

1. ANNEXES

1.1. Annex A – Code of the interface

```
#Importing all the libraries needed.
import tkinter as tk
from tkinter import *
from tkinter import ttk

#Defines the type of font and the size of it to a variable
LARGEFONT = ("Futura", 45)

#Creation of the back-end that will contain on a dictionary the other
pages/windows. This will make really simple the creation and
implementation of other the secundary pages
class tkinterApp(tk.Tk): #Inheritance from another Tkinter class

    #Init method that initialize the tkinterApp class
    #Function that gets called everytime the class is called.
    def __init__(self):

        #This will initialize the inherited Tkinter class
        tk.Tk.__init__(self)

        #Establishing a specific size for the window, this is the size
        #of the biggest container
        self.geometry("1024x600")

        #Creates the biggest container.
        container = tk.Frame(self, bg = "pink") #master = self
        #This method puts all the widgets inside the container
        previously declared.
```

```

        container.pack(side = "top", fill = tk.BOTH, expand = True)

        #Empty dictionary that will contain the classes of the pages.
        self.frames = {}

        #Here all the classes of the other windows are saved in the
        dictionary.

        for F in (StartPage, RecordingPage, EQPage, Finish_Recording):

            #The first parameter passed is the container previously
            declared. So which ever class has been called, will have as master =
            container, and that is how it will
            #load all it's widgets in the container passed that is
            positined in the main App.
            #And self = controller, this indicates the secondary class
            that it has acces to the methods of the main one, and that is necessary
            for the Show_frame method.
            frame = F(container, self)

            self.frames[F] = frame
            frame.place(x=0, y=0, width= 1024, height = 600) #This will
            define the size of the windows that are packed and placed in the bigger
            one
            frame.configure( bg = "#D9D0CA")

            self.show_frames(StartPage) #Since this is the initialization of
            the app,
                                #this makes that the start Page the
            one that shows first

        def show_frames(self, Page_name):
            frame = self.frames[Page_name] #It will show the page that is
            being sent by parameter, in this case Page_name is the key from the
            dictionary.
            frame.tkraise() #Raises the Page_name to the front.
            frame.event_generate("<<ShowFrame>>")

#Page 1. Secondary class.
class StartPage(tk.Frame): #Inherits from the Frame class.

    #Initialization of the StartPage class.
    def __init__(self, parent, controller):

        #Initialization of the class inherited. Master = parent. This
        will be assigned when the StarPage is called and the paremetres are
        passed.
        tk.Frame.__init__(self, parent)

        #The use of the ttk library that helps with the layout of the
        app.
        #Creation of Label
        label = ttk.Label(self, text = "Start Page", font =
        LARGEFONT,background = "#D9D0CA", foreground = "#748B8C")

```

```

label.place(x = 365, y = 70)

#Creation of button
style = ttk.Style()
style.configure('Tbutton', font=('Futura', 55), bg='#232323',
foreground='white')
button1 = ttk.Button(self, text = "Start Recording", command =
lambda:controller.show_frames(RecordingPage))
style.theme_use('clam')
button1.place(x = 390, y = 280, width = 200, height = 80)

self.bind("<<ShowFrame>>", self.on_show_frame)

#This will be shown in the terminal, not in the screen.
def on_show_frame(self, event):
    print("Start page is being shown")

#Page 2. Secondary class.
class RecordingPage(tk.Frame): #Inherits from the Frame class.

    #Initialization of the StartPage class.
    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)

        button1 = ttk.Button(self, text = "Back to Start Page", command =
= lambda:controller.show_frames(StartPage))
        button1.place(x = 150, y = 500, width = 180, height = 60)

        button2 = ttk.Button(self, text = "Next", command =
lambda:controller.show_frames(EQPage))
        button2.place(x = 650, y = 500, width = 180, height = 60)

        label = ttk.Label(self, text = "Recording ...", font =
LARGEFONT, background = "#D9D0CA", foreground = "#748B8C")
        label.place(x = 350, y = 70)

        #Gifs work between 15 and 24 frames.
        frameCnt = 24
        #Loading the gif.
        frames =
[PhotoImage(file='C:/Users/paula/Desktop/TFG/Codigo/gif3.gif', format =
'gif -index %i' %(i)) for i in range(frameCnt)]

        #loop for play the gif.
        def update(ind):

            frame = frames[ind]
            ind += 1
            if ind == frameCnt:
                ind = 0
            label2.configure(image=frame)
            self.after(100, update, ind)

```

```

label2 = tk.Label(self,background = "#D9D0CA")
label2.place(x=210, y =160)

self.bind("<<ShowFrame>>", self.on_show_frame)

self.after(0, update, 0)

def on_show_frame(self, event):
    print("Recording page is being shown")

class EQPage (tk.Frame):
    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)

        button1 = ttk.Button(self,text = "Record Again", command =
lambda:controller.show_frames(RecordingPage) )
        button1.place(x = 150, y = 500, width = 180, height = 60)

        button2 = ttk.Button(self, text = "Next", command =
lambda:controller.show_frames(Finish_Recording) # parent.destroy and
button2.place(x = 650, y = 500, width = 180, height = 60)

        label = ttk.Label(self, text = "Equalization in Progress",
justify = CENTER , wraplength = 450, font = LARGEFONT, background =
"#D9D0CA", foreground = "#748B8C")
        label.place(x = 300, y = 200)

class Finish_Recording (tk.Frame):
    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)

        button1 = ttk.Button(self,text = "Record Again", command =
lambda:controller.show_frames(RecordingPage) )
        button1.place(x = 150, y = 500, width = 180, height = 60)

        button2 = ttk.Button(self, text = "Close", command =
controller.destroy) # parent.destroy and
button2.place(x = 650, y = 500, width = 180, height = 60)

        label = ttk.Label(self, text = "The equalization is finished",
justify = CENTER , wraplength = 450, font = LARGEFONT, background =
"#D9D0CA", foreground = "#748B8C")
        label.place(x = 300, y = 170)

app = tkinterApp()
app.mainloop()

```