UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

PROGRAMACIÓN DE UN DOWNSAMPLER

Autores: Marta Enrich i Rafel Eduardo Moncayo
Assignatura: Algorísmia i Programació Audiovisual (APA)
UPC ESEIAAT
01/06/2023

Tabla de contenidos

1. Introducción

Este proyecto “Samp-Less” consiste en la programación de un efecto de “downsampling” en el lenguaje de programación Python. El downsampling es una técnica de tratamiento de datos usado en muchos ámbitos desde procesamiento de imagen y audio hasta en aplicaciones con grandes conjuntos de datos como monitoreo de sensores o incluso en redes sociales y análisis de texto. En este trabajo se busca investigar las oportunidades que surgen en el ámbito del audio e imagen al aplicar downsampling y su consecuente pérdida de datos.

Desde el inicio de la música y la imagen en digital, en el ámbito del procesamiento de datos, se ha hecho mucho hincapié en la mejora de la calidad del sonido y la imagen. Sin embargo, en la música, no fue hasta finales de la década de los 80 y principios de los 90 cuando el efecto de downsampling, junto con el bit-crushing (otra técnica de reducción de calidad), ganó popularidad en los géneros de música electrónica y experimental. En particular, artistas como Aphex Twin fueron pioneros en su uso y tuvieron un impacto significativo en la evolución del sonido.

En imagen, este efecto se empieza a implementar durante los 70 por los investigadores y los pioneros en el campo de la imagen digital y comenzaron a explorar técnicas de reducción de resolución. No sería hasta los 80 en los que este efecto se usa de una manera más artística con los gráficos para videojuegos.

Este trabajo representa una manera de entender cómo es el tratamiento de los datos en los audiovisuales con el fin de recrear este efecto de deterioro de señal, y de esta manera aproximarnos a la música y a las imágenes con las que hemos crecido durante tantos años.

SAMP-LESS

2. Objetivos

El objetivo principal desde el inicio ha sido crear una aplicación con la que se pudiera introducir un archivo de audio y que se pudiera reducir la calidad de este en función de un valor introducido por el usuario.

También se ha tenido en mente hacer una interfaz gráfica sencilla para que el usuario pudiera trabajar en esta sin ninguna dificultad por mucho que no sepa que sucede a nivel de procesamiento del sonido. Al querer trabajar de la misma manera en el ámbito de las imágenes, se plantea la idea de representar la degradación del audio con la pérdida de calidad de una imagen de decoración dentro de la misma aplicación para hacer más visual el cambio y más atractivo para el usuario.

La finalidad intrínseca de esta aplicación como tal es meramente creativa. No pretende resolver ningún problema relacionado con el control de calidad, simplemente se busca expandir las posibilidades del productor musical a la hora de encontrar nuevos sonidos y nuevas texturas para experimentar con el sonido en todos los aspectos.

Su funcionamiento consiste en que al bajar la frecuencia de muestreo se produce un efecto que se llama Aliasing que ocurre al digitalizar una señal de audio a una frecuencia por debajo de la frecuencia de Nyquist. En la mayoría de los casos, este efecto es no deseado, pero no ocurre así en Samp-Less ya que estas modulaciones entre frecuencias alias es lo que hace que podamos sacar nuevos sonidos y texturas y por tanto son el objetivo de esta aplicación.

3. Enfoque de desarrollo

Por tal de llevar a cabo el desarrollo de la aplicación, se ha dividido el proyecto en dos partes diferenciadas.

Por un lado nos encontramos la parte del desarrollo del **backend**. Este se ha dividido en dos partes:

- En esta primera, se ha desarrollado todo el código que tiene que ver con las funcionalidades de la lectura y escritura de ficheros, y el procesado de audio para poder modificar la frecuencia de muestreo del fichero a modificar.
- En este segundo apartado, se ha centrado en poner una imagen que da soporte visual a la hora de ver como de distorsionado se puede escuchar el audio que se está modificando. Esta imagen se vuelve más distorsionada a medida que el audio es degradado.

Por otro lado, tenemos una parte de **frontend**. En esta, se ha desarrollado el código en el que el usuario puede interactuar con las funciones de la aplicación. Se ha diseñado un entorno gráfico que consta de diversos botones (browse, play, pause y download) y de un fader que nos facilita el input del porcentaje de downsampling que se le quiere aplicar al audio. A parte se muestra una imagen en el lateral derecho con la que se puede ver de forma visual la bajada de calidad que se aplica en el audio.

4. Tecnologías y Herramientas utilizadas

Para poder realizar este proyecto se han usado los siguientes lenguajes de programación, frameworks, librerías y herramientas:

4.1. Lenguajes de programación

4.1.1. Python

Se ha usado Python como lenguaje principal para desarrollar la aplicación. Python es un lenguaje de programación que cuenta con una amplia variedad de bibliotecas y módulos que facilitan el desarrollo de aplicaciones gráficas y el procesamiento de imágenes y audio. Con estas bibliotecas hemos podido analizar, leer, escribir el fichero de audio. También gracias a las librerías hemos podido diseñar un entorno gráfico para así poder interactuar con el funcionamiento de la aplicación de una forma más ágil

4.2. Frameworks y librerías

4.2.1. Tkinter

Tkinter es la biblioteca estándar de Python para crear interfaces gráficas de usuario (GUI). Tkinter se ha escogido por su facilidad de uso e integración con Python. Con este se ha implementado una ventana en la que se ha subido una imagen de plantilla y seguidamente se han implementado un fader para poder regular el downsampling. También se han implementado con el varios botones que permiten la selección del archivo a modificar, la descarga del fichero de audio modificado y la reproducción y pausa del mismo en ese preciso momento.

4.2.2. Python Imagining Library (PIL)

PIL es una biblioteca de procesamiento de imágenes de Python que se utiliza para cargar, redimensionar y mostrar imágenes en la interfaz gráfica de la aplicación. También proporciona una amplia gama de funciones para el procesamiento de imágenes. Con ella hemos podido incorporar y redimensionar las imágenes que se muestran en la interfaz gráfica de la aplicación.

4.2.3. Sounddevice

Sounddevice es una biblioteca de Python que se utiliza para reproducir archivos de audio. En el proyecto, se utilizó para reproducir el audio después de aplicar el proceso de downsampling. Este es combinado con la función de Play del entorno gráfico.

4.2.4. Soundfile

Soundfile es una biblioteca de Python que se utiliza para leer y escribir archivos de audio en varios formatos. Dentro del proyecto se ha utilizado en funciones para trabajar con archivos de audio y obtener información sobre ellos, como la tasa de muestreo y los datos de audio.

4.2.5. NumPy

NumPy es una biblioteca de Python que se utiliza para trabajar con matrices y operaciones numéricas. En el proyecto, se utilizó para manipular los datos de audio y realizar el proceso de downsampling.

4.3. Herramientas

4.3.1. GitHub

GitHub es una plataforma de desarrollo colaborativo que se utiliza para almacenar y gestionar el código fuente del proyecto. Se eligió GitHub para facilitar la colaboración entre los miembros del equipo y para mantener un control de versiones del proyecto.

5. Diseño y arquitectura

El diseño del diseño está basado en un software creado con Python y Tkinter. Los principales componentes y cómo interactúan es lo siguiente:

- **Clase FaderApp**: Esta clase es la aplicación principal, derivada de la clase Tk de Tkinter. Es responsable de crear y administrar la ventana de la aplicación. Incluye los elementos de la interfaz de usuario como botones, etiquetas, navegación e imágenes de fondo. Además, es responsable de manejar las interacciones del usuario, como la selección de archivos, el movimiento de audio y la reproducción de audio.
- **Interfaz de usuario**: la ventana de la aplicación contiene muchos elementos de la interfaz de usuario. Contiene el fader y botones de acción (cómo seleccionar archivos, reproducir/detener música y descargar archivos). El fader se encarga de cambiar la calidad del sonido del fichero de audio seleccionado y de la imagen del busto.
- **Módulo backend**: el archivo "backend.py" contiene funciones relacionadas con el procesamiento del audio. Incluye funciones para leer archivos de audio, descargar archivos de audio y escribirlos en el formato deseado. El equipo de FaderApp utiliza estas funciones para acceder al audio, aplicar faders, reproducir o descargar archivos.
- **Módulo backendIMG**: El archivo "backendIMG.py" contiene funciones de procesamiento de imágenes. En particular, incluye la capacidad de reducir la calidad de la imagen en función del valor del fader. La clase FaderApp usa esta función para obtener la imagen modificada y mostrarla en la pantalla.

Las decisiones de diseño son las siguientes:

- **Elección de Python y Tkinter**: Python es el lenguaje más utilizado y muchas bibliotecas admiten el desarrollo de programas de software. Tkinter es una biblioteca popular de Python para crear interfaces visuales, lo que la convierte en una interfaz de programación natural y fácil de usar.
- **Separación funcional**: Se utilizó una arquitectura prototipo donde el procesamiento de audio y el procesamiento de imágenes estaban en módulos separados. Esto permite una mayor flexibilidad y reutilización de código, ya que el procesamiento de audio y el procesamiento de imágenes se pueden usar de forma independiente.
- **Interacción entre componentes**: El grupo FaderApp es donde se integran y organizan las actividades. Funciona con componentes de interfaz de usuario para admitir la interacción del usuario y utiliza servicios de backend y componentes backendIMG para crear audio y gráficos. La comunicación entre unidades se realiza a través de llamadas de servicio y transferencias de datos.
- **Uso de bibliotecas especializadas**: PIL, herramientas de sonido, archivos de sonido y bibliotecas NumPy se han utilizado para tareas específicas como gráficos y audio. Estas bibliotecas proporcionan funciones y optimizaciones avanzadas para facilitar el desarrollo y mejorar el rendimiento.

6. Implementación

El proceso de implementación del sistema se divide en las siguientes etapas:

- **Planificación del desarrollo:** Python es el lenguaje de programación que se usará para el desarrollo. En él se utilizaran bibliotecas como Python Tkinter para crear interfaces gráficas de usuario. También nos encontramos con PIL (Pillow), una librería con herramientas de reproducción de audio (sounddevice), para obtener datos de los ficheros de sonido (soundfile) y otras bibliotecas como NumPy para el procesamiento de imágenes y sonido.
- **Backend y BackendIMG:** Estas funciones incluyen la lectura de archivos de audio, la descarga, la escritura de archivos de audio y la edición de imágenes.
- **Diseño de interfaz de usuario:** la interfaz de usuario está diseñada utilizando la biblioteca Tkinter. Los elementos básicos como texto, botones y paneles creados se colocan en la ventana de tareas principal.
- **Implementación de clases y funciones:** implementación de la clase FaderApp, herencia de Tkinter y como punto de entrada. Los componentes de la interfaz de usuario están integrados con funciones backend y backendIMG para facilitar la interacción del usuario y crear efectos visuales y de audio.
- **Pruebas y depuración:** Se han realizado pruebas para que la aplicación funcione correctamente. Se han corregido errores y realizado los cambios necesarios en la implementación y su correcto funcionamiento.

7. Resultados y posibles nuevas funcionalidades

El resultado del programa son las siguientes tareas completadas:

- **Seleccionar archivo de audio:** El usuario puede seleccionar un archivo de audio de sus archivos y el programa mostrará el nombre del archivo especificado.
- **Control vía fader:** El usuario puede ajustar el nivel de reducción de frecuencia de muestreo para seleccionar la cantidad de pérdida de calidad deseada. Al mover el fader para ajustar el audio, y hacer play veremos una degradación de la imagen que también va a depender de el nivel en que estés situado el fader.
- **Reproducción de audio:** el usuario puede usar el botón "Reproducir" para reproducir un archivo de audio del formato seleccionado. Ajuste en consecuencia presionando un botón y reproduzca el sonido resultante.
- **Representación gráfica de la degradación de calidad:** A medida que el audio pierde calidad la imagen que tenemos a la derecha del programa nos sirve como una representación aproximada de la pérdida de la calidad del audio.
- **Descargar archivos de audio:** los usuarios pueden descargar archivos en el formato especificado haciendo clic en el botón "Descargar". Al presionar el botón se realizarán los cambios correspondientes y se guardará el archivo entrante en la misma ubicación del proyecto.

Estas funciones permiten al usuario interactuar con el programa entendiendo y realizando las operaciones de audio y con el soporte de la interfaz gráfica y el soporte visual de la imagen.

También comentar que como futuras implementaciones para el proyecto nos gustaría contemplar la siguientes posibilidades:

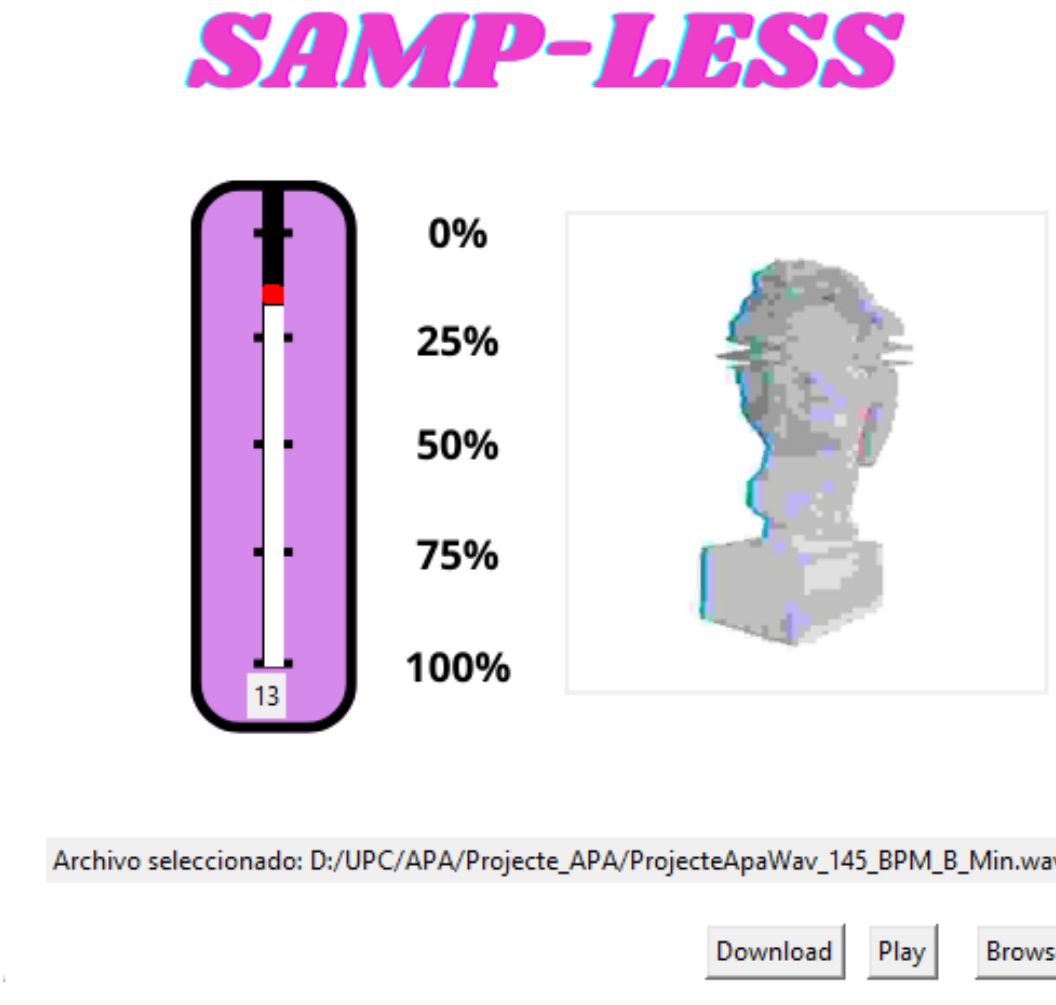
- **Reproducción a tiempo real:** poder reproducir los ficheros de audio de forma que no se tenga que dar al play y al pause para poder escuchar el audio cambiado, sino que este estuviese en reproducción continua, y que mientras se subiera y bajara el fader tener el audio en el que se escuchara los cambios.
- **Diseño de una aplicación web:** Poder implementar este mismo programa pero que esté disponible para usarse desde una interfaz web.
- **Programa para imagen:** poder realizar los mismos cambios que nos ofrece esta aplicación en el audio, pero en este caso aplicarlos a imágenes para bajarles las calidad.

8. Conclusiones

En términos de objetivos del proyecto, se ha logrado desarrollar una aplicación funcional que cumple con los requisitos establecidos inicialmente. Se han implementado las funcionalidades principales y se logró una interacción fluida con los archivos de audio. El proyecto se ha desarrollado con éxito y da la posibilidad a los usuarios de controlar la calidad de reproducción y descarga de los archivos de audio de manera efectiva.

En general, el proyecto proporciona la posibilidad al usuario de poder establecer un control sobre la pérdida calidad de audio que se le quiere proporcionar al fichero. También se le proporciona al usuario un apoyo visual para poder comprender como es la distorsión en ese momento.

Finalmente, y para concluir, comentar que este proyecto ha sido una buena oportunidad para aplicar los conocimientos adquiridos en clase, y de aprender ya de forma paralela, cómo diseñar pequeños entornos gráficos usando librerías de python.



ANNEX

ANNEX 1. Cómo ejecutar l'aplicación

El código del proyecto está recogido en el siguiente repositorio de GitHub, donde puedes descargarlo:

https://github.com/m4rt401/Projecte_APA.git

Para poder ejecutar el código de la aplicación, primero debes tener instalados los requisitos para evitar conflictos en la ejecución con diferentes versiones de las bibliotecas. Estas bibliotecas se pueden encontrar en el archivo "requeriment.txt" y son las siguientes:

Python

```
asttokens==2.2.1
backcall==0.2.0
cffi==1.15.1
colorama==0.4.6
comm==0.1.3
debugpy==1.6.7
decorator==5.1.1
executing==1.2.0
ipykernel==6.23.0
ipython==8.13.2
jedi==0.18.2
jupyter_client==8.2.0
jupyter_core==5.3.0
matplotlib-inline==0.1.6
nest-asyncio==1.5.6
numpy==1.24.3
packaging==23.1
parso==0.8.3
pickleshare==0.7.5
Pillow==9.5.0
platformdirs==3.5.1
prompt-toolkit==3.0.38
psutil==5.9.5
pure-eval==0.2.2
pycparser==2.21
pygame==2.4.0
Pygments==2.15.1
python-dateutil==2.8.2
pywin32==306
```

```
pyzmq==25.0.2
six==1.16.0
sounddevice==0.4.6
soundfile==0.12.1
stack-data==0.6.2
tornado==6.3.1
traitlets==5.9.0
wcwidth==0.2.6
```

Para poder realizar la instalación, desde la misma carpeta donde se encuentra el archivo, debes escribir el siguiente comando en la terminal::

Unset

```
$ pip install -r requirements.txt
```

Para ejecutar el código, debes ubicarte en el archivo "frontendplus.py" y ejecutarlo.

ANNEX 2. Código de la aplicación

Los archivos de código de toda la aplicación son los siguientes:

frontendplus.py

Python

```python
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import sounddevice as sd
import backend as bk
import backendIMG as bkim

class FaderApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.geometry("500x500")
        self.title("Fader App")
        self.value = tk.IntVar(value=0)

        # Cargar la imagen de fondo
        self.image = ImageTk.PhotoImage(file="DOWNSAMPLING.png")

        # Crear un widget Label y establecer la imagen de fondo
        background_label = tk.Label(self, image=self.image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)

        # Cargar la imagen "img.jpg"
        self.img = Image.open("Proba.jpg")
        self.img = self.img.resize((220, 220))  # Redimensionar
la imagen a 220x220

        # Convertir la imagen a PhotoImage
        self.img_tk = ImageTk.PhotoImage(self.img)

        # Crear un widget Label para mostrar la imagen
        img_label = tk.Label(self, image=self.img_tk)
```

```
        img_label.place(x=260, y=140)  # Posición deseada de la
imagen

        self.canvas = tk.Canvas(self, width=10, height=225,
bg="black", highlightthickness=0)
        self.canvas.place(x=120, y=126)  # Posición deseada del
fader

        self.fader = self.canvas.create_rectangle(0, 230, 280, 0,
fill="white")
        self.knob = self.canvas.create_oval(-5, -5, 15, 15,
fill="red")

        self.canvas.bind("<B1-Motion>", self.move_knob)
        self.canvas.bind("<Button-1>", self.move_knob)

        self.label = tk.Label(self, textvariable=self.value)
        self.label.place(x=113, y=354)  # Posición deseada de la
etiqueta

        self.selected_file_label = tk.Label(self, text="Archivo
seleccionado: ")
        self.selected_file_label.place(x=20, y=430)  # Posición
deseada de la etiqueta de ruta seleccionada

        self.browse_button = tk.Button(self, text="Browse",
command=self.open_file_dialog)
        self.browse_button.place(x=450, y=470)  # Posición
deseada del botón de navegación

        self.play_button = tk.Button(self, text="Play",
command=self.play_pause_file)
        self.play_button.place(x=400, y=470)  # Posición deseada
del botón de reproducción
        self.is_playing = False

        self.download_button = tk.Button(self, text="Download",
command=self.download_file)
```

```
        self.download_button.place(x=325, y=470) # Posición
deseada del botón de descarga

    def move_knob(self, event):
        y = event.y
        if y < 25:
            y = 25
        elif y > 225:
            y = 225
        self.canvas.coords(self.fader, 0, y, 300, 230)  # Ajustar
coordenadas del rectángulo
        self.canvas.coords(self.knob, -5, y-5, 15, y+5)
        self.value.set(int((y-25) * 100 / 200))

    def open_file_dialog(self):
        '''
        Función que nos permite buscar un fichero de audio por
nuestro ordenador para poderlo usar más adelante.
        '''
        file_path = filedialog.askopenfilename()
        self.org_data, self.samplerate, self.info =
bk.lectura_audio(file_path)

        if file_path:
            self.selected_file_label.config(text="Archivo
seleccionado: " + file_path)

    def play_pause_file(self):
        '''
        Función que regula el boton de play/pause.
        Cada vez que se da al play se consigue el valor en el que
esta el fader(porcentaje de calidad) para poder reproducir la
señal downsampleada.
        En el momento en el que se da Play, este mismo botón
cambia a ser un botón de Pause y se se apreta lo que hace es
parar de reproducir y prepararse para volver a reproducir el
audio.
        '''
        if self.is_playing == False:
```

```
            fader=self.value.get()
            self.data_DWS, self.samplerate, self.info =
bk.downsampler(self.org_data, self.samplerate, self.info, fader)

            #Imagen
            imgpath=bkim.downsample_img("Proba.jpg", fader)
            self.img = Image.open(imgpath)
            self.img = self.img.resize((220, 220))  #
Redimensionar la imagen a 220x220
            # Convertir la imagen a PhotoImage
            self.img_tk = ImageTk.PhotoImage(self.img)
            # Crear un widget Label para mostrar la imagen
            img_label = tk.Label(self, image=self.img_tk)
            img_label.place(x=260, y=140)  # Posición deseada de
la imagen


            sd.play(self.data_DWS, self.samplerate)
            self.is_playing = True
            self.play_button.config(text="Pause")
        else:

            sd.stop()
            self.is_playing = False
            self.play_button.config(text="Play")

    def download_file(self):
        '''
        Función que, de manera independiente a la posición del
fader con la que se haya dado a reproducir, agarra el valor de
fader en el que esté
        colocado y escribe un fichero wave con ese porcentaje de
calidad. Para ello se usa la función "escritura_wave" definida en
back end que
        agarra los datos y los codifica en formato wave. Este
fichero nuevo tiene como salida la misma carpeta del proyecto.
        '''
```

```
        self.data_DWS, self.samplerate, self.info =
bk.downsampler(self.org_data, self.samplerate,
self.info,self.value.get())
        bk.escritura_wave(self.data_DWS, self.samplerate,
self.info)

if __name__ == "__main__":
    app = FaderApp()
    app.mainloop()
```

backend.py

Python

```
import soundfile as sf
import sounddevice as sd
import numpy as np
import math
import wave as wv

def lectura_audio(Ficheroaudio):
    '''
    Lee el fichero de audio introducido y retorna los datos, el
sample rate y la información de la cabecera del archivo sea del
tipo que sea.
    En el caso de que sea un archivo no valido saltará una
excepción que nos llevarà a una ventana de alerta.
    '''
    try:
        data, samplerate = sf.read(Ficheroaudio)
        info = sf.info(Ficheroaudio)
        # saca la información de la cabecera con la libreria SF
para después
        # codificar el archivo de salida con la misma información
con la que
        # entra sea cual sea el tipo de archivo de entrada
```

```
    except (TypeError, sf.SoundFileError) as e:
        # mensaje de alerta por introducir un fichero incorrecto
        print("Error al leer el archivo:", str(e))

    return data,samplerate,info

def downsampler(data, samplerate, info, fader):
    '''
    Esta función reduce la calidad del audio introducido como
"data" en función a un porcentaje de calidad "fader".
    El valor máximo de calidad es 100% y el valor mínimo es 1%.
    '''
    if 101 > fader & fader>0:
        factor_degradacion = (fader/100)
    elif fader == 0:
        factor_degradacion = 1
    else:
        raise Exception("Valor fuera de valores permitidos") from
None

    print(factor_degradacion)
    samplerate_nuevo = int(samplerate*factor_degradacion/2)
    factor_downsampling = math.ceil(samplerate/samplerate_nuevo)
    print(factor_downsampling)
    downsampled_data = np.zeros_like(data)
    downsampled_data[::factor_downsampling] =
data[::factor_downsampling]

    return downsampled_data, samplerate, info

def escritura_wave(data, samplerate, info):

    out_file = "Downsampled_out.wav"
    sf.write(out_file, data, samplerate, subtype=info.subtype,
format='WAV')
```

backendIMG.py

```python
from PIL import Image
import math

def downsample_img(image_path, fader):
    # Abrir la imagen
    image = Image.open(image_path)

    # Obtener la calidad actual de la imagen
    current_quality = image.info.get('quality', 100)

    # Calcular la nueva calidad en función del fader
    new_quality = math.ceil((current_quality * (fader / 100))/10)

    # Guardar la imagen con la nueva calidad
    image.save("reduced_image.jpg", quality=new_quality)

    # Cargar la imagen reducida
    downsampled_image = Image.open("reduced_image.jpg")

    # Redimensionar la imagen al tamaño original
    downsampled_image = downsampled_image.resize(image.size)

    # Guardar y devolver la imagen reducida
    downsampled_image.save("downsampled_image.jpg")
    return "downsampled_image.jpg"
```

Además de estos archivos de código, también se adjuntan dentro del archivo .zip las imágenes y audios necesarios.