tuvo problemas al intentar ejecutar SuperCollider debido a que los sistemas operativos desde servidores no cuentan con una interfaz gráfica requerida por SuperCollider para su correcto funcionamiento, por tal se tuvo que desplegar en una maquina local de uno de los integrantes del equipo.

3. Trabajo futuro

Se debe tener en cuenta que el desarrollo del proyecto se realizó en un tiempo limitado en el cual el alcance debía ser bien definido para lograr los objetivos planteados, a pesar de esto, el proyecto puede expandirse a nuevas funcionalidades. Como trabajo futuro existen varias oportunidades para mejorar el sistema e implementara nuevas funcionalidades.

1. Calidad de las letras generadas: Actualmente, el componente de generación de letra fue entrenado con un dataset de 3150 letras de canciones de reggaetón con algunos errores en su semántica y ortografía, dado que se usó un dataset ya creado y no tenía un formato definido, la mejora de esta información puede lograr que al entrenar el algoritmo con un data set de mayor calidad se obtenga mejores resultados. Además, la ejecución de más iteraciones al momento del entrenamiento también puede mejorar la calidad de las letras de las canciones.
2. Diseño de la página web: El enfoque actual de la tesis no se basaba en la página web, ya que el objetivo de esta era facilitar el uso del usuario final, pero esto abre la oportunidad de que sea posible realizar grandes mejoras a esta, haciéndola más amigable para el usuario.
3. Voz: el componente de voz actualmente no reproduce el canto de una voz muy similar a la humana y suena artificial, por lo que como trabajo futuro se propone buscar o implementar un software de síntesis de voz que genere una voz cantada en español de mejor calidad, es decir, más natural y parecida a la voz humana.
4. Agregar sonidos, ritmos y patrones: Se pueden agregar más patrones, melodías, ritmos y sonidos al componente de música esto con el fin lograr generar canciones más diversas y con posibles nuevos ritmos que se usen en el género musical.

5. Otros géneros musicales: La implementación del sistema se basa en generar únicamente canciones de reggaetón y es posible hacer la implementación de más géneros musicales, con un dataset de letras de canciones de otros géneros e implementando los algoritmos de generación musical en base a la teoría y patrones de otros géneros.

IX- REFERENCIAS

[1] P. Mackworth, y Goebel «which provides the version that is used in this article. These authors use the term "computational intelligence" as a synonym for artificial intelligence», 1998.

[2] A. Pannu, «Artificial Intelligence and its Application in Different Areas», vol. 4, n.º 10, p. 6, 2015.

[3] C. Roads, «Research in music and artificial intelligence», *ACM Comput. Surv.*, vol. 17, n.º 2, pp. 163-190, jun. 1985, doi: 10.1145/4468.4469.

[4] «What is a Song? - Definition & Examples. » *Study.com*, 31 May 2017, https://study.com/academy/lesson/what-is-a-song-definition-examples.html

[5] A. Suprem y M. Ruprem, «A New Composition Algorithm for Automatic Generation of Thematic Music from the Existing Music Pieces», p. 5, 2013.

[6] P. Gervás Gómez-Navarro, «Un modelo computacional para la generación automática de poesía formal en castellano», *Procesamiento del lenguaje natural, nº 26 (septiembre 2000); pp. 27-34*, 2000.

[7] P. D. Agüero, «Síntesis de voz aplicada a la traducción voz a voz», 2012.

[8] The SuperCollider Book | The MIT Press». https://mitpress.mit.edu/books/supercollider-book (accedido may 24, 2021).

[9] «'Daddy's Car' Is A Pop Song Composed By Artificial Intelligence», *Flow Machines*. https://www.flow-machines.com/history/press/daddys-car-pop-song-composed-artificial-intelligence/ (accedido nov. 02, 2020).

[10] B. Iturbe, «Estilo musical: reggaetón», *Padres y Maestros / Journal of Parents and Teachers*, n.º 315, Art. n.º 315, 2008, Accedido: nov. 02, 2020. [En línea]. Disponible en: https://revistas.comillas.edu/index.php/padresymaestros/article/view/1548.

[11] T. Rivero Pérez y others, «Filosofía y Reggaetón: una perspectiva interseccional», 2020.

[12] K. Schwaber, «Scrum development process», en Business object design and implementation, Springer, 1997, pp. 117–134.

[13] S. Gasik, «Comparison of ISO 21500 and PMBOK® Guide», Obtenido de www. sybena. pl/dokumenty/ISO-21500-and-PMBoK-Guide.pdf, 2013.

[14] G. Ozcan, C. Isikhan, y A. Alpkocak, «Melody extraction on MIDI music files», en Seventh IEEE International Symposium on Multimedia (ISM'05), 2005, p. 8 pp.-, doi: 10.1109/ISM.2005.77.

[15] xphnx, «La música: ritmo, melodía y armonía. Fundamentos.», *Colaboratorio*, ene. 17, 2017.

[16] «El 'dembow': el género que divide a los dominicanos | Cultura | EL PAÍS». https://elpais.com/cultura/2013/09/17/actualidad/1379427097_644929.html (accedido may 24, 2021).

[17] «FoxDot | Home». https://foxdot.org/ (accedido may 24, 2021).

[18] «Phonic Character Recognition Based on Multi-media WAV File--《Computer Engineering》2000年11期». https://en.cnki.com.cn/Article_en/CJFDTotal-JSJC200011049.htm (accedido may 24, 2021).

[19] S. Hochreiter y J. Schmidhuber, «Long Short-term Memory», *Neural computation*, vol. 9, pp. 1735-80, dic. 1997, doi: 10.1162/neco.1997.9.8.1735.

[20]P. A. Taylor, *Text-to-speech synthesis*. Cambridge, UK; New York: Cambridge University Press, 2009.

[21]«eSpeak: Speech Synthesizer». http://espeak.sourceforge.net/ (accedido may 24, 2021)

[22]«MuseScore, A Free Open-Souce Music Composition And Notation Program | MusTech.Net - Music Education & Technology». https://mustech.net/2008/10/musescore-a-free-open-souce-music/ (accedido may 24, 2021).

[23]«MusicXML for Exchanging Digital Sheet Music», MusicXML. https://www.musicxml.com/ (accedido may 24, 2021).

[24]«Sinsy». http://sinsy.sourceforge.net/ (accedido may 24, 2021).

[25]O. Vechtomova, H. Bahuleyan, A. Ghabussi, y V. John, «Generating lyrics with variational autoencoder and multi-modal artist embeddings», p. 5.

[26]M. N. Cáceres, H. R. Oliveira, y J. M. Rodríguez, «Towards an automated composer of popular Spanish songs: integrating a music generator and a song lyrics generator»

[27]X. Gao, X. Tian, Y. Zhou, R. K. Das, y H. Li, «Personalized Singing Voice Generation Using WaveRNN», en *Odyssey 2020 The Speaker and Language Recognition Workshop*, nov. 2020, pp. 252-258, doi: 10.21437/Odyssey.2020-36

[28]Python.org. (2018). Welcome to Python.org. [online] Available at: https://www.python.org/about/ [Accessed 13 Aug. 2018]

[29]«About Docker - Management & History | Docker». https://www.docker.com/company (accedido may 24, 2021).

[30]«Angular». https://angular.io/ (accedido nov. 13, 2020)

[31]«OpenStack Docs: Wallaby». https://docs.openstack.org/wallaby/ (accedido may 24, 2021).

[32]«Droplets on DigitalOcean - More than just virtual machines». https://web.archive.org/web/20180623173511/https://www.digitalocean.com/products/droplets/ (accedido may 24, 2021).

[33]«PuTTY Licence». https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html (accedido may 24, 2021).

[34]«FFmpeg». https://www.ffmpeg.org/ (accedido may 24, 2021).

[35]V. Mahnič y T. Hovelja, «On using planning poker for estimating user stories», Journal of Systems and Software, vol. 85, n.º 9, pp. 2086–2095, 2012.

[36]M. Abadi *et al.*, «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems», p. 19.

[37]I. M. González, «Comparación del rendimiento de plataformas para el desarrollo de sistemas de aprendizaje profundo», p. 80.

X- APÉNDICES

1. Versión final de la propuesta
2. Plan de proyecto
3. SRS
4. ECTC
5. SDD
6. Manual de usuario – Backend
7. Manual de usuario – Frontend
8. Certificación de calidad