# Playlist creation with voice recognition

## Idea

Una aplicación la cual con tu voz pueda crear una playlist con canciones que tengan que ver con los temas que le digas por ejemplo “Quiero una playlist que tenga canciones felices para un viaje largo en coche” o “Quiero una playlist que tenga canciones parecidas a Atmosphere de Fisher” y que se guarde dentro de la aplicación.

## Propósitos

* Tener la facilidad de crear nuevas playlist personalizadas con tu voz
* Conocer nuevas canciones mediante series de instrucciones
* Hacer que el usuario tenga una experiencia única por cada playlist

## Objetivos

* Juntar voice recognition con la creación de playlist con API´s
* Uso de IA entrenadas para hacer decisiones más especificas
* Crear playlists con canciones que realmente le interesen y gusten al usuario
* Juntar canciones que cumplan con las necesidades que especifique el usuario

## Funcionalidad

**1. Entrada del usuario: Comandos de voz**

* **Reconocimiento de voz:** La app debe integrar una API o servicio que convierta los comandos de voz en texto. Servicios como Google Speech-to-Text, Azure Speech Services o iOS Speech Recognition pueden ser utilizados para esto.
* **Comandos claros y naturales:** Los usuarios deben ser capaces de usar comandos simples como:
  + "Hazme una playlist de música relajante para estudiar."
  + "Quiero una lista de canciones para hacer ejercicio."
  + "Ponme música de los 80 para una fiesta."
* **Personalización avanzada:** La app podría permitir comandos más detallados:
  + "Quiero una playlist con música rock de los 90."
  + "Crea una lista con mis artistas favoritos."
  + "Hazme una playlist de 30 minutos con música indie para correr."

**2. Procesamiento de la solicitud**

* **Análisis de intención:** Implementar un sistema de procesamiento de lenguaje natural (NLP) para entender la intención del usuario a partir del comando de voz.
  + **Identificación de géneros, artistas y estados de ánimo:** El sistema debería identificar si el usuario quiere un tipo de música específico (por ejemplo, "rock"), un artista en particular, o una playlist para una actividad o estado de ánimo (por ejemplo, "música para relajarse").
  + **Duración o número de canciones:** El usuario podría especificar la duración de la playlist o el número de canciones, y la app deberá ajustar la lista en función de eso.

**3. Generación de la playlist**

* **Acceso a una base de datos musical:** La app debe estar conectada a una API de música, como Spotify, Apple Music, Deezer o YouTube Music, para obtener canciones que correspondan a los criterios del usuario.
  + **Filtrar canciones según el estado de ánimo, género o artista:** La app debe generar listas de reproducción con base en los parámetros que el usuario ha proporcionado.
* **Algoritmos de recomendación personalizados:**
  + **Historial del usuario:** Utilizar el historial de reproducción del usuario para recomendar canciones que se alineen con sus gustos previos.
  + **Machine Learning:** Un sistema de recomendación que mejore con el tiempo basándose en las preferencias y los patrones de comportamiento del usuario.
  + **Canciones populares o nuevas:** La app podría recomendar canciones nuevas o populares que coincidan con los intereses del usuario.

**4. Interacción con el usuario: Feedback y ajustes**

* **Respuesta y confirmación:** Una vez que la app haya generado la playlist, debería confirmar con el usuario algo como: "He creado una playlist de 30 minutos con canciones indie. ¿Quieres hacer algún cambio?"
* **Modificaciones por voz:** Permitir ajustes mediante nuevos comandos de voz, como:
  + "Cambia las últimas dos canciones."
  + "Agrega más música de rock."
  + "Pon canciones más rápidas."

**5. Funciones adicionales**

* **Sugerencias automáticas:** La app podría proponer playlists automáticamente basándose en el momento del día, la ubicación o el comportamiento previo del usuario.
* **Sincronización con dispositivos:** Ofrecer integración con altavoces inteligentes (Google Home, Alexa) u otros dispositivos para que los usuarios puedan controlar la app sin necesidad de usar sus manos.
* **Compartir playlists:** Dar la opción de compartir la playlist generada en redes sociales o con otros usuarios de la app.

**6. Retroalimentación de la experiencia**

* **Mejora continua:** Implementar un sistema de feedback donde el usuario pueda calificar las playlists creadas para mejorar los algoritmos de recomendación.

**Ejemplo de flujo funcional:**

1. **Usuario:** "Hazme una playlist para hacer ejercicio."
2. **App (Reconocimiento de voz):** Transcribe el comando y lo analiza.
3. **App (NLP):** Identifica la intención ("playlist de ejercicio"), duración (una hora) y puede sugerir música energizante.
4. **App (Generación de playlist):** Crea la playlist utilizando los datos del usuario y el historial de reproducción.
5. **App (Confirmación):** "He creado una lista de reproducción con música para hacer ejercicio durante una hora. ¿Te gustaría cambiar algo?"
6. **Usuario:** "Sí, añade canciones de rock."
7. **App:** Ajusta la playlist y reproduce.

## Necesidad

En la actualidad, los usuarios de servicios de streaming musical enfrentan el reto de crear playlists personalizadas que se adapten a sus gustos, estados de ánimo o actividades. Con el vasto catálogo de música disponible en plataformas como Spotify, seleccionar canciones de manera manual puede resultar tedioso y consumir mucho tiempo. Además, aunque existen asistentes de voz que permiten la reproducción de música, la falta de personalización y el esfuerzo requerido para encontrar la música adecuada siguen siendo un problema.

Este proyecto busca abordar esta necesidad creando una aplicación que, a través de comandos de voz, permita a los usuarios generar playlists personalizadas de forma rápida y automática, mejorando la experiencia de descubrimiento musical y ofreciendo una solución sencilla y accesible para la creación de listas de reproducción personalizadas en tiempo real.

## Plan de proyecto

**1. Definición del Proyecto**

**Objetivo:**

Desarrollar una aplicación donde los usuarios puedan crear playlists personalizadas mediante comandos de voz, con un frontend en **React** y un backend en **Python**.

**Alcance:**

* Funcionalidad básica: Crear playlists basadas en comandos de voz.
* Backend en Python para procesar las peticiones y conectar con APIs de música.
* Frontend en React para la interfaz de usuario.
* Integraciones: APIs de reconocimiento de voz y servicios de música.

**2. Estructura de Desglose del Proyecto (EDT)**

**Fase 1: Análisis y Diseño**

* **Tareas:**
  + **Requisitos iniciales:** Define las características básicas que tendrá tu app, como los tipos de comandos de voz que quieres soportar.
  + **Wireframes:** Haz bocetos simples de cómo quieres que se vea la interfaz en React. Puedes usar herramientas como Figma para visualizarlo.
  + **Arquitectura técnica:** Planifica cómo se comunicarán React y Python. Por ejemplo, si usarás **Flask** o **FastAPI** en el backend.

**Fase 2: Desarrollo Backend en Python**

* **Tareas:**
  + **Configuración del servidor:** Elige y configura el framework de backend (Flask o FastAPI).
  + **Integración con APIs de voz:** Implementa el reconocimiento de voz utilizando servicios como **Google Speech-to-Text** o **Azure Cognitive Services**.
  + **Algoritmo de recomendación:** Desarrolla una lógica simple para recomendaciones basadas en géneros o artistas.
  + **Integración con API de música:** Conéctate a Spotify u otro servicio mediante sus APIs para obtener las canciones.

**Fase 3: Desarrollo Frontend en React**

* **Tareas:**
  + **Interfaz de usuario:** Crea la estructura de la app en React, incluyendo formularios o botones para iniciar los comandos de voz.
  + **Interacción con el backend:** Configura las llamadas a tu API de Python desde React, usando **fetch** o **Axios** para hacer solicitudes HTTP.
  + **Manejo de estados:** Utiliza **React Hooks** o **Redux** para gestionar el estado de la aplicación.

**Fase 4: Integración y Pruebas**

* **Tareas:**
  + **Integración Frontend y Backend:** Asegúrate de que React y Python se comuniquen sin problemas. Prueba la creación de playlists desde la interfaz.
  + **Pruebas de usuario:** Simula varios escenarios de usuario para verificar que los comandos de voz se interpretan correctamente y que las playlists se generan.
  + **Pruebas de rendimiento:** Asegúrate de que la app es eficiente en cuanto a tiempos de respuesta y carga.

**Fase 5: Lanzamiento**

* **Tareas:**
  + **Despliegue:** Publica tu backend en un servicio como **Heroku** o **AWS** y tu frontend en **Netlify** o **Vercel**.
  + **Feedback inicial:** Lanza la app a un grupo pequeño de usuarios o amigos y recoge retroalimentación para mejorar.

**3. Tecnologías a Utilizar**

* **Frontend (React):**
  + React, React Hooks, Axios/Fetch para la comunicación con el backend.
  + HTML/CSS para la interfaz.
* **Backend (Python):**
  + **Flask** o **FastAPI** para la API.
  + **Requests** para conectarse a APIs externas (por ejemplo, API de Spotify).
  + **SpeechRecognition** o APIs externas para el reconocimiento de voz.
* **Base de Datos:**
  + Si necesitas almacenar datos del usuario, puedes usar **SQLite** o **PostgreSQL**.
* **APIs de Música y Voz:**
  + **Spotify API** para la música.
  + **Google Cloud Speech-to-Text** o **Azure Cognitive Services** para el reconocimiento de voz.

**4. Gestión de Riesgos Simplificada**

* **Reconocimiento de voz poco preciso:** Empieza con comandos básicos y ajusta según avances.
* **Problemas con la integración de APIs:** Comienza integrando primero una API (por ejemplo, Spotify) antes de agregar más.
* **Tiempo limitado:** Enfócate en la funcionalidad básica y considera hacer versiones posteriores con características más avanzadas.