UNIVERSIDAD LUTERANA SALVADOREÑA

FACULTAD DE CIENCIA DEL HOMBRE Y LA NATURALEZA

LICENCIATURA EN CIENCIAS DE LA COMPUTACION

CICLO 2 - 2023

CÁTEDRA: INGENIERIA DE SOFTWARE

DOCENTE: INGA. LESBIA MANCIA

TEMA: PROYECTO FINAL

INTEGRANTES:

KENIA DALILA CAMPOS RAMIREZ CARNET: CR23130

DIEGO ANDRES MIRANDA ROGEL CARNET: MR2327

JASON ADILMAN SERPAS MARTINEZ CARNET: SM01137241

SANDRA MARISOL TINO ALFARO CARNET: TA2363

SAN SALVADOR 25 NOVIEMBRE 2023

Tabla de contenido

Introducción

En esta etapa final se ha creado la estructura completa de la aplicación de reproductor de música interactivo. Se ha diseñado una interfaz gráfica simple utilizando la biblioteca PySimpleGUI y se han implementado funcionalidades esenciales para cargar y reproducir música en formato MP3. La interfaz de usuario incluye botones para seleccionar una carpetade música, seleccionar una canción específica, reproducir/pausar la música, avanzar a la siguiente canción y retroceder a la canción anterior, y la opción de minimizar la ventana para poder hacer otras actividades en el ordenador.

Objetivo general y objetivos específicos

El objetivo general de esta etapa final; La aplicación del reproductor de música ya finalizada, y funcionando en su totalidad. Permitiendo a los usuarios cargar y reproducir canciones en formato MP3 de manera interactiva.

objetivos específicos: podemos ver el diseño la interfaz gráfica utilizando PySimpleGUI, que incluye elementos como botones, texto y una imagen de fondo. En esta etapa final ya está Implementada la funcionalidad de usuario y contraseña para dar al usuario una confiabilidad del reproductor donde solo una persona tendrá acceso a la carpeta de música, donde se busca las canciones en formato MP3; que se muestran en la lista de canciones disponibles. El usuario podrá implementar la funcionalidad para cargar, la canción que desea escuchar, teniendo la satisfacción de escuchar su música favorita.

Tema del Proyecto:

Desarrollo de un Reproductor de Música Interactivo en Python.

Descripción del proyecto. Descripción: El proyecto se centra en la creación de un reproductor de música interactivo utilizando el lenguaje de programación Python y las bibliotecas pygame y tkinter. El objetivo principal de este reproductor de música es proporcionar a los usuarios una interfaz gráfica intuitiva y funcional para cargar, reproducir y gestionar su colección de música en formato MP3.

Justificación del proyecto.

Aprendizaje y Desarrollo Personal: Crear un reproductor de música es un proyecto práctico que puede ayudarte a aprender y mejorar tus habilidades de programación en Python. Te permite aplicar conceptos de programación y trabajar con bibliotecas relacionadas con el audio.

Personalización:

Un reproductor de música desarrollado por ti mismo te brinda la oportunidad de personalizarlo según tus preferencias y necesidades. Puedes agregar funciones específicas que desees y adaptarlo a tu gusto.

Proyecto de Estudio: Puede ser un proyecto académico o de estudio que te permita profundizar en la programación en Python, así como en la manipulación de archivos de audio y la interfaz de usuario.

Desarrollo de Habilidades en Python: Desarrollar un reproductor de música implica trabajar con varias bibliotecas y módulos en Python, lo que puede ayudarte a mejorar tus habilidades en este lenguaje.

Listado de requerimientos.

Requerimientos funcionales.

1. El usuario podrá visualizar la interfaz del reproductor.

1.1 El sistema permitirá que el usuario pueda visualizar el logo del reproductor al iniciar.

1.2 El sistema permitirá que el usuario pueda observar las diferentes funciones que tendrá el reproductor.

1.3 El sistema permitirá al usuario iniciar sesión.

2. El usuario podrá realizar las siguientes acciones.

2.1 El sistema permitirá al usuario seleccionar una carpeta y la canción a reproducir.

2.2 El sistema mostrará al usuario el nombre de la canción que se está reproduciendo.

2.3 El sistema permitirá que el usuario pause la canción que está reproduciendo, cuando él lo desee.

2.4 El sistema permitirá que el usuario pueda cambiar la canción por la anterior.

2.5 El sistema permitirá al usuario pasar a la siguiente canción.

2.6 El sistema permitirá al usuario, minimizar y cerrar el programa.

Requerimientos no funcionales.

1. El sistema debe ser fácil de usar e intuitivo para los usuarios.

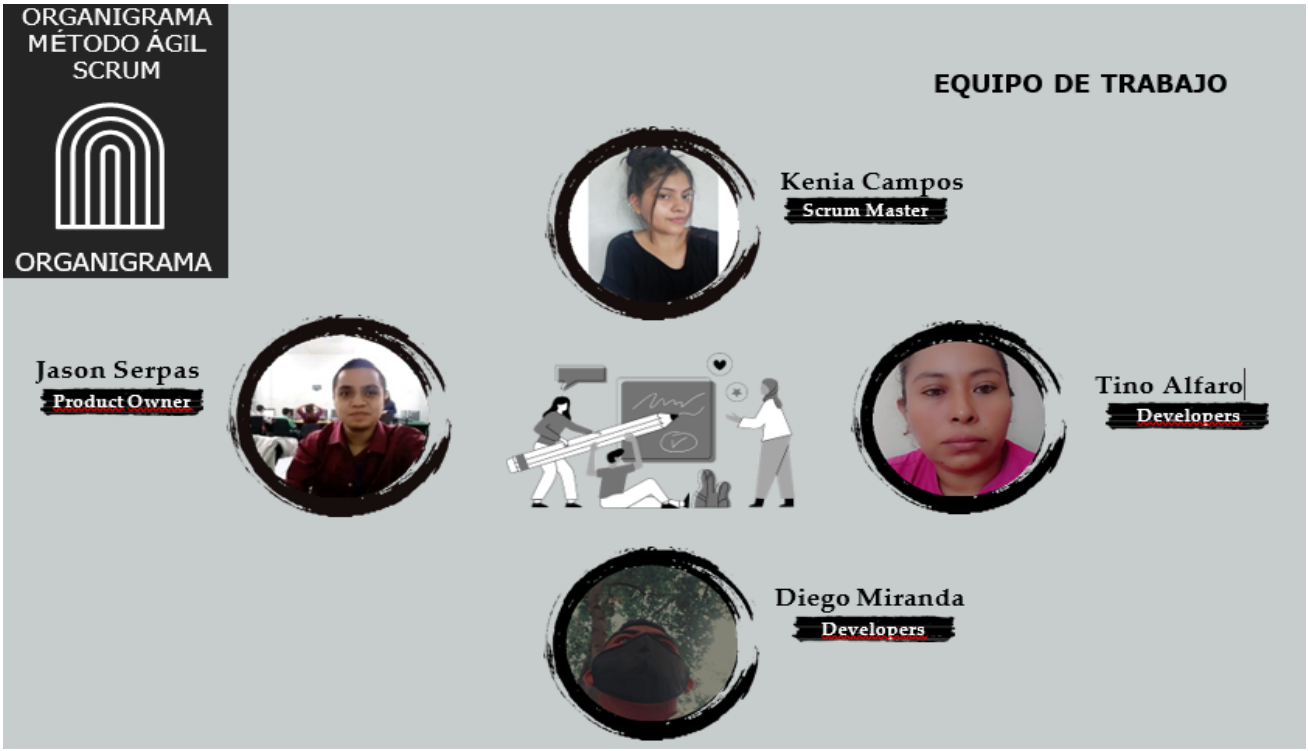
2. El reproductor debe ser compatible para diferentes sistemas operativos, como Windows, Linux y MacOs.

3. El sistema debe tener una interfaz gráfica, atractiva y personalizada.

4. El sistema debe brindar seguridad.

5. El sistema permitirá hacer otras tareas al mismo tiempo.

6. El programa se realizará en Python.



Nombre de la tarea	Responsable	Fecha inicio	Fecha final	Días	Estado
HU01 El usuario podrá visualizar la interfaz delreproductor.	-	**29/09/23**	**04/10/23**	**5**	Pendiente
1.1 El sistema permitirá que el usuario pueda visualizar el logo del reproductor al iniciar.	-	04/10/23	07/10/23	3	Pendiente
1.2 El sistema permitirá que el usuario pueda observar las diferentes funciones que tendrá el reproductor.	-	07/10/23	15/10/23	8	Pendiente

Nombre de la tarea	Responsable	Fecha inicio	Fecha final	Días	Estado
HU02 El usuario podrá realizar las siguientesacciones.	-	**15/10/23**	**15/10/23**	**1**	**Pendiente**
2.1 El sistema permitirá al usuario seleccionar unacarpeta y la canción a reproducir.	-	16/10/23	21/10/23	5	Pendiente
2.2 El sistema mostrará al usuario el nombre de lacanción que se está reproduciendo.	-	21/10/23	26/10/23	5	Pendiente
2.3 El sistema permitirá que el usuario pause la canción que está reproduciendo, cuando él lodesee.	-	26/10/23	31/10/23	5	Pendiente
2.4 El sistema permitirá que el usuario puedacambiar la canción por la anterior.	-	1/11/23	05/11/23	4	Pendiente
2.5 El sistema permitirá al usuario pasar a lasiguiente canción.	-	05/11/23	08/11/23	3	Pendiente
2.6 El sistema permitirá al usuario salir/cerrar elprograma.	-	08/11/23	12/11/23	4	Pendiente

Nombre de la tarea	Responsable	Fecha inicio	Fecha final	Días	Estado
HU03 El sistema debe ser fácil de usar e intuitivopara los usuarios.	-	-	-	-	Pendiente
HU04 El reproductor debe ser compatible para diferentes sistemas operativos, como Windows, Linuxy MacOs.	-	-	-	-	Pendiente
HU05 El sistema debe tener una interfaz gráfica,atractiva y personalizada.	-	-	-	-	Pendiente
HU06 El sistema debe brindar seguridad.	-	-	-	-	Pendiente
HU07 El sistema permitirá hacer otras tareas almismo tiempo.	-	-	-	-	Pendiente
HU08 El programa se realizará en Python.	-	-	-	-	Pendiente

Desarrollo ágil

Carga de Canciones: Los usuarios pueden cargar su colección de canciones en formato MP3 en la lista de reproducción del reproductor.

Reproducción y Pausa: El reproductor permite reproducir y pausar las canciones en curso, con la capacidad de reanudar la reproducción desde el punto en que se pausó.

Información de Canción: El reproductor muestra información sobre la canción en reproducción, incluyendo el título de la canción y el nombre del archivo.

Navegación de Canciones: Los usuarios pueden cambiar de canción utilizando los botones de "Anterior" y "Siguiente", así como una función de reproducción aleatoria.

Interfaz Gráfica Atractiva: La interfaz gráfica se ha diseñado para ser atractiva y de fácil uso, con botones claros y elementos de diseño elegantes.

Tecnologías a ocupar para el desarrollo del proyecto.

conexión estable

Se necesita una conexión estable para no tener alguna dificultad al momento de ejecutar el programa.

visual studio code

IDE de Visual Studio es una plataforma de lanzamiento creativa que puede utilizar para editar, depurar y compilar código y, finalmente, publicar una aplicación

lenguaje python

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, por ejemplo: Instagram, Netflix, Spotify, Panda3D, entre otros.

pysimplegui

PySimpleGUI es un contenedor de la API de Python del módulo Tkinter que permite al programador utilizar los mismos elementos de la interfaz de usuario que con Tkinter, pero con una interfaz más intuitiva.

Bibliografía.

Python:

https://www.python.org/

Visual Studio Code:

https://code.visualstudio.com/

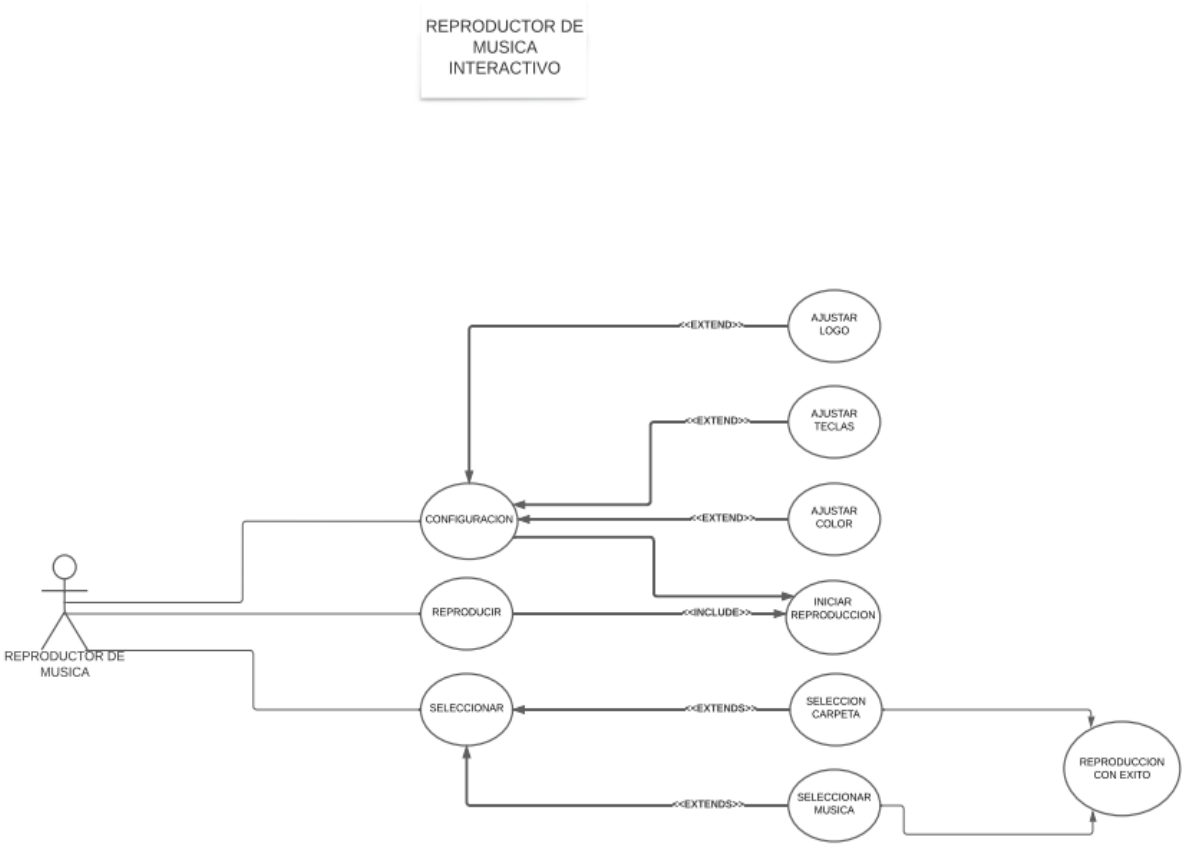
Pygame:

https://www.pygame.org/docs/

PySimpleGUI:

https://pypi.org/project/PySimpleGUI/

Diagrama de casos de uso.



Descripción de casos de uso.

Caso de uso	**1. Configuración**
Objetivo	Permitir al usuario configurar diferentes aspectos del reproductor de música según sus preferencias.
Actor principal	Usuario.
Personal involucrado	El usuario que desea realizar configuraciones en el reproductor de música.
Precondición	La aplicación de reproductor de música está en ejecución y la interfaz gráfica está visible.

Garantías de éxito	Las configuraciones se aplican correctamente y se reflejan en el funcionamiento del reproductor.
Escenario principal	– El usuario selecciona la opción de configuración en la interfaz del reproductor de música. – La aplicación muestra un menú de configuración que permite al usuario ajustar diversas opciones – El usuario realiza las configuraciones deseadas. – La aplicación guarda las configuraciones y las aplica al reproductor de música.
Flujos alternativos	– Si el usuario decide cancelar la configuración en cualquier momento, se le proporciona una opción para salir del menú de configuración sin aplicar cambios.
Requisitos especiales	– La interfaz de configuración debe ser intuitiva y fácil de usar. – Las configuraciones deben reflejar las preferencias del usuario en el funcionamiento del reproductor de música.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de ajustar la configuración del reproductor de música. Puede ocurrir cuando el usuario desee personalizar su experiencia de reproducción musical o cuando cambien sus preferencias**.**

Caso de uso	**2. Reproducir**
Objetivo	Permitir al usuario iniciar el reproductor de música.
Actor principal	Usuario.
Personal involucrado	El usuario que desea utilizar el reproductor de música.

Precondición	La aplicación de reproductor de música está instalada y lista para ejecutarse en el sistema del usuario.
Garantías de éxito	La aplicación se inicia correctamente y muestra la interfaz gráfica.
Escenario principal	– El usuario ejecuta la aplicación de reproductor de música. – La aplicación muestra la interfaz gráfica del reproductor de música.
Flujos alternativos	– En caso de que la aplicación no se inicie correctamente, se muestra un mensaje de error. – El usuario puede cerrar la aplicación en cualquier momento.
Requisitos especiales	– Se requiere que el sistema operativo del usuario sea compatible con la aplicación de reproductor de música. – Deben estar instaladas las dependencias necesarias para la ejecución de la aplicación, como bibliotecas de Python y recursos gráficos.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de utilizar el reproductor de música. Puede ocurrir cada vez que el usuario desee iniciar la aplicación.

Caso de uso	**3. Seleccionar.**
Objetivo	Permitir al usuario cambiar de canción dentro de la lista de reproducción.
Actor principal	Usuario.
Personal involucrado	El usuario que desea cambiar de canción.
Precondición	Una lista de reproducción está cargada y al menos una canción se está reproduciendo
Garantías de éxito	La canción siguiente o anterior se reproduce correctamente.

Escenario principal	– El usuario selecciona la opción de cambiar a la canción siguiente o anterior en la interfaz. – La aplicación cambia a la siguiente o anterior canción de la lista y la reproduce.
Flujos alternativos	– Si la acción no se realiza correctamente, se muestra un mensaje de error
Requisitos especiales	– Debe haber al menos dos canciones en la lista de reproducción para que el usuario pueda cambiar entre ellas – La interfaz debe proporcionar botones claros y visibles para que el usuario pueda realizar estas acciones de manera intuitiva.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de cambiar de canción dentro de la lista de reproducción. Puede ocurrir varias veces durante la reproducción de una lista de reproducción.

Caso de uso	**4. Ajustar Logo**
Objetivo	Permitir al usuario personalizar el logo del reproductor de música ajustando su apariencia mediante una foto PNG desde el código en Visual Studio Code.
Actor principal	Usuario.
Personal involucrado	El usuario que desea personalizar el logo del reproductor de música con una foto PNG.
Precondición	El reproductor de música está en funcionamiento y la interfaz gráfica es visible.
Garantías de éxito	La foto PNG personalizada se aplica correctamente como logo en la interfaz del reproductor de música
Escenario principal	– El usuario abre el código fuente del reproductor de música en Visual Studio Code.

	– Busca la sección del código que está relacionada con la configuración y visualización del logo en la interfaz. – El usuario debe cargar la foto PNG que desea utilizar como logo en el reproductor. Esto puede implicar agregar un nuevo archivo PNG al proyecto. – Dentro del código, se configuran las propiedades necesarias para cargar y mostrar la foto PNG como logo. Esto incluye la ubicación del archivo PNG, el tamaño, y otros atributos visuales. – Una vez que el usuario ha realizado las configuraciones, guarda el archivo de código. – El usuario ejecuta el reproductor de música, y la foto PNG personalizada se refleja en la interfaz del reproductor como logo.
Flujos alternativos	– Si el usuario decide cancelar las configuraciones en cualquier momento, puede cerrar Visual Studio Code sin guardar los cambios en el archivo de código.
Requisitos especiales	– Los usuarios deben tener conocimientos de programación en Python y comprensión de la estructura del código del reproductor para realizar estas configuraciones. – La foto PNG debe ser compatible con el reproductor de música y estar ubicada en una ubicación accesible para el código.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de personalizar el logo del reproductor de música con

	una foto PNG. Puede ocurrir cuando el usuario desee ajustar la apariencia visual del logo utilizando una imagen personalizada.

Caso de uso	**5. Ajustar Teclas**
Objetivo	Permitir al usuario personalizar las teclas de pausa, retroceso y avance rápido en el reproductor de música directamente desde el código utilizando Visual Studio Code.
Actor principal	Usuario.
Personal involucrado	El usuario que desea personalizar la apariencia o funcionalidad de los botones de pausa, retroceso y avance rápido en el reproductor de música.
Precondición	El reproductor de música está iniciado y la interfaz gráfica es visible
Garantías de éxito	Las personalizaciones realizadas por el usuario se aplican correctamente a los botones de pausa, retroceso y avance rápido.
Escenario principal	– El usuario abre el código fuente del reproductor de música en Visual Studio Code. – Busca la sección del código que está relacionada con la apariencia y funcionalidad de los botones de pausa, retroceso y avance rápido. – El usuario puede realizar configuraciones en el código para definir aspectos como el diseño, el comportamiento de estos botones. – Una vez que el usuario ha realizado las configuraciones deseadas, guarda el archivo de código.

	– El usuario ejecuta el reproductor de música, y las personalizaciones se reflejan en la interfaz gráfica con los botones de pausa, retroceso y avance rápido personalizados.
Flujos alternativos	– Si el usuario decide cancelar las configuraciones en cualquier momento, puede cerrar Visual Studio Code sin guardar los cambios en el archivo de código.
Requisitos especiales	– Los usuarios deben tener conocimientos de programación en Python y comprensión de la estructura del código del reproductor para realizar estas configuraciones. – El código del reproductor de música debe estar correctamente organizado y comentado para que el usuario pueda identificar la sección relevante para personalizar los botones.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de personalizar los botones de pausa, retroceso y avance rápido en el reproductor de música. Puede ocurrir cuando el usuario desea adaptar la apariencia y el comportamiento de los botones a sus preferencias personales.

Caso de uso	**6. Ajustar Color**
Objetivo	Permitir al usuario personalizar los colores de la interfaz gráfica y establecer el color de

	fondo de la ventana del reproductor de música directamente desde el código utilizando Visual Studio Code.
Actor principal	Usuario.
Personal involucrado	El usuario que desea personalizar la apariencia de la interfaz gráfica y el fondo de la ventana del reproductor de música.
Precondición	El reproductor de música está iniciado y la interfaz gráfica es visible.
Garantías de éxito	Las personalizaciones realizadas por el usuario se aplican correctamente a la interfaz gráfica y al fondo de la ventana.
Escenario principal	– El usuario abre el código fuente del reproductor de música en Visual Studio Code. – Busca la sección del código que está relacionada con la configuración de colores de la interfaz gráfica y el fondo de la ventana. – El usuario puede realizar configuraciones en el código para definir los colores de los elementos de la interfaz, como botones, fondos. – Además, el usuario puede establecer el color de fondo de la ventana del reproductor de música. – Una vez que el usuario ha realizado las configuraciones deseadas, guarda el archivo de código. – El usuario ejecuta el reproductor de música, y las personalizaciones se reflejan en la interfaz gráfica con los colores y el fondo de ventana personalizados.
Flujos alternativos	– Si el usuario decide cancelar las configuraciones en

	cualquier momento, puede cerrar Visual Studio Code sin guardar los cambios en el archivo de código.
Requisitos especiales	– Los usuarios deben tener conocimientos de programación en Python y comprensión de la estructura del código del reproductor para realizar estas configuraciones. – El código del reproductor de música debe estar correctamente organizado y comentado para que el usuario pueda identificar la sección relevante para personalizar los colores y el fondo.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de personalizar los colores de la interfaz gráfica y el fondo de la ventana del reproductor de música. Puede ocurrir cuando el usuario desea adaptar la apariencia visual del reproductor a sus preferencias personales.

Caso de uso	**7. Iniciar Reproducción**
Objetivo	Permitir al usuario reproducir una canción de la lista de reproducción.
Actor principal	Usuario.
Personal involucrado	El usuario que desea reproducir una canción.
Precondición	La lista de reproducción está cargada y visible en la interfaz gráfica.
Garantías de éxito	La canción seleccionada se reproduce correctamente.

Escenario principal	– El usuario selecciona una canción de la lista de reproducción en la interfaz. – La aplicación comienza la reproducción de la canción seleccionada.
Flujos alternativos	– Si la canción no se reproduce correctamente, se muestra un mensaje de error.
Requisitos especiales	– Se requiere que el sistema tenga acceso a los archivos de las canciones y los recursos de audio necesarios para la reproducción. – Deben estar instalados las bibliotecas necesarias para reproducir archivos de audio en formato MP3.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de reproducir una canción. Puede ocurrir cada vez que el usuario elija reproducir una canción de la lista de reproducción

Caso de uso	**8. Seleccionar Carpeta**
Objetivo	Permitir al usuario cargar una lista de reproducción de canciones seleccionando una carpeta que contenga archivos de música en formato MP3.
Actor principal	Usuario.
Personal involucrado	El usuario que desea cargar una lista de reproducción
Precondición	El reproductor de música está iniciado y la interfaz gráfica es visible.
Garantías de éxito	La lista de reproducción se carga correctamente y se muestra en la interfaz.
Escenario principal	– El usuario selecciona la opción "Seleccionar Carpeta" en la interfaz del reproductor de música. – La aplicación abre un cuadro de diálogo que permite al usuario explorar y seleccionar una carpeta en

	su sistema que contenga archivos de música en formato MP3. – Una vez seleccionada la carpeta, la aplicación escanea su contenido en busca de archivos de música y crea una lista de reproducción con las canciones encontradas. – La lista de canciones disponibles se muestra en la interfaz gráfica, ya sea en forma de lista. – El usuario puede seleccionar una canción de la lista para iniciar la reproducción.
Flujos alternativos	– Si no se encuentran archivos de música en la carpeta seleccionada, se no se reproducirá ninguna canción, y se le da al usuario la opción de seleccionar una carpeta diferente.
Requisitos especiales	– Se requiere que el sistema operativo del usuario tenga acceso a las carpetas y archivos necesarios para cargar las canciones. – La carpeta seleccionada por el usuario debe contener archivos de canciones en formato MP3 para que puedan ser cargados.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de cargar una lista de reproducción. Puede ocurrir cada vez que el usuario desee agregar nuevas canciones a la lista de reproducción.

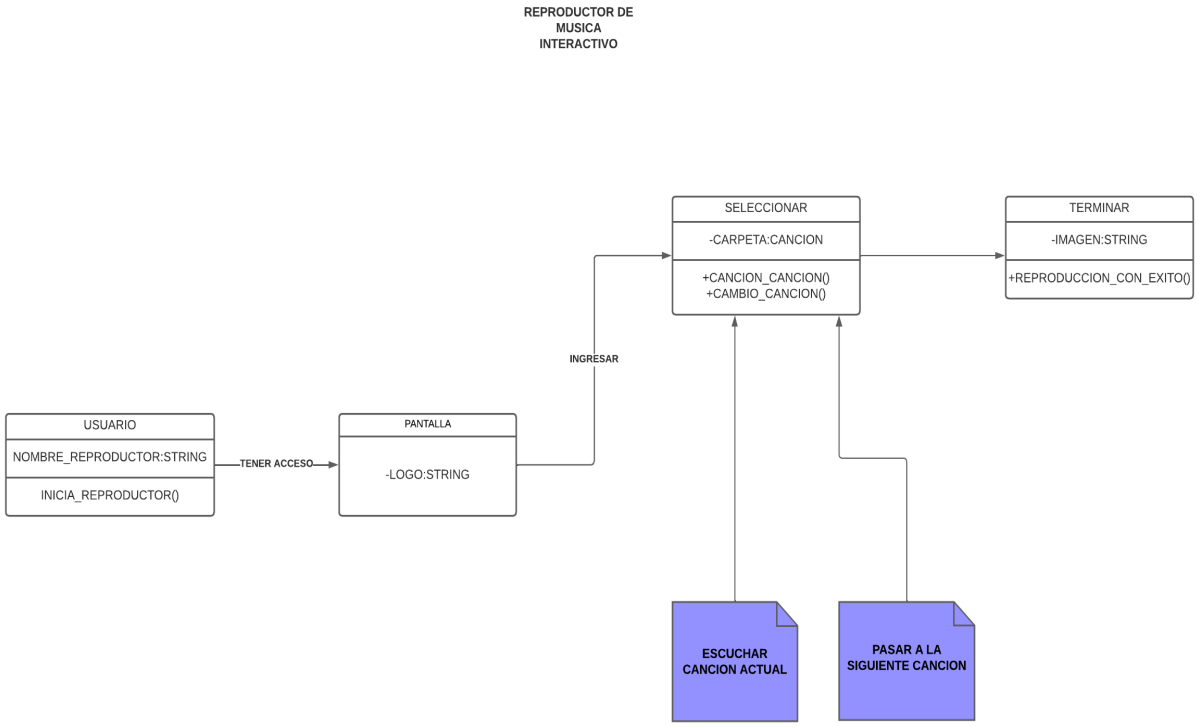
Caso de uso	**9. Seleccionar Música**
Objetivo	Permitir al usuario seleccionar y cargar música en el reproductor de música.
Actor principal	Usuario
Personal involucrado	El usuario que desea cargar música en el reproductor.
Precondición	El reproductor de música está iniciado y la interfaz gráfica es visible.
Garantías de éxito	La música seleccionada por el usuario se carga correctamente en el reproductor y está lista para su reproducción.
Escenario principal	– El usuario selecciona la opción "Seleccionar Canción" en la interfaz del reproductor de música. – La aplicación muestra una ventana de exploración de archivos que permite al usuario navegar por su sistema de archivos y seleccionar las canciones que desea cargar en el reproductor. – El usuario navega a la ubicación de las canciones en su sistema y selecciona la canción que desee cargar. – Una vez seleccionada la canción, el usuario confirma su selección. – La aplicación carga la canción seleccionada en la lista de reproducción del reproductor de música. – La canción cargada se muestra en la interfaz gráfica del reproductor, ya sea como una lista de reproducción.
Flujos alternativos	– Si el usuario no selecciona ninguna canción o decide cancelar la operación en cualquier momento, la aplicación no carga ninguna canción y permanece en el estado anterior.
Requisitos especiales	– La aplicación debe proporcionar una interfaz de selección de archivos intuitiva y fácil de usar. – Deben admitirse formatos de archivo de música común, en MP3.

	– La aplicación debe permitir al usuario navegar por su sistema de archivos y seleccionar la canción desde diferentes ubicaciones.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de cargar música en el reproductor. Puede ocurrir cada vez que el usuario desee actualizar o cambiar la lista de reproducción.

Caso de uso	**10. Reproducción con Éxito**
Objetivo	Describir el escenario en el que un usuario reproduce una canción con éxito en el reproductor de música.
Actor principal	Usuario.
Personal involucrado	El usuario que desea reproducir una canción.
Precondición	La lista de reproducción está cargada en el reproductor y la interfaz gráfica es visible.
Garantías de éxito	La canción seleccionada se reproduce correctamente.
Escenario principal	– El usuario visualiza la lista de canciones en la interfaz gráfica del reproductor de música, que puede presentarse en forma de lista o carátulas de álbumes. – El usuario selecciona una canción de la lista de reproducción haciendo clic en ella o utilizando un botón de reproducción junto a la canción. – La aplicación inicia la reproducción de la canción seleccionada. – La interfaz gráfica muestra información sobre la canción que se está reproduciendo, como el título, el artista y la carátula del álbum. – El usuario puede controlar la reproducción, pausa, reanudación, utilizando los controles disponibles en la interfaz.

	– La canción se reproduce con éxito, y el usuario puede disfrutar de la música.
Flujos alternativos	– Si la canción no se reproduce correctamente debido a problemas técnicos, se muestra un mensaje de error en la interfaz y se ofrece al usuario la opción de intentar nuevamente.
Requisitos especiales	– La aplicación debe tener acceso a los archivos de las canciones y a los recursos de audio necesarios para la reproducción. – Deben estar instaladas las bibliotecas necesarias para reproducir archivos de audio en el formato admitido.
Frecuencia	– La frecuencia de este caso de uso depende de la decisión del usuario de reproducir una canción de la lista de reproducción. Puede ocurrir cada vez que el usuario elija reproducir una canción.

Diagrama de clases.



Listado de requerimientos que se ajustan para garantizar la confiabilidad y seguridad del software a desarrollar.

Requerimientos de Seguridad:

1- El sistema debe requerir que los usuarios se identifiquen antes de acceder a funciones sensibles.
2- El sistema debe autenticar a los usuarios de manera segura mediante contraseñas o métodos de autenticación adecuados.
3- El sistema debe definir roles de usuario y otorgar permisos de acceso basados en estos roles.
3.1 Solo los usuarios autorizados pueden realizar acciones como agregar, eliminar o modificar canciones.

4- El sistema debe contar con mecanismos de detección de intrusiones que alerten sobre posibles ataques o comportamientos anómalos.
5- El sistema debe cumplir con las regulaciones de protección de datos personales y garantizar la privacidad de la información del usuario.
6- El sistema debe llevar un registro de actividades de los usuarios, incluyendo acciones realizadas y eventos de seguridad.
6.1 Los registros de actividad deben ser accesibles solo por administradores autorizados.
7- El sistema debe ser capaz de recibir y aplicar actualizaciones de seguridad periódicas.
8- El sistema debe contar con medidas de protección contra malware que puedan afectar la integridad de los archivos de música
9- El sistema debe definir un plan de respuesta a incidentes que incluya la notificación y mitigación de posibles vulnerabilidades o brechas de seguridad.

Requerimientos de Confiabilidad:

1-El sistema debe estar disponible para los usuarios en un alto porcentaje del tiempo, con un objetivo de disponibilidad del 99% o superior

2- El sistema debe realizar copias de seguridad periódicas de la información de los usuarios.

2.1 Debe ser posible recuperar los datos en caso de fallos o pérdida de información.

3- El sistema debe ser capaz de tolerar fallas en componentes individuales sin que esto cause la interrupción completa del servicio.

4- El sistema debe permitir la aplicación de actualizaciones sin interrumpir la disponibilidad del servicio.

5- El sistema debe ser capaz de manejar un número razonable de usuarios concurrentes sin degradación significativa del rendimiento.

6- El sistema debe someterse a pruebas rigurosas de seguridad y confiabilidad antes de su implementación.

Identificar las vulnerabilidades asociadas con las opciones de tecnología.

Vulnerabilidades.

1. Errores de programación**:** Bugs o errores en el código pueden ser explotados para comprometer la seguridad del software.

2. Gestión insegura de contraseñas: Almacenar contraseñas de manera insegura o no proteger adecuadamente los datos de autenticación puede ser una vulnerabilidad.

3. Desbordamiento de búfer: Un error común que puede ser explotado para ejecutar código malicioso mediante la manipulación de datos que superan los límites de almacenamiento asignados.

4. Problemas de configuración: Configuraciones inseguras por defecto o una administración deficiente de la configuración pueden dejar el software expuesto a riesgos.

5. Falta de actualizaciones y parches: No mantener el software actualizado con las últimas correcciones de seguridad puede dejarlo vulnerable a exploits conocidos.

6. Problemas de seguridad en bibliotecas y dependencias: Si el reproductor de música utiliza bibliotecas o dependencias de terceros, las vulnerabilidades en esas bibliotecas pueden afectar la seguridad del proyecto.

7. Falta de control de acceso: No implementar adecuadamente el control de acceso puede permitir que usuarios no autorizados accedan a funciones o datos sensibles.

Bibliografía:

Python: https://www.python.org/

Visual Studio Code: https://code.visualstudio.com/

Pygame: https://www.pygame.org/docs/

PySimpleGUI: https://pypi.org/project/PySimpleGUI/

Establecer y justificar el tipo de software avanzado que aplica en su proyecto (características, ventajas, desventajas)

justificación

- Amplia Comunidad: Python cuenta con una amplia comunidad de desarrolladores y abundante documentación, facilitando el desarrollo y la resolución de problemas.

- Librerías de interfaz gráfica:

Tkinter**:** Es un paquete de interfaz gráfica de usuario (GUI) estándar que viene con Python. Proporciona herramientas para la creación de ventanas, botones, cuadros de texto y otros elementos de interfaz de usuario.

Pygame: Es una biblioteca de Python diseñada para facilitar el desarrollo de videojuegos y aplicaciones multimedia. Aunque se centra principalmente en el desarrollo de juegos, también se puede utilizar para crear simulaciones interactivas y otras aplicaciones que involucren gráficos y sonido.

PySimpleGUI: Es una biblioteca que facilita la creación de interfaces gráficas de usuario de manera sencilla y rápida. A diferencia de Tkinter, PySimpleGUI busca simplificar la creación de interfaces de usuario, proporcionando una sintaxis más intuitiva y menos código.

- Versatilidad: Python es un lenguaje versátil que permite el desarrollo de aplicaciones tanto basadas en GUI como en CLI.

- Rápido Desarrollo: Python es conocido por su sintaxis clara y su capacidad para desarrollar rápidamente, lo que puede ser beneficioso en un proyecto de reproductor de música.

Características

1. Interfaz gráfica de usuario (GUI): Una interfaz visualmente atractiva que permite a los usuarios interactuar con el reproductor de música de manera intuitiva.

2. Interfaz basada en texto: Interacción mediante comandos en la terminal.

3. Funciones básicas: Reproducción, pausa, siguiente, etc.

4. Eficiencia: Puede ser más ligero y rápido que una aplicación con GUI.

Ventajas

1. Experiencia de Usuario Rica**:** Una GUI bien diseñada puede ofrecer una experiencia de usuario más atractiva y personalizable.

2. Funciones Avanzadas: Puede ofrecer un conjunto más amplio de funciones avanzadas comparado con enfoques basados en línea de comandos.

3. Accesibilidad desde cualquier lugar y dispositivo.

4. No se necesita internet para ejecutarlo.

Desventajas

1. Limitaciones en funciones avanzadas comparadas con aplicaciones de escritorio.

2. Limitaciones en la visualización debido al tamaño de la pantalla.

Establecer y describir los riesgos del proyecto a desarrollar (enfoque a proyecto no a software). A la vez, indicar probabilidad y efectos de cada riesgo.

TIPO DE RIESGO	RIESGOS POSIBLES
Tecnológico	El software no está bien estructurado. (1) No funcione el sistema. (2)
Personal	Un integrante se salga del equipo. (3) Porque se enferma un integrante. (4) Porque tengan discusiones. (5)
De organización	Cambio de algún miembro del equipo del personal. (6) Que el equipo no posea el conocimiento de código y su estructura. (7)
Herramientas	Tener las librerías actualizadas e instaladas correctamente. (8) Fallas con el lector del código, conexión de internet. (9)
Requerimientos	Que el sistema no funcione en MacOS. (10) El sistema no permita hacer otra función mientras se use el programa. (11)
Estimación	4 meses para realizar el software. (12) Si el software no funciona se puedan hacer correcciones. (13) No sobre pasar el tamaño del software. (14)

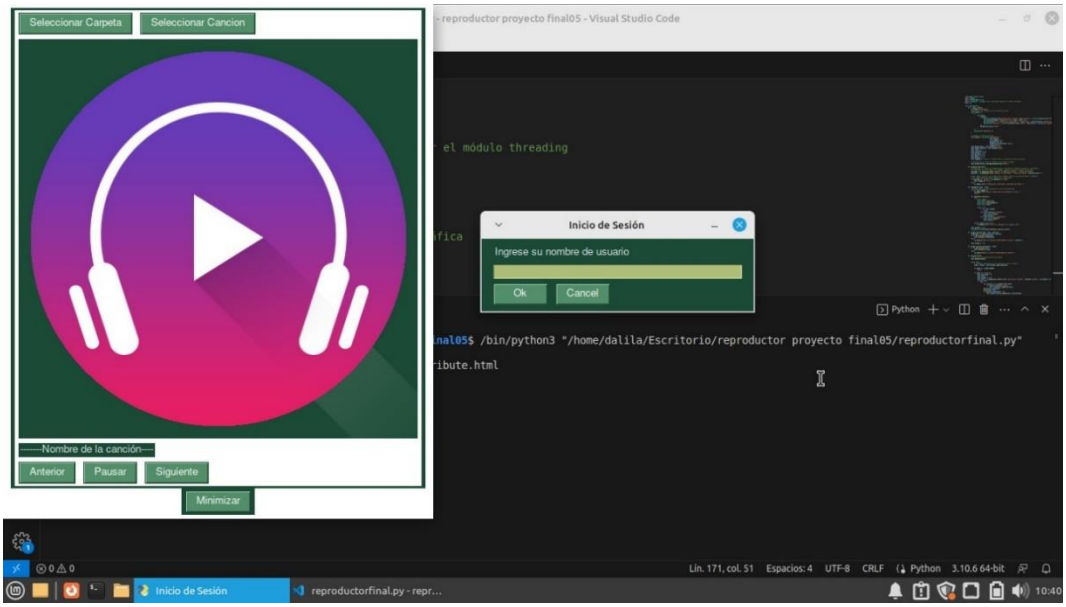
RIESGO	POSIBILIDAD	EFECTOS
El software no está bien estructurado. (1)	MODERADA	GRAVE
No funcione el sistema. (2)	BAJA	GRAVE
Un integrante se salga del equipo. (3)	BAJA	TOLERABLE
Porque se enferma un integrante. (4)	BAJA	TOLERABLE
Porque tengan discusiones. (5)	MODERADA	GRAVE
Cambio de algún miembro del equipo del personal. (6)	MODERADA	GRAVE
Que el equipo no posea el conocimiento de código y su estructura. (7)	MODERADA	GRAVE
Tener las librerías actualizadas e instaladas correctamente. (8)	ALTA	CATASTROFICO
Fallas con el lector del código, conexión de internet. (9)	MODERADA	GRAVE
Que el sistema no funcione en MacOS. (10)	MODERADA	CATASTROFICO
El sistema no permita hacer otra función mientras se use el programa. (11)	BAJA	INSIGNIFICANTE
4 meses para realizar el software. (12)	BAJA	TOLERABLE
Si el software no funciona se puedan hacer correcciones. (13)	BAJA	TOLERABLE
No sobre pasar el tamaño del software. (14)	BAJA	INSIGNIFICANTE

Establecer y describir estrategias para gestionar los riesgos del proyecto.

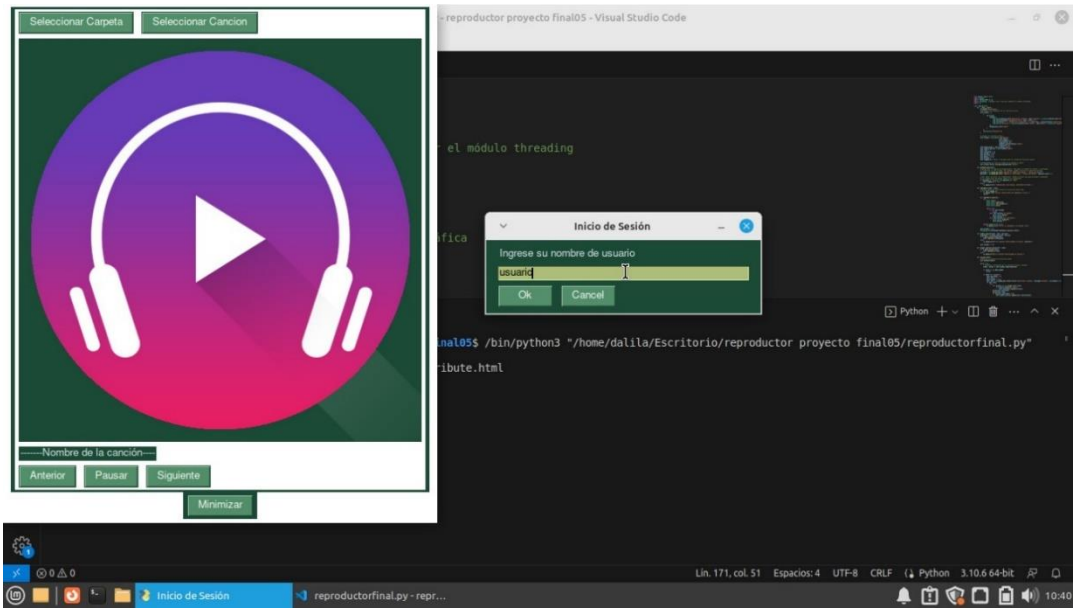
RIESGO	ESTRATEGIA
El software no está bien estructurado. (1)	- Realizar revisiones de código periódicas. - Implementar buenas prácticas de desarrollo de software. - Utilizar patrones de diseño y arquitecturas sólidas.
No funcione el sistema. (2)	- Desarrollar pruebas exhaustivas y automatizadas. - Realizar pruebas de integración frecuentes.
Un integrante se salga del equipo. (3)	- Mantener una buena comunicación y ambiente de trabajo. - Tener un plan de contingencia para redistribuir tareas.
Porque se enferma un integrante. (4)	- Mantener un registro actualizado de responsabilidades y avances. - Contar con un plan de contingencia para redistribuir tareas.
Porque tengan discusiones. (5)	- Fomentar una comunicación abierta y respetuosa. - Implementar prácticas de resolución de conflictos. - Designar un líder de equipo para mediar.
Cambio de algún miembro del equipo del personal. (6)	- Documentar procesos y roles claves. - Mantener una reserva de conocimiento compartido.
Que el equipo no posea el conocimiento de código y su estructura. (7)	- Capacitar al equipo en las tecnologías utilizadas. - Fomentar el aprendizaje continuo y la colaboración. - Utilizar documentación detallada.
Tener las librerías actualizadas e instaladas correctamente. (8)	- Automatizar la instalación de librerías mediante herramientas como gestores de paquetes.
Fallas con el lector del código, conexión de internet. (9)	- Realizar pruebas de rendimiento y de conectividad.
Que el sistema no funcione en MacOS. (10)	- Desarrollar y probar en diferentes plataformas.
El sistema no permita hacer otra función mientras se use el programa. (11)	- Implementar multihilo o procesos en segundo plano. - Realizar pruebas de usabilidad.
4 meses para realizar el software. (12)	- Utilizar metodologías ágiles para iteraciones rápidas. - Establecer hitos y seguimientos regulares.
Si el software no funciona se puedan hacer correcciones. (13)	- Implementar un proceso de retroalimentación y corrección continua.
No sobre pasar el tamaño del software. (14)	- Realizar análisis de impacto antes de agregar nuevas funciones.

Manual de Usuario.

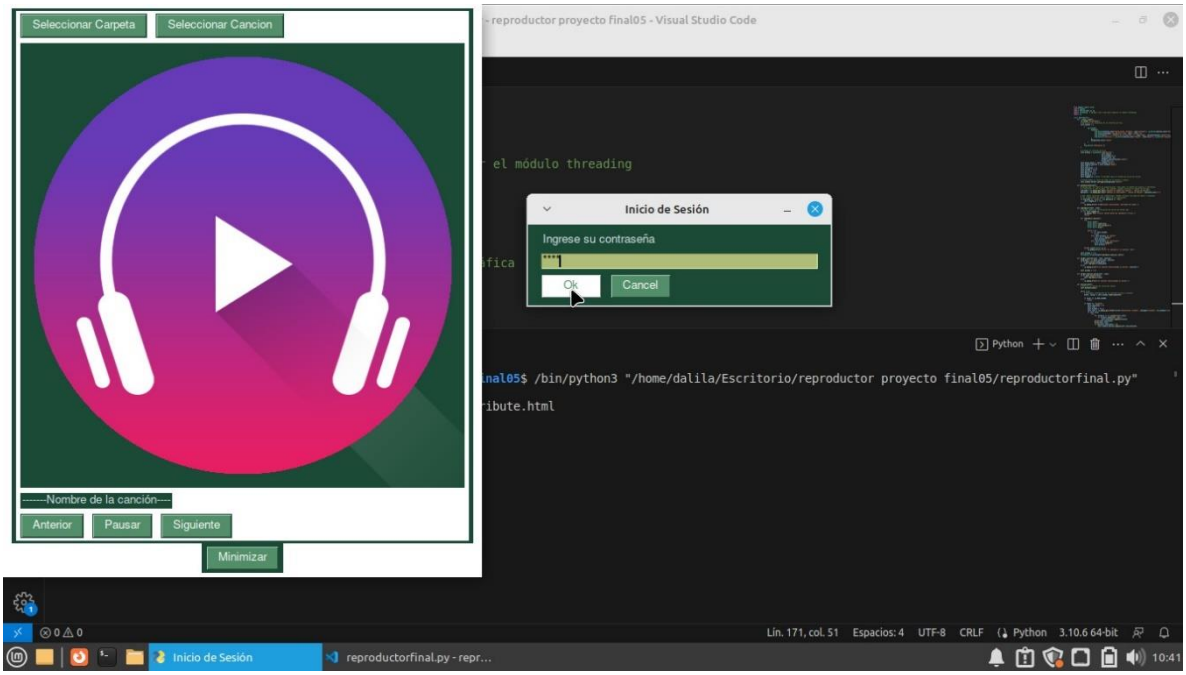
Al ejecutar el programa, nos aparece una ventana para iniciar con nuestro usuario.



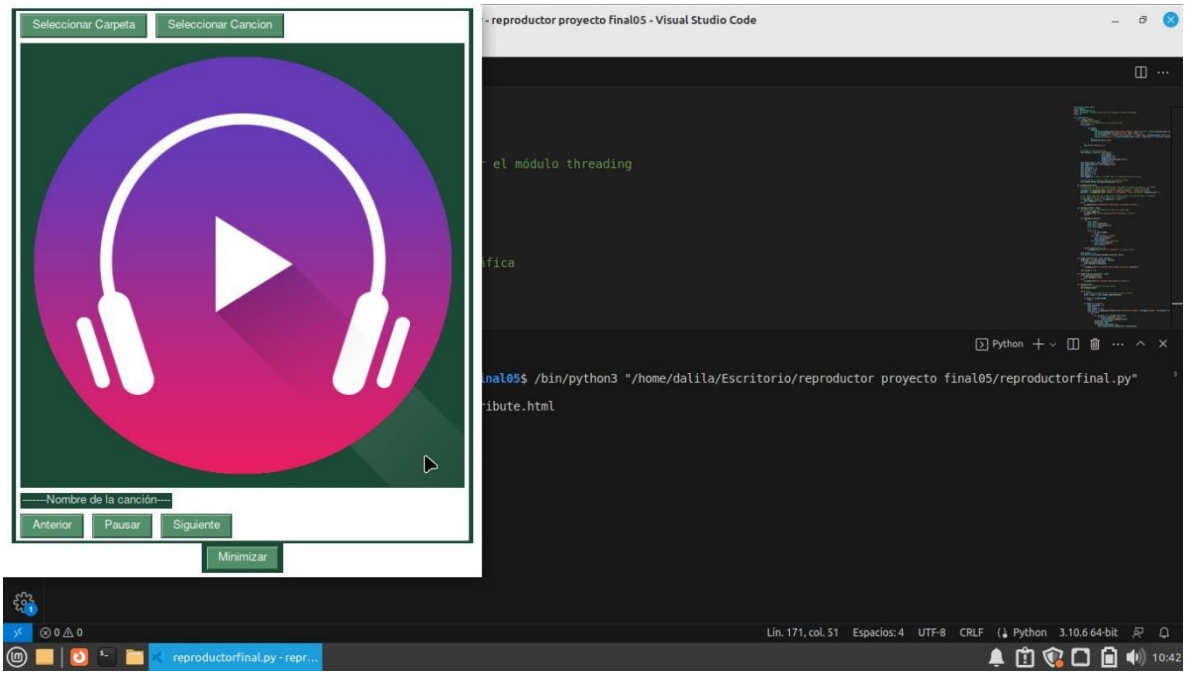
Seguidamente colocamos el usuario (usuario).



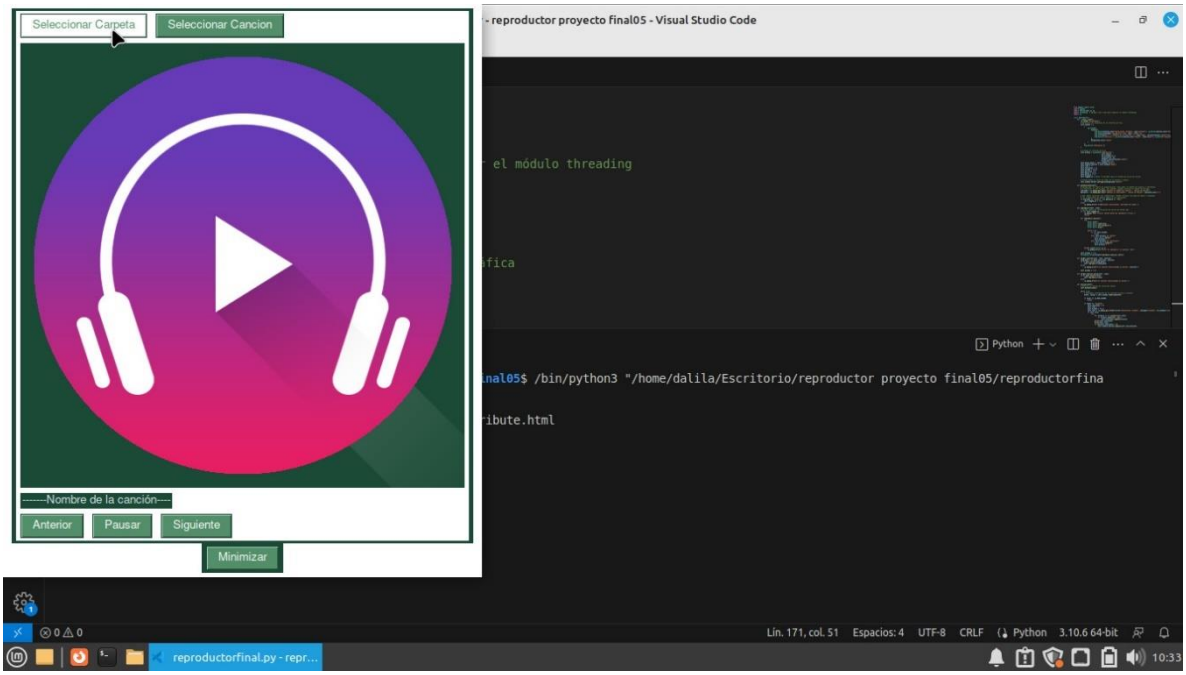
Luego colocamos la contraseña (1234) para poder iniciar el software.



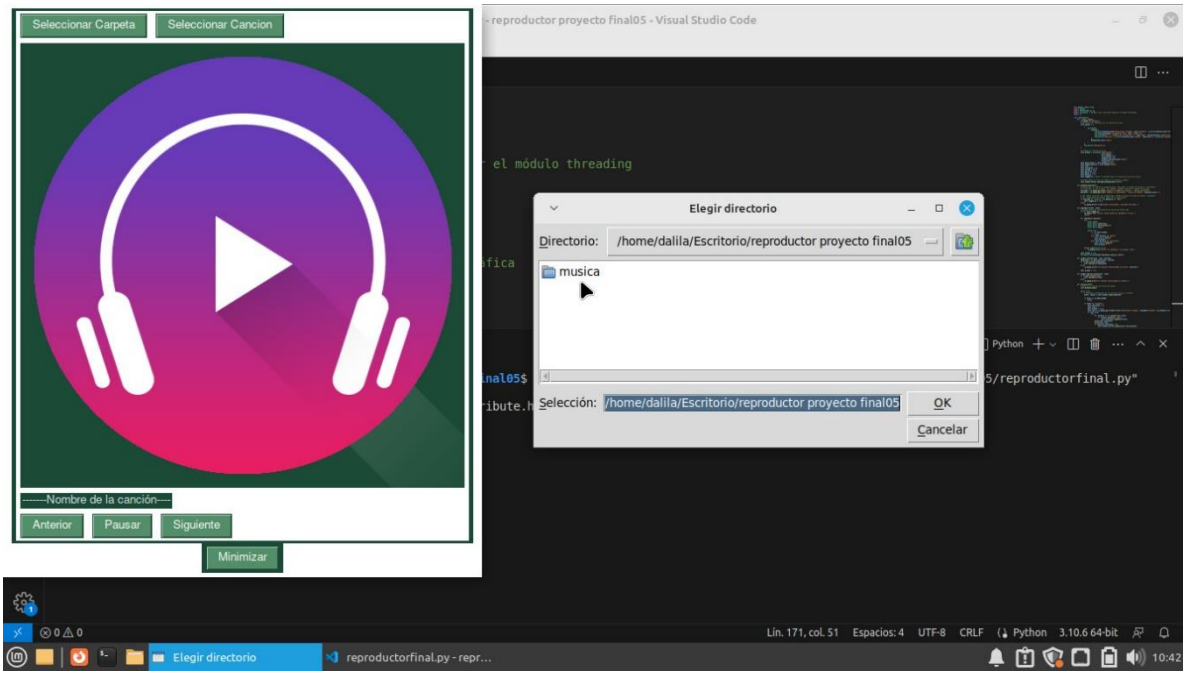
Ahora ya podemos acceder a nuestro reproductor.



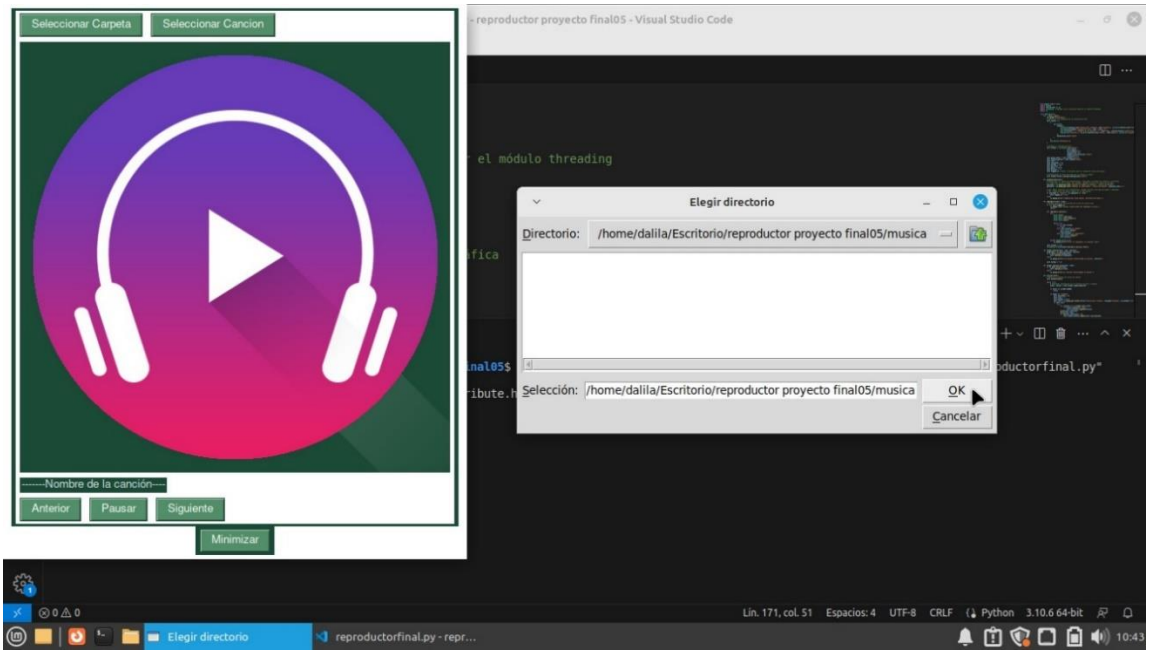
Al presionar el botón de “Seleccionar Carpeta” …



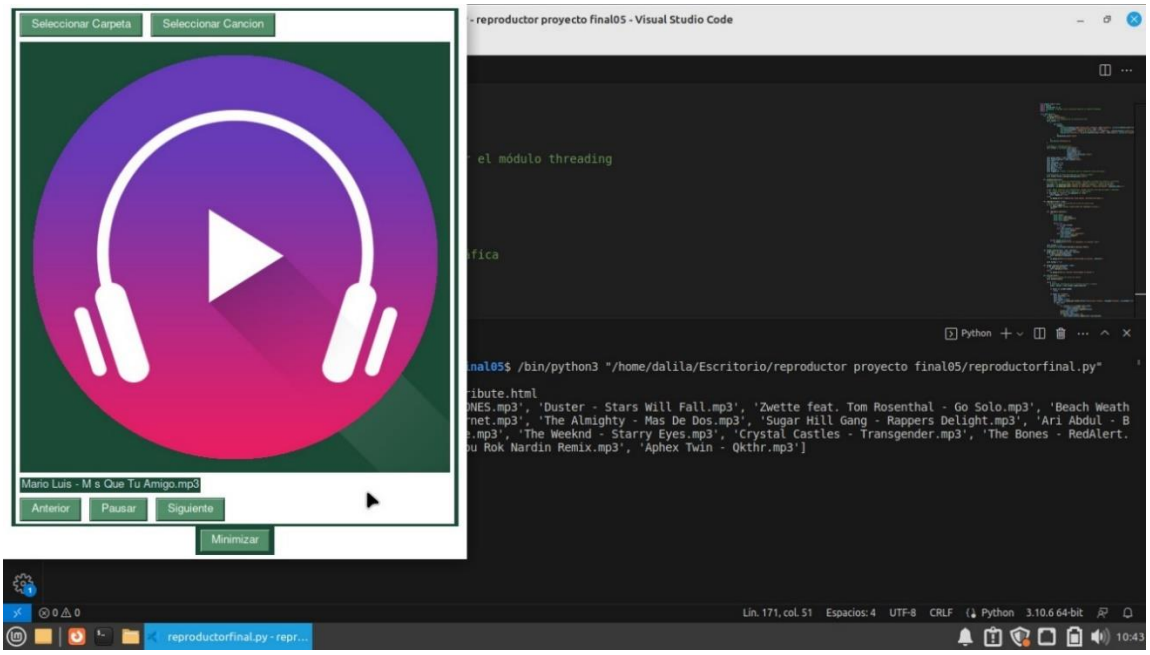
Se nos abre una ventana para seleccionar la carpeta con nuestra música en formato mp3.



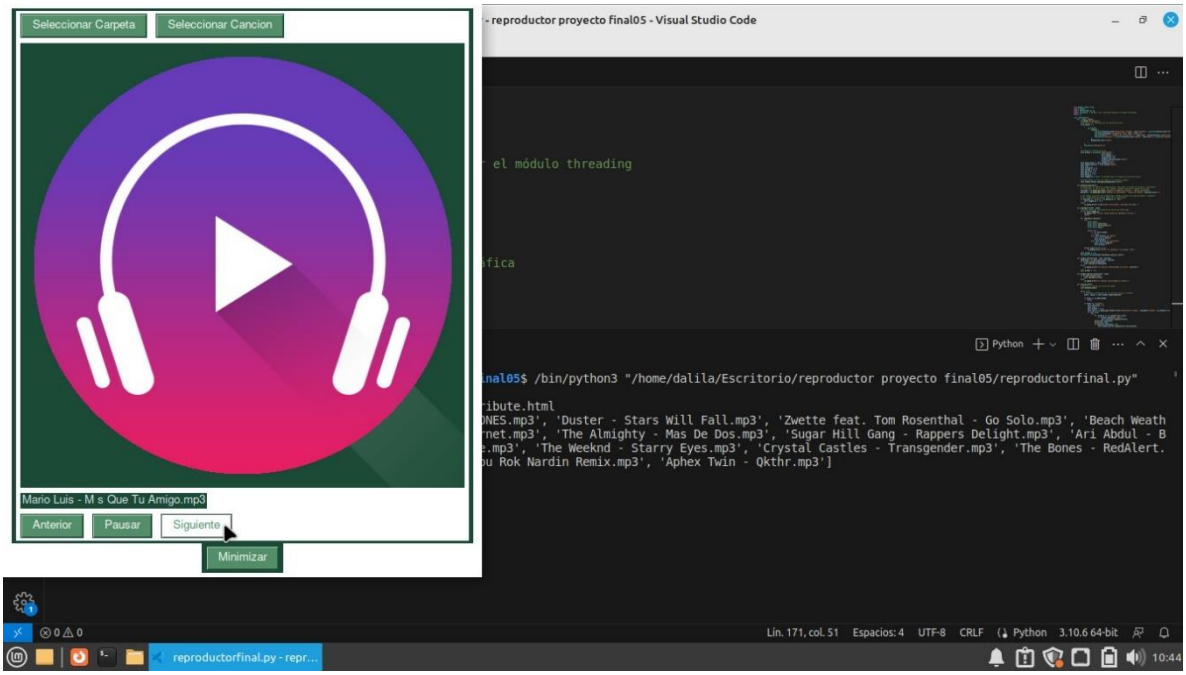
Damos clic en la carpeta y ya estando dentro, presionamos “OK”.



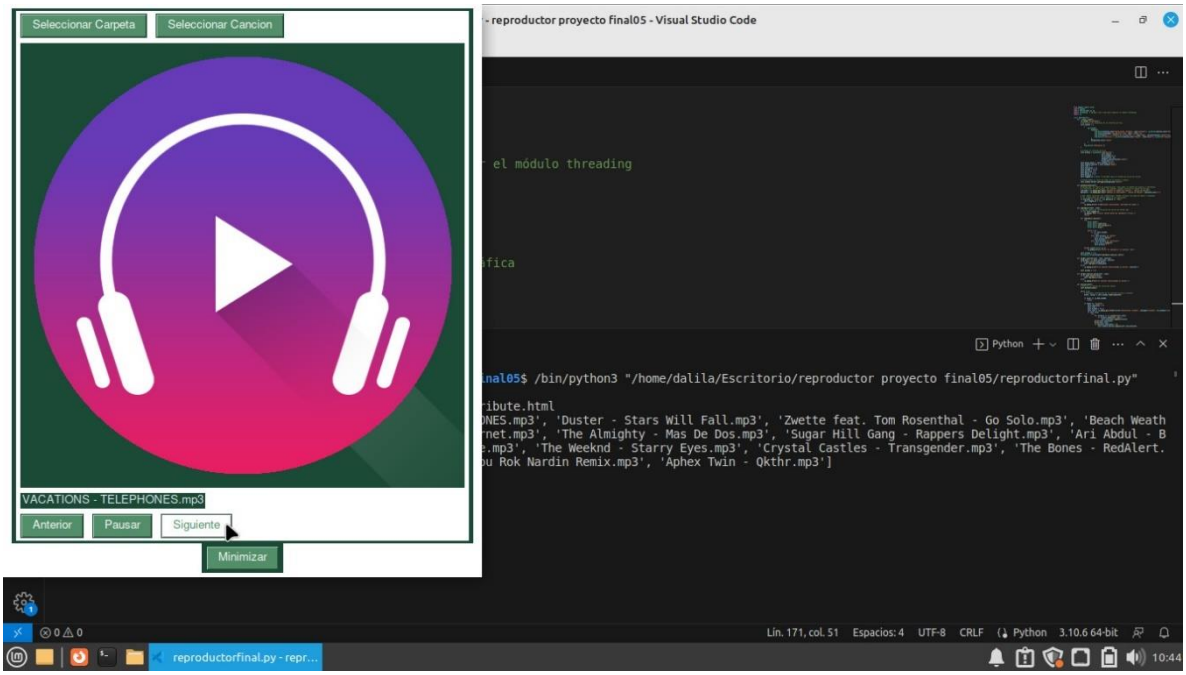
Automáticamente empieza a sonar la lista de reproducción que se encuentra en la carpeta antes seleccionada.



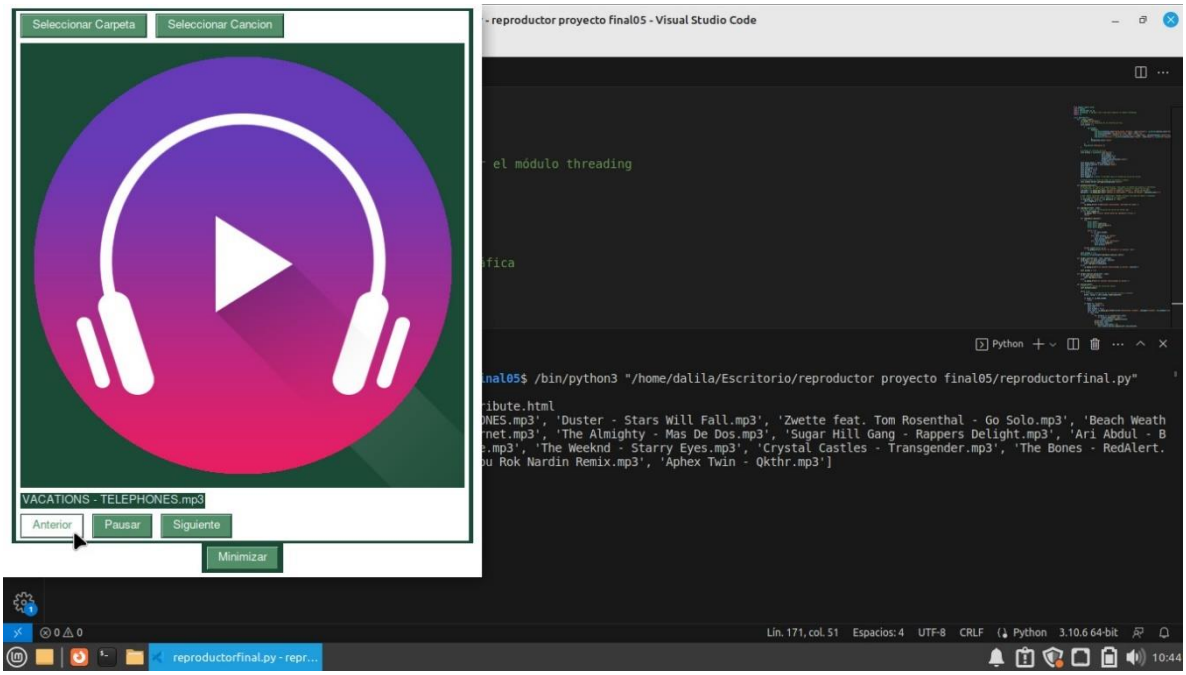
Ahora si presionamos el botón o tecla “Siguiente” …



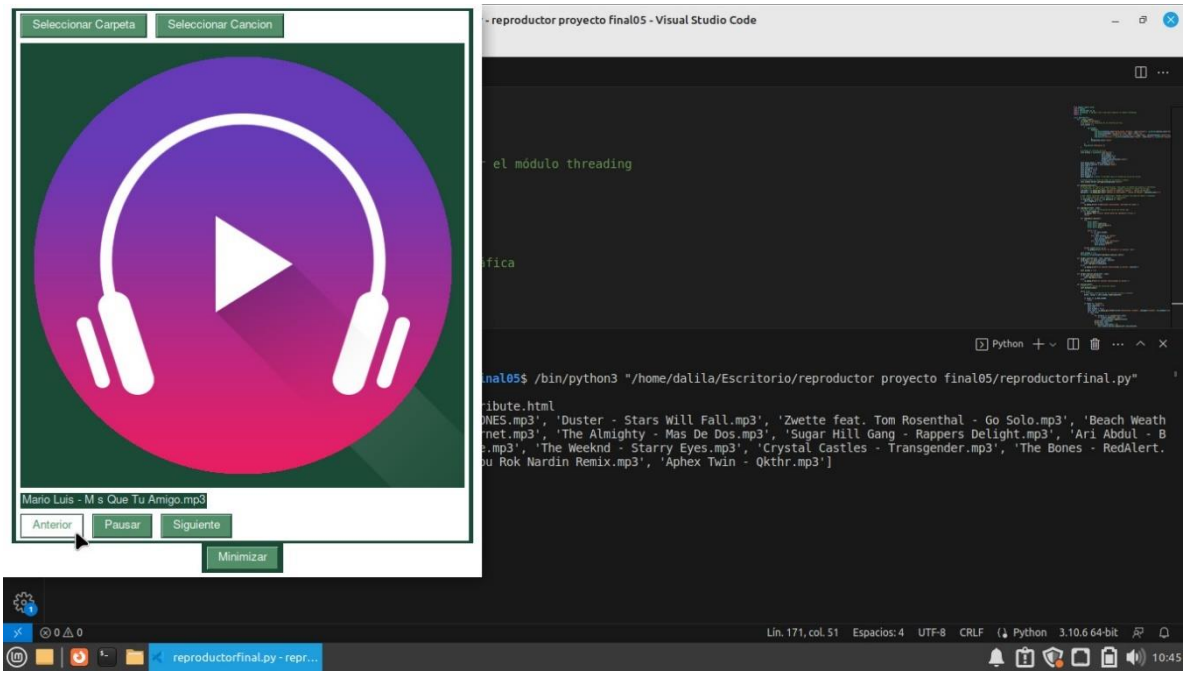
Podemos observar que cambia a la siguiente canción de la lista que está en la carpeta.



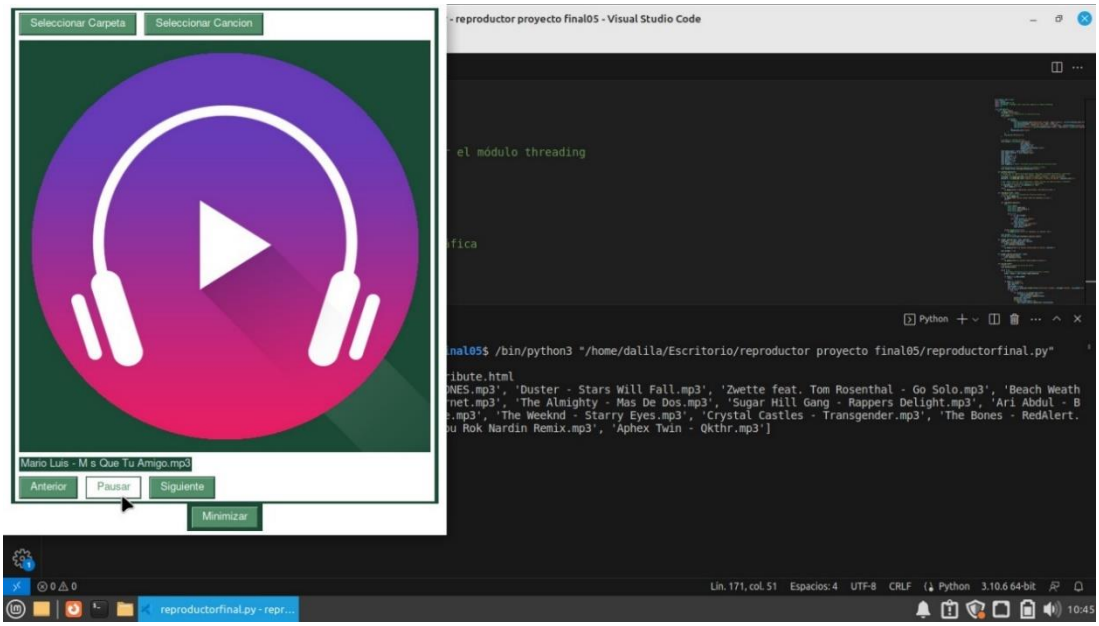
Y si nos posicionamos en el botón “Anterior” …



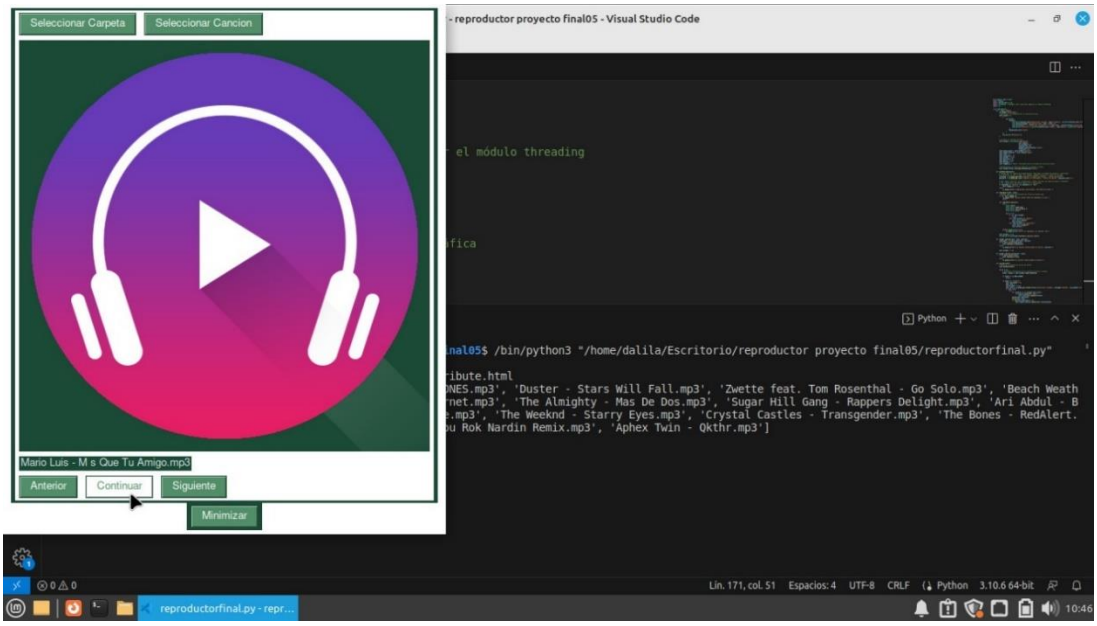
Observamos que cambia a la canción que fue reproducida anteriormente.



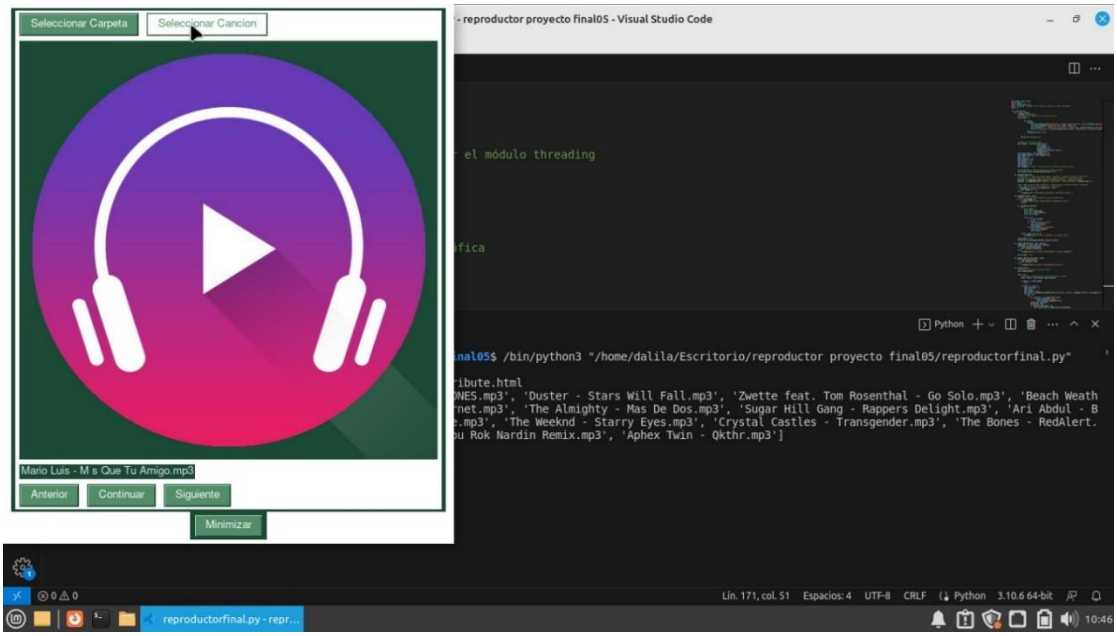
Ahora nos situamos en el botón “Pausar”.



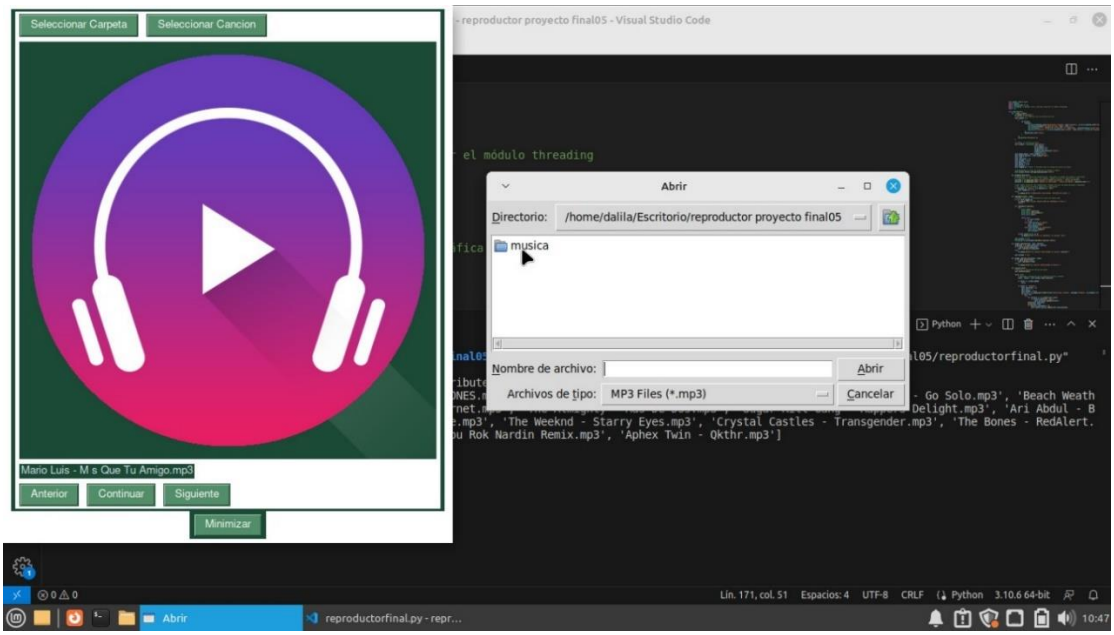
Si presionamos el botón, podemos observar que cambió de pausar a continuar, y al mismo tiempo la canción se detuvo.



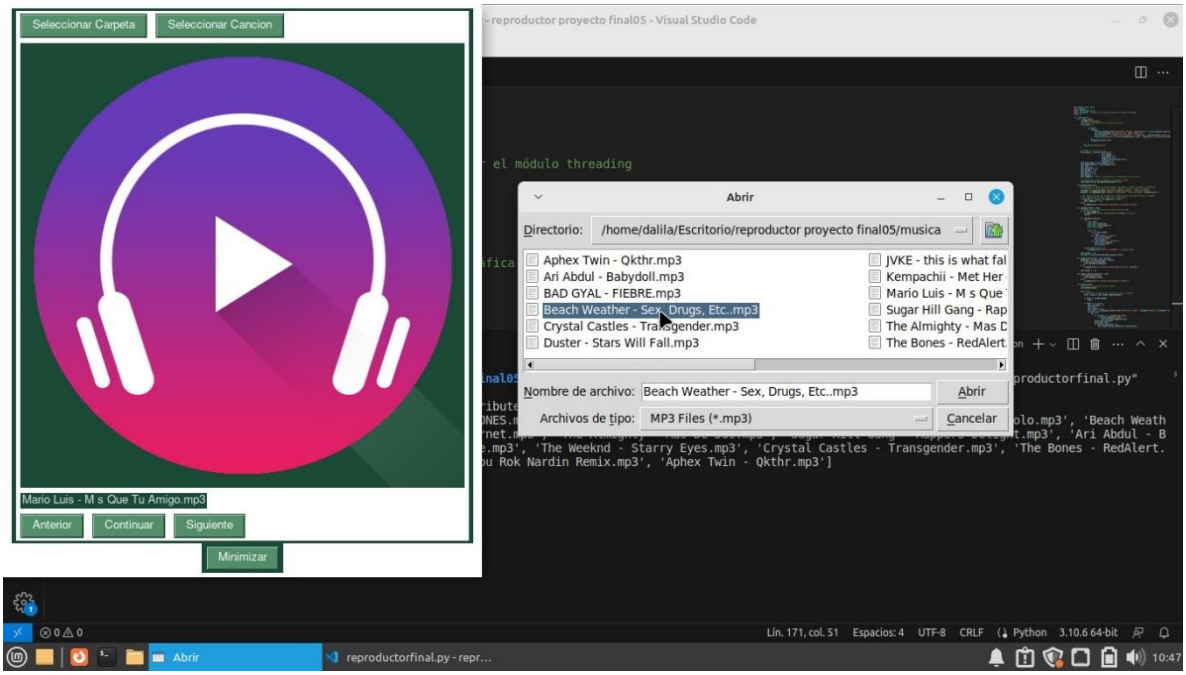
Ahora probemos la función “Seleccionar Canción”.



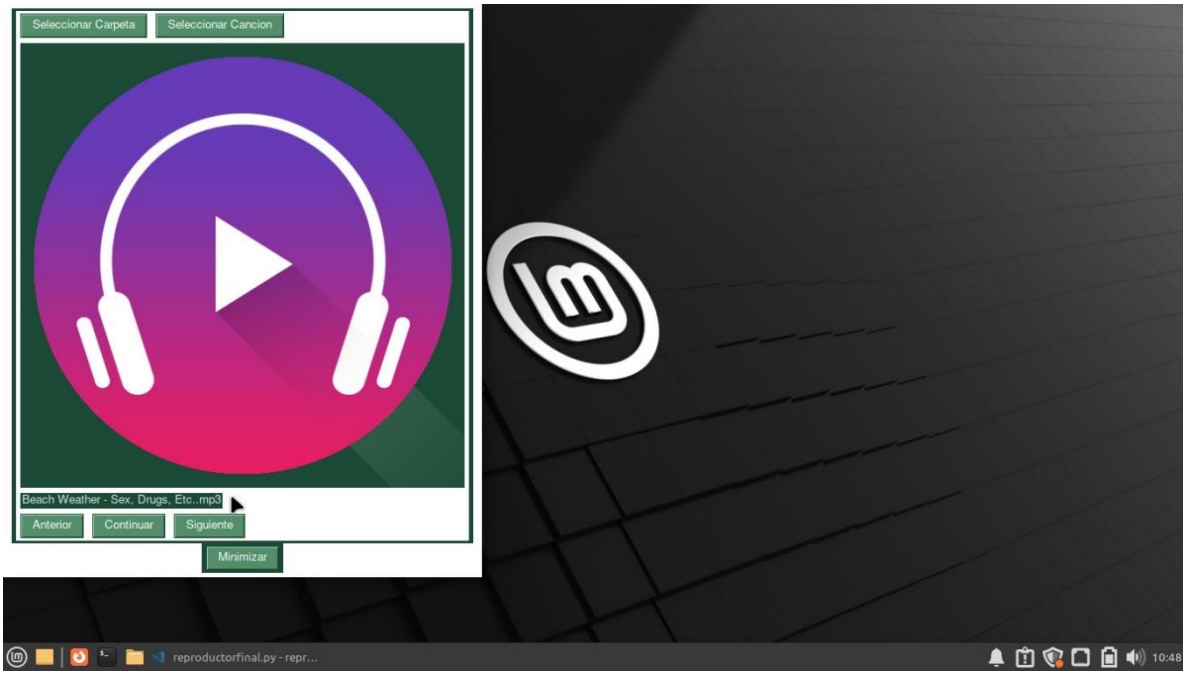
Al igual que “Seleccionar Carpeta”, se nos despliega una ventana para seleccionar la carpeta que contenga la música.



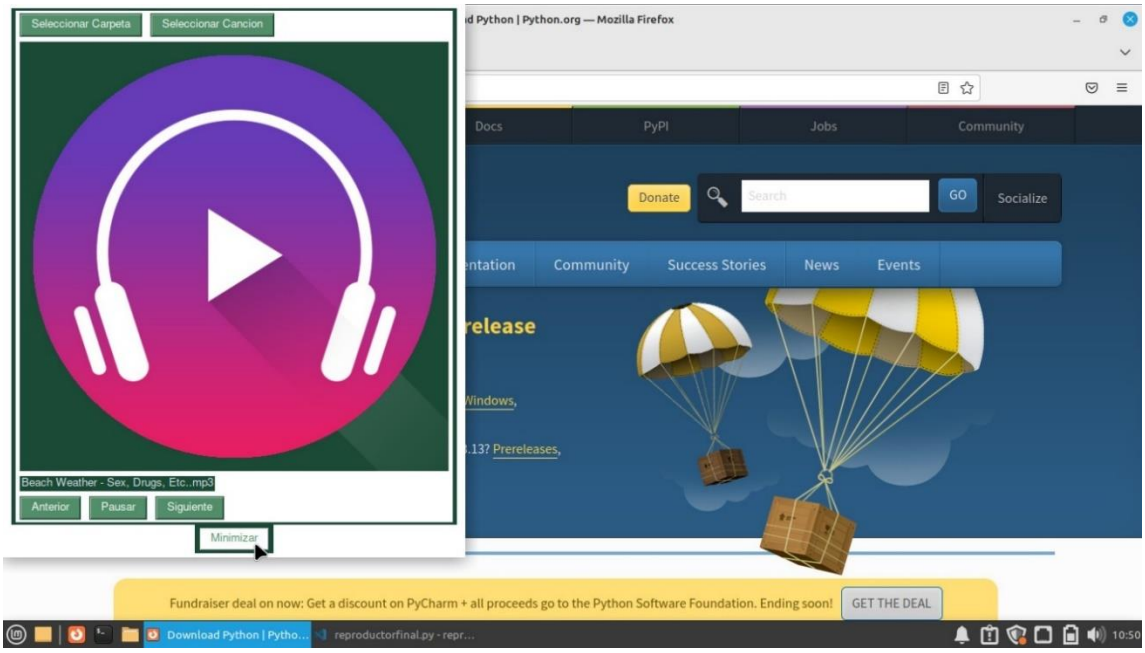
Luego buscamos la canción que queremos reproducir.



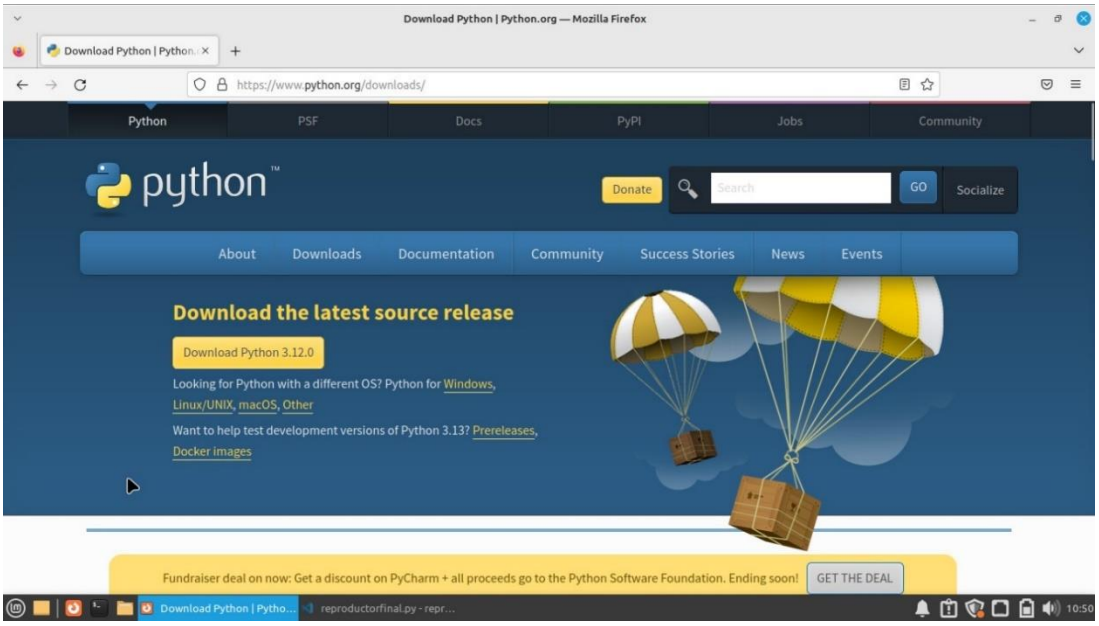
Damos clic en “Abrir” y automáticamente suena la canción seleccionada.



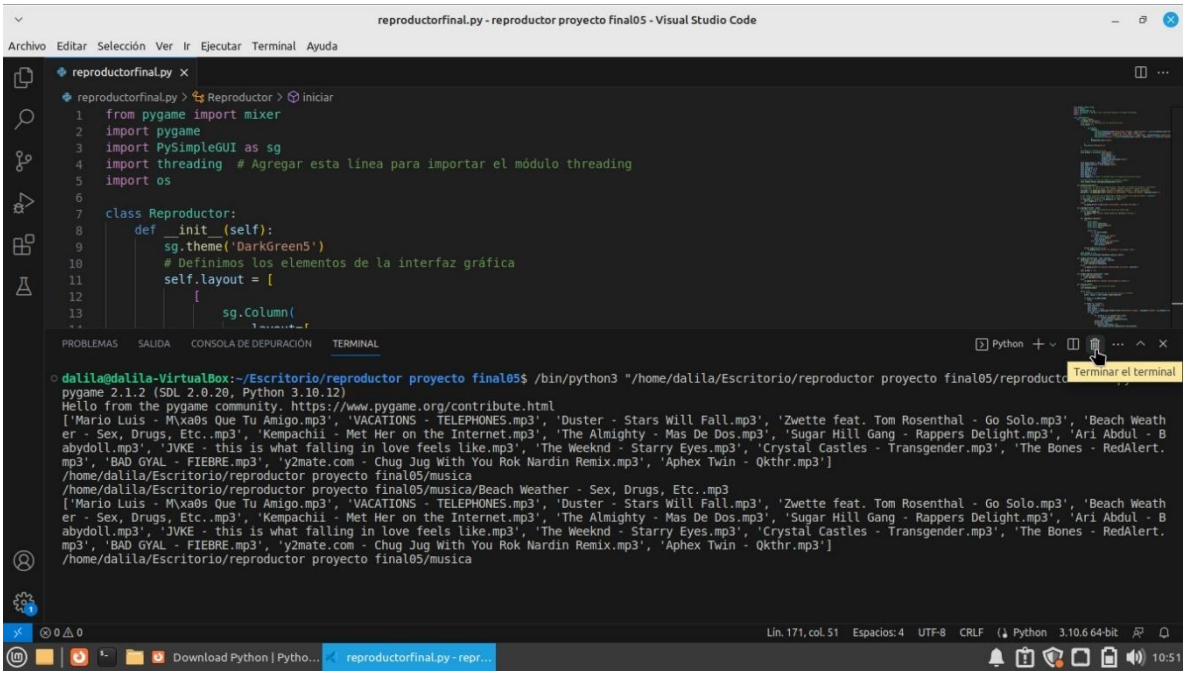
Otra función que realiza nuestro reproductor es la de “Minimizar” por si estamos haciendo otras tareas en el mismo ordenador.



Al darle clic a esta función, podemos observar que desaparece la ventana, pero la canción sigue sonando.



Por último, si queremos cerrar el programa por completo, solo presionamos el icono del basurero para terminar con la ejecución.



Incluir link de videotutorial de uso del programa, subido en plataforma como YouTube.

https://youtu.be/8MgG4hyxviY

Conclusiones de la primera etapa

Desarrollo de un Reproductor de Música Interactivo en Python:

La primera etapa del proyecto se centra en la planificación y definición de requisitos, tanto funcionales como no funcionales, para la creación de un reproductor de música interactivo en Python. El objetivo principal es proporcionar a los usuarios una interfaz gráfica intuitiva y funcional para gestionar su colección de música en formato MP3.

Se han identificado varios requisitos funcionales clave que el sistema debe cumplir para ofrecer una experiencia de usuario completa. Estos incluyen la capacidad de cargar canciones, reproducirlas y pausarlas, mostrar información de la canción en reproducción y permitir la

navegación entre canciones.

Se han establecido requisitos no funcionales importantes, como la facilidad de uso e inductividad,la compatibilidad con diferentes sistemas operativos, la necesidad de una interfaz gráfica atractivay personalizada, la seguridad del sistema y la capacidad de realizar otras tareas mientras se usa elreproductor.

Se menciona que el programa se desarrollará en Python y se utilizarán las bibliotecas pygame y Pysimplegui para la implementación. Esto demuestra una selección consciente de tecnologías que son adecuadas para la creación de una aplicación de este tipo.

Se describen las características clave que el reproductor de música tendrá, como la capacidad de cargar canciones, reproducirlas, pausarlas, mostrar información de la canción y permitir la navegación entre ellas. Estas características son esenciales para la funcionalidad básica del sistema.

Se destaca la importancia de una interfaz gráfica atractiva y fácil de usar. Esto es fundamental paragarantizar que los usuarios puedan interactuar de manera efectiva con el reproductor de música y disfrutar de una experiencia agradable.

Se menciona la necesidad de brindar seguridad en el sistema, aunque no se especifican detalles concretos sobre las medidas de seguridad que se implementarán. En futuras etapas del proyecto, se deberá definir y desarrollar estrategias de seguridad adecuadas.

Conclusiones etapa 2:

En la segunda etapa del proyecto se ha centrado en la creación de un reproductor de música interactivo confiable y seguro. La implementación de medidas de confiabilidad y seguridad, junto con la identificación y mitigación de vulnerabilidades, son pasos esenciales para garantizar que el software sea una opción sólida y confiable para los usuarios. Además, la aplicación de buenas prácticas de desarrollo y pruebas rigurosas contribuirá a un producto de alta calidad. Confiabilidad: En esta etapa, se ha dado un enfoque adecuado a la confiabilidad del software. Se han establecido requisitos para garantizar la disponibilidad del sistema, la capacidad de realizar copias de seguridad y la tolerancia a fallos. Estos elementos son cruciales para ofrecer un servicio ininterrumpido y confiable a los usuarios. Seguridad: La seguridad ha sido una prioridad en esta etapa del proyecto. Se han establecido requisitos de seguridad que incluyen la autenticación de usuarios, el control de acceso basado en roles y medidas de protección contra intrusiones. Esto garantiza que la información del usuario esté protegida y que el acceso no autorizado sea prevenido. Vulnerabilidades y Amenazas: Se ha reconocido la importancia de identificar y abordar las vulnerabilidades asociadas con las opciones de tecnología utilizadas. La aplicación de actualizaciones periódicas y la monitorización constante son estrategias efectivas para mitigar posibles amenazas informáticas.

Conclusiones etapa final.

El desarrollo de un reproductor de música interactivo en Python ha sido un proceso riguroso y detallado que ha abarcado desde la definición de requisitos hasta la identificación de riesgos y estrategias de mitigación. A lo largo de este documento, se ha delineado un camino claro para la creación de un software que no solo busca ofrecer una interfaz intuitiva y funcional para la reproducción de música, sino también considera aspectos cruciales como la seguridad, confiabilidad y adaptabilidad a diferentes plataformas.

La descripción de casos de uso ha permitido identificar y comprender las interacciones clave entre los usuarios y el sistema, ofreciendo una visión detallada de las funcionalidades requeridas. Asimismo, los requisitos funcionales y no funcionales han sentado las bases para garantizar la usabilidad, compatibilidad y seguridad del reproductor de música.

La identificación de vulnerabilidades asociadas con las tecnologías utilizadas y la definición de estrategias para gestionar los riesgos han sido fundamentales para mitigar posibles obstáculos a lo largo del desarrollo.

Este proyecto no solo se enfoca en la creación de un software de calidad, sino que también aborda aspectos críticos de seguridad, confiabilidad y adaptabilidad. La implementación de buenas prácticas de desarrollo, pruebas rigurosas y una gestión eficaz de riesgos son pilares fundamentales para el éxito y la entrega de un reproductor de música interactivo en Python que cumpla con las expectativas y necesidades de los usuarios.

Este documento proporciona una guía sólida y detallada que servirá como referencia a lo largo de todas las etapas del desarrollo del proyecto, asegurando un enfoque sistemático y orientado a resultados.

Bibliografía.

Python:

https://www.python.org/

Visual Studio Code:

https://code.visualstudio.com/

Pygame:

https://www.pygame.org/docs/

PySimpleGUI:

https://pypi.org/project/PySimpleGUI/

Tkinter:

https://tkinter.updatestar.com/es#google_vignette

Anexos

Autoevaluación

Autoevaluación		
Nombre del Estudiante: Sandra Marisol Tino Alfaro		
N°	**Aspecto a Evaluar**	**Rúbrica (Nota)**
1	Colaboré con las tareas que me fueron asignadas	9
2	Participé en forma activa en el trabajo de equipo	9
3	Mantuve comunicación con el equipo	9
4	Cumplí a tiempo con las actividades designadas	9
5	Aporté ideas de calidad	8
Total		44

Autoevaluación		
Nombre del Estudiante: Kenia Dalila Campos		
N°	**Aspecto a Evaluar**	**Rúbrica (Nota)**
1	Colaboré con las tareas que me fueron asignadas	9
2	Participé en forma activa en el trabajo de equipo	9
3	Mantuve comunicación con el equipo	9
4	Cumplí a tiempo con las actividades designadas	9
5	Aporté ideas de calidad	9
Total		45

Autoevaluación		
Nombre del Estudiante: Jason Adilman Serpas		
N°	**Aspecto a Evaluar**	**Rúbrica (Nota)**
1	Colaboré con las tareas que me fueron asignadas	9
2	Participé en forma activa en el trabajo de equipo	9
3	Mantuve comunicación con el equipo	9
4	Cumplí a tiempo con las actividades designadas	9
5	Aporté ideas de calidad	9
Total		45

Autoevaluación		
Nombre del Estudiante: Diego Andrés Miranda		
N°	**Aspecto a Evaluar**	**Rúbrica (Nota)**
1	Colaboré con las tareas que me fueron asignadas	9
2	Participé en forma activa en el trabajo de equipo	9

3	Mantuve comunicación con el equipo	9
4	Cumplí a tiempo con las actividades designadas	9
5	Aporté ideas de calidad	9
Total		45

Coevaluación

Nombre del Evaluador: Sandra Marisol Tino		
Nombre del Evaluado: Kenia Dalila Campos		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	10
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10
5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		50

Nombre del Evaluador: Sandra Marisol Tino		
Nombre del Evaluado: Jason Adilman Serpas		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	10
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10
5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		50

Nombre del Evaluador: Sandra Marisol Tino		
Nombre del Evaluado: Diego Andrés Miranda		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	9
3	Aportó al desarrollo del proyecto	9
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		45

Nombre del Evaluador: Kenia Dalila Campos		
Nombre del Evaluado: Jason Adilman Campos		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		47

Nombre del Evaluador: Kenia Dalila Campos		
Nombre del Evaluado: Sandra Marisol Tino		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10

5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		49

Nombre del Evaluador: Kenia Dalila Campos		
Nombre del Evaluado: Diego Andrés Miranda		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	9
3	Aportó al desarrollo del proyecto	9
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		45

Nombre del Evaluador: Jason Adilman Serpas		
Nombre del Evaluado: Diego Andrés Miranda		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	9
3	Aportó al desarrollo del proyecto	9
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		45

Nombre del Evaluador: Jason Adilman Serpas		
Nombre del Evaluado: Kenia Dalila Campos		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9

2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		47

Nombre del Evaluador: Jason Adilman Serpas		
Nombre del Evaluado: Sandra Marisol Tino		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	9
2	Demostró respeto y tolerancia hacia las opiniones de los demás	9
3	Aportó al desarrollo del proyecto	9
4	Propicia un clima de trabajo agradable	9
5	Antes de entregar la tarea, fue revisado por el evaluado	9
Total		45

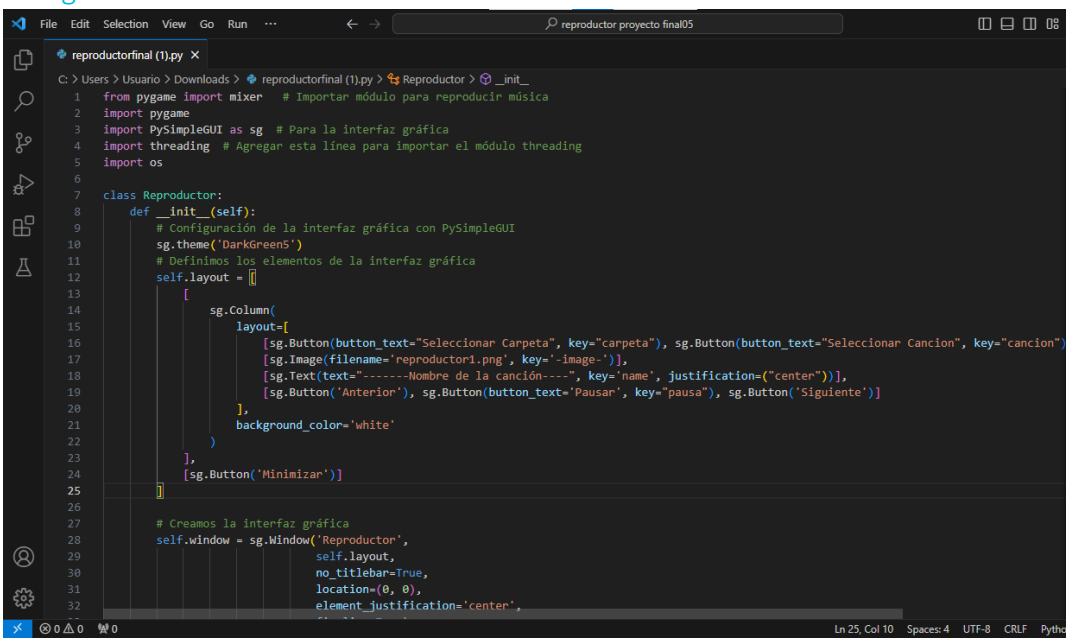
Nombre del Evaluador: Diego Andrés Miranda		
Nombre del Evaluado: Jason Adilman Serpas		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	10
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10
5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		50

Nombre del Evaluador: Diego Andrés Miranda

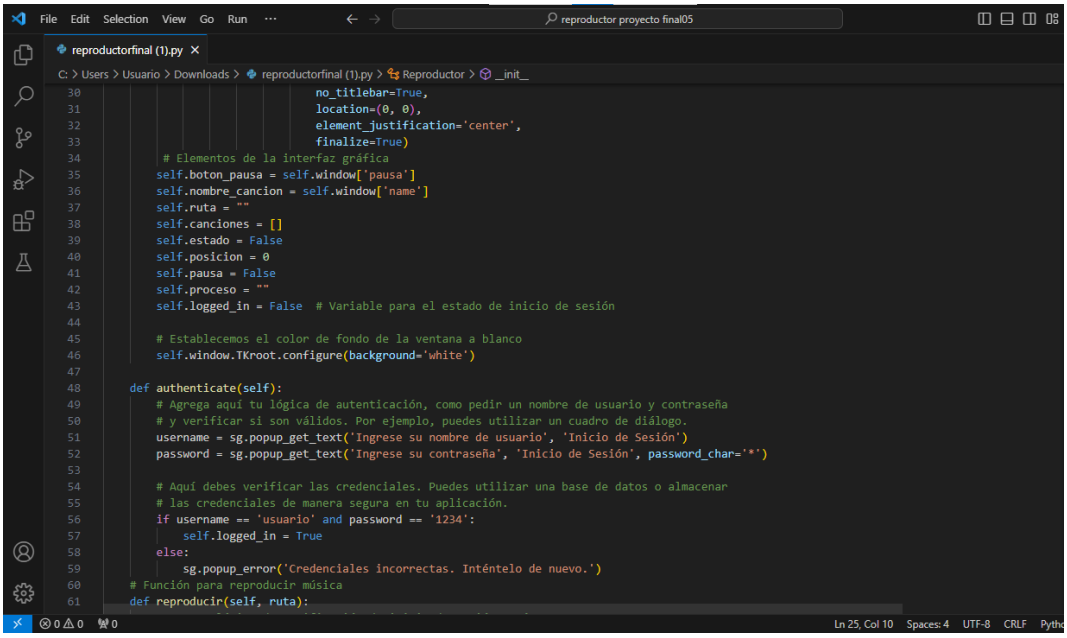
Nombre del Evaluado: Kenia Dalila Campos		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	10
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10
5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		50

Nombre del Evaluador: Diego Andrés Miranda		
Nombre del Evaluado: Sandra Marisol Tino		
N°	Aspecto a Evaluar	Rúbrica (Nota)
1	Demostró compromiso y responsabilidad con el grupo	10
2	Demostró respeto y tolerancia hacia las opiniones de los demás	10
3	Aportó al desarrollo del proyecto	10
4	Propicia un clima de trabajo agradable	10
5	Antes de entregar la tarea, fue revisado por el evaluado	10
Total		50

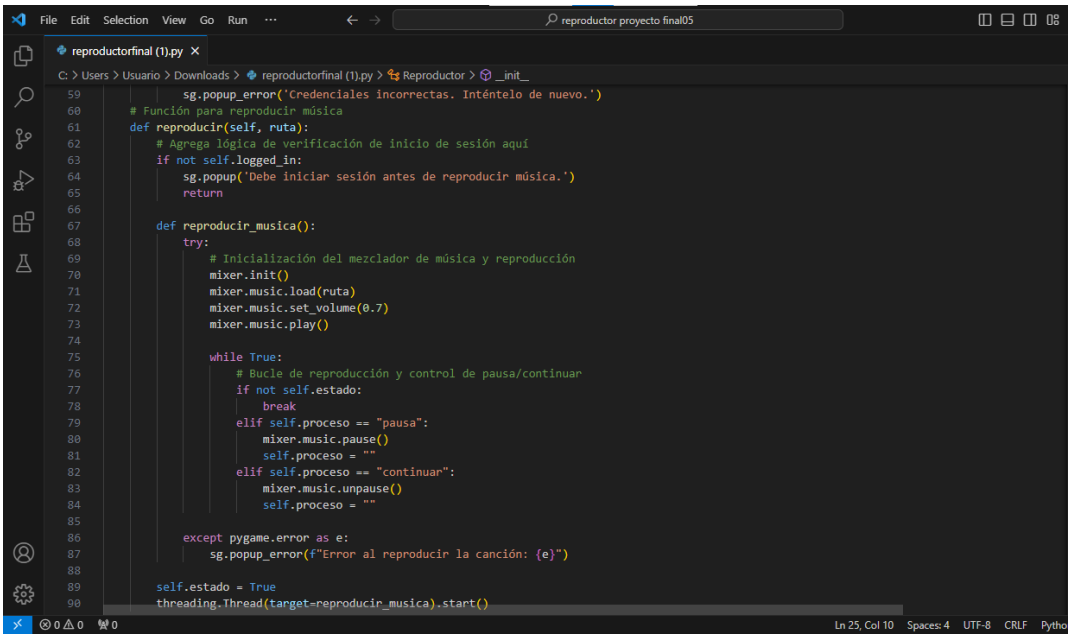
Código fuente

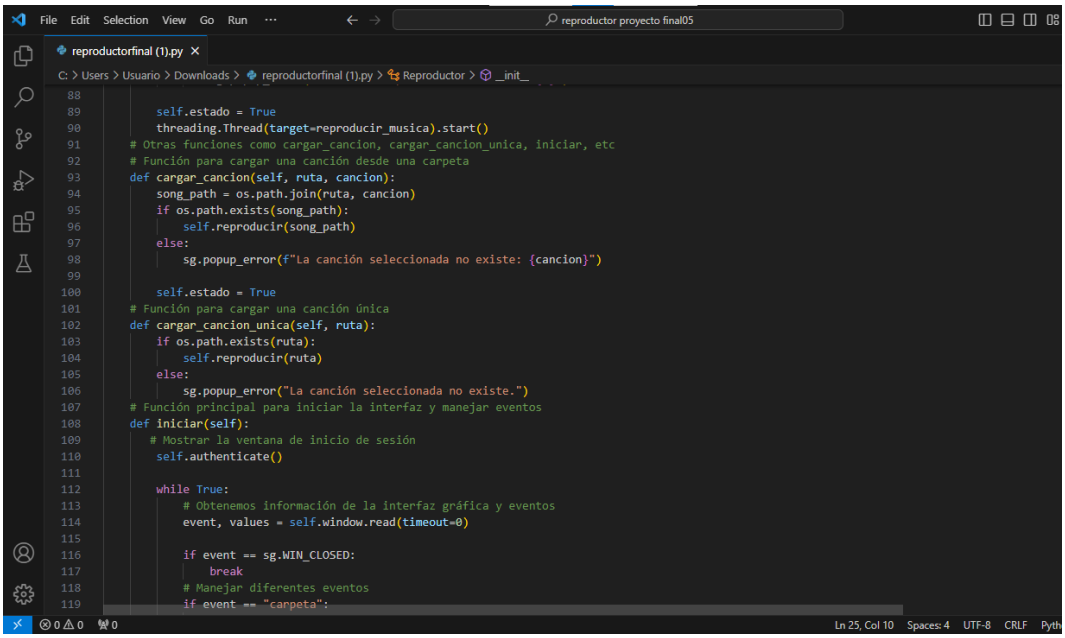


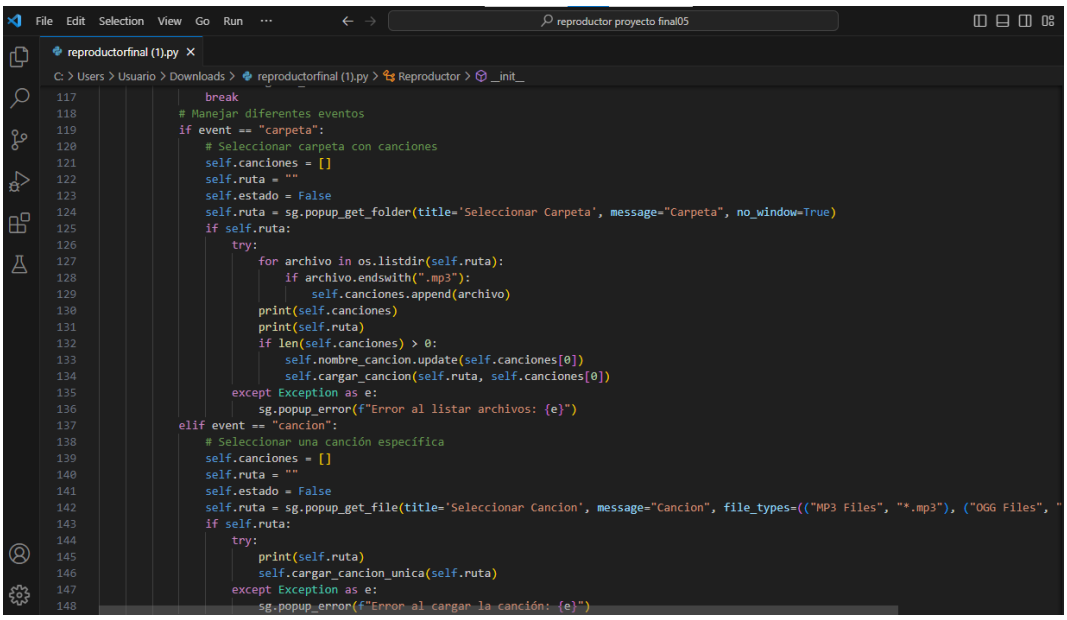
```python
from pygame import mixer   # Importar módulo para reproducir música
import pygame
import PySimpleGUI as sg  # Para la interfaz gráfica
import threading  # Agregar esta línea para importar el módulo threading
import os

class Reproductor:
    def __init__(self):
        # Configuración de la interfaz gráfica con PySimpleGUI
        sg.theme('DarkGreen5')
        # Definimos los elementos de la interfaz gráfica
        self.layout = [
            [
                sg.Column(
                    layout=[
                        [sg.Button(button_text="Seleccionar Carpeta", key="carpeta"), sg.Button(button_text="Seleccionar Cancion", key="cancion")],
                        [sg.Image(filename='reproductor1.png', key='-image-')],
                        [sg.Text(text="-------Nombre de la canción----", key='name', justification=("center"))],
                        [sg.Button('Anterior'), sg.Button(button_text='Pausar', key="pausa"), sg.Button('Siguiente')]
                    ],
                    background_color='white'
                )
            ],
            [sg.Button('Minimizar')]
        ]

        # Creamos la interfaz gráfica
        self.window = sg.Window('Reproductor',
                                self.layout,
                                no_titlebar=True,
                                location=(0, 0),
                                element_justification='center',
```

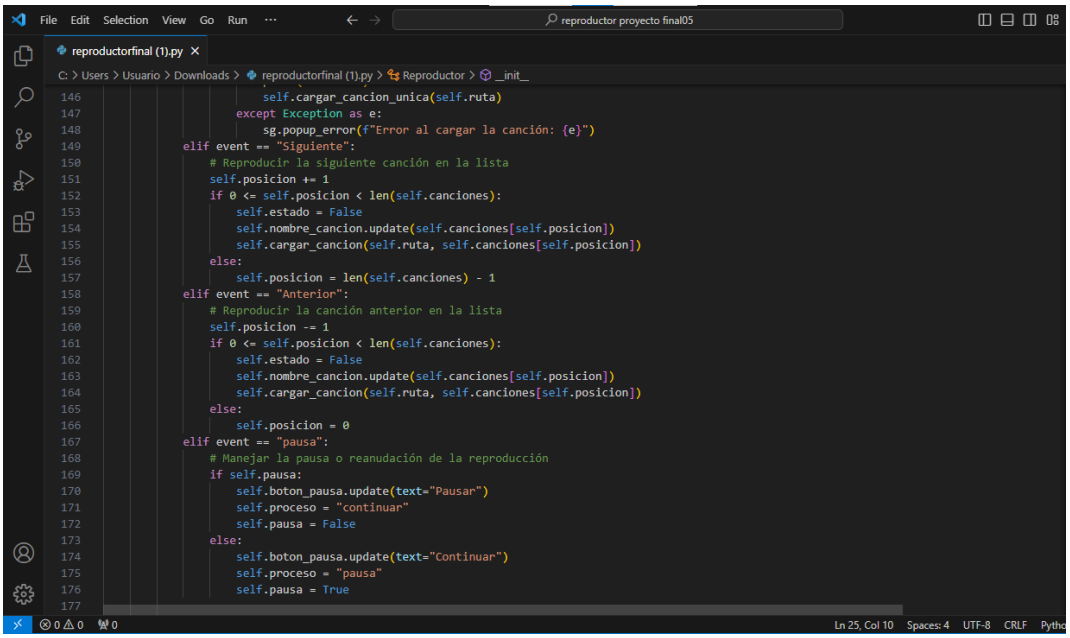


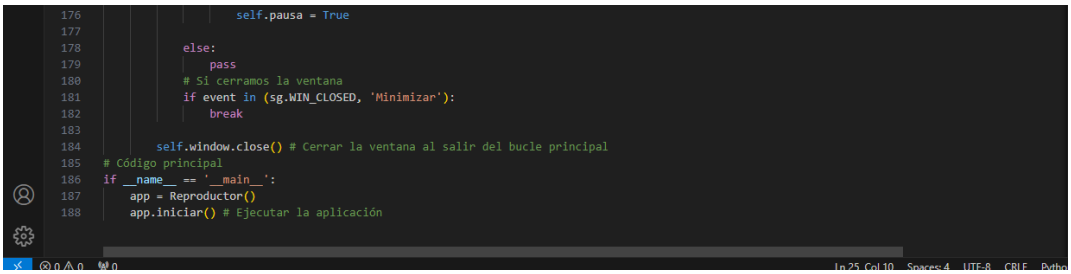
```python
                                no_titlebar=True,
                                location=(0, 0),
                                element_justification='center',
                                finalize=True)
        # Elementos de la interfaz gráfica
        self.boton_pausa = self.window['pausa']
        self.nombre_cancion = self.window['name']
        self.ruta = ""
        self.canciones = []
        self.estado = False
        self.posicion = 0
        self.pausa = False
        self.proceso = ""
        self.logged_in = False  # Variable para el estado de inicio de sesión

        # Establecemos el color de fondo de la ventana a blanco
        self.window.TKroot.configure(background='white')

    def authenticate(self):
        # Agrega aquí tu lógica de autenticación, como pedir un nombre de usuario y contraseña
        # y verificar si son válidos. Por ejemplo, puedes utilizar un cuadro de diálogo.
        username = sg.popup_get_text('Ingrese su nombre de usuario', 'Inicio de Sesión')
        password = sg.popup_get_text('Ingrese su contraseña', 'Inicio de Sesión', password_char='*')

        # Aquí debes verificar las credenciales. Puedes utilizar una base de datos o almacenar
        # las credenciales de manera segura en tu aplicación.
        if username == 'usuario' and password == '1234':
            self.logged_in = True
        else:
            sg.popup_error('Credenciales incorrectas. Inténtelo de nuevo.')
    # Función para reproducir música
    def reproducir(self, ruta):
```


File Edit Selection View Go Run ···
reproductor proyecto final05
reproductorfinal (1).py
C: > Users > Usuario > Downloads > reproductorfinal (1).py > Reproductor > __init__
sg.popup_error('Credenciales incorrectas. Inténtelo de nuevo.')
# Función para reproducir música
def reproducir(self, ruta):
# Agrega lógica de verificación de inicio de sesión aquí
if not self.logged_in:
sg.popup('Debe iniciar sesión antes de reproducir música.')
return
def reproducir_musica():
try:
# Inicialización del mezclador de música y reproducción
mixer.init()
mixer.music.load(ruta)
mixer.music.set_volume(0.7)
mixer.music.play()
while True:
# Bucle de reproducción y control de pausa/continuar
if not self.estado:
break
elif self.proceso == "pausa":
mixer.music.pause()
self.proceso = ""
elif self.proceso == "continuar":
mixer.music.unpause()
self.proceso = ""
except pygame.error as e:
sg.popup_error(f"Error al reproducir la canción: {e}")
self.estado = True
threading.Thread(target=reproducir_musica).start()
Ln 25, Col 10 Spaces: 4 UTF-8 CRLF Pytho


File Edit Selection View Go Run ···
reproductor proyecto final05
reproductorfinal (1).py
C: > Users > Usuario > Downloads > reproductorfinal (1).py > Reproductor > __init__
self.estado = True
threading.Thread(target=reproducir_musica).start()
# Otras funciones como cargar_cancion, cargar_cancion_unica, iniciar, etc
# Función para cargar una canción desde una carpeta
def cargar_cancion(self, ruta, cancion):
song_path = os.path.join(ruta, cancion)
if os.path.exists(song_path):
self.reproducir(song_path)
else:
sg.popup_error(f"La canción seleccionada no existe: {cancion}")
self.estado = True
# Función para cargar una canción única
def cargar_cancion_unica(self, ruta):
if os.path.exists(ruta):
self.reproducir(ruta)
else:
sg.popup_error("La canción seleccionada no existe.")
# Función principal para iniciar la interfaz y manejar eventos
def iniciar(self):
# Mostrar la ventana de inicio de sesión
self.authenticate()
while True:
# Obtenemos información de la interfaz gráfica y eventos
event, values = self.window.read(timeout=0)
if event == sg.WIN_CLOSED:
break
# Manejar diferentes eventos
if event == "carpeta":
Ln 25, Col 10 Spaces: 4 UTF-8 CRLF Pyth


File Edit Selection View Go Run ···
reproductor proyecto final05
reproductorfinal (1).py
C: > Users > Usuario > Downloads > reproductorfinal (1).py > Reproductor > __init__
117 break
118 # Manejar diferentes eventos
119 if event == "carpeta":
120 # Seleccionar carpeta con canciones
121 self.canciones = []
122 self.ruta = ""
123 self.estado = False
124 self.ruta = sg.popup_get_folder(title='Seleccionar Carpeta', message="Carpeta", no_window=True)
125 if self.ruta:
126 try:
127 for archivo in os.listdir(self.ruta):
128 if archivo.endswith(".mp3"):
129 self.canciones.append(archivo)
130 print(self.canciones)
131 print(self.ruta)
132 if len(self.canciones) > 0:
133 self.nombre_cancion.update(self.canciones[0])
134 self.cargar_cancion(self.ruta, self.canciones[0])
135 except Exception as e:
136 sg.popup_error(f"Error al listar archivos: {e}")
137 elif event == "cancion":
138 # Seleccionar una canción específica
139 self.canciones = []
140 self.ruta = ""
141 self.estado = False
142 self.ruta = sg.popup_get_file(title='Seleccionar Cancion', message="Cancion", file_types=(("MP3 Files", "*.mp3"), ("OGG Files", "
143 if self.ruta:
144 try:
145 print(self.ruta)
146 self.cargar_cancion_unica(self.ruta)
147 except Exception as e:
148 sg.popup_error(f"Error al cargar la canción: {e}")


File Edit Selection View Go Run ···
reproductor proyecto final05
reproductorfinal (1).py
C: > Users > Usuario > Downloads > reproductorfinal (1).py > Reproductor > __init__
146 self.cargar_cancion_unica(self.ruta)
147 except Exception as e:
148 sg.popup_error(f"Error al cargar la canción: {e}")
149 elif event == "Siguiente":
150 # Reproducir la siguiente canción en la lista
151 self.posicion += 1
152 if 0 <= self.posicion < len(self.canciones):
153 self.estado = False
154 self.nombre_cancion.update(self.canciones[self.posicion])
155 self.cargar_cancion(self.ruta, self.canciones[self.posicion])
156 else:
157 self.posicion = len(self.canciones) - 1
158 elif event == "Anterior":
159 # Reproducir la canción anterior en la lista
160 self.posicion -= 1
161 if 0 <= self.posicion < len(self.canciones):
162 self.estado = False
163 self.nombre_cancion.update(self.canciones[self.posicion])
164 self.cargar_cancion(self.ruta, self.canciones[self.posicion])
165 else:
166 self.posicion = 0
167 elif event == "pausa":
168 # Manejar la pausa o reanudación de la reproducción
169 if self.pausa:
170 self.boton_pausa.update(text="Pausar")
171 self.proceso = "continuar"
172 self.pausa = False
173 else:
174 self.boton_pausa.update(text="Continuar")
175 self.proceso = "pausa"
176 self.pausa = True
177
Ln 25, Col 10 Spaces: 4 UTF-8 CRLF Pytho



```
                    self.pausa = True

                else:
                    pass
                # Si cerramos la ventana
                if event in (sg.WIN_CLOSED, 'Minimizar'):
                    break

        self.window.close() # Cerrar la ventana al salir del bucle principal
# Código principal
if __name__ == '__main__':
    app = Reproductor()
    app.iniciar() # Ejecutar la aplicación
```