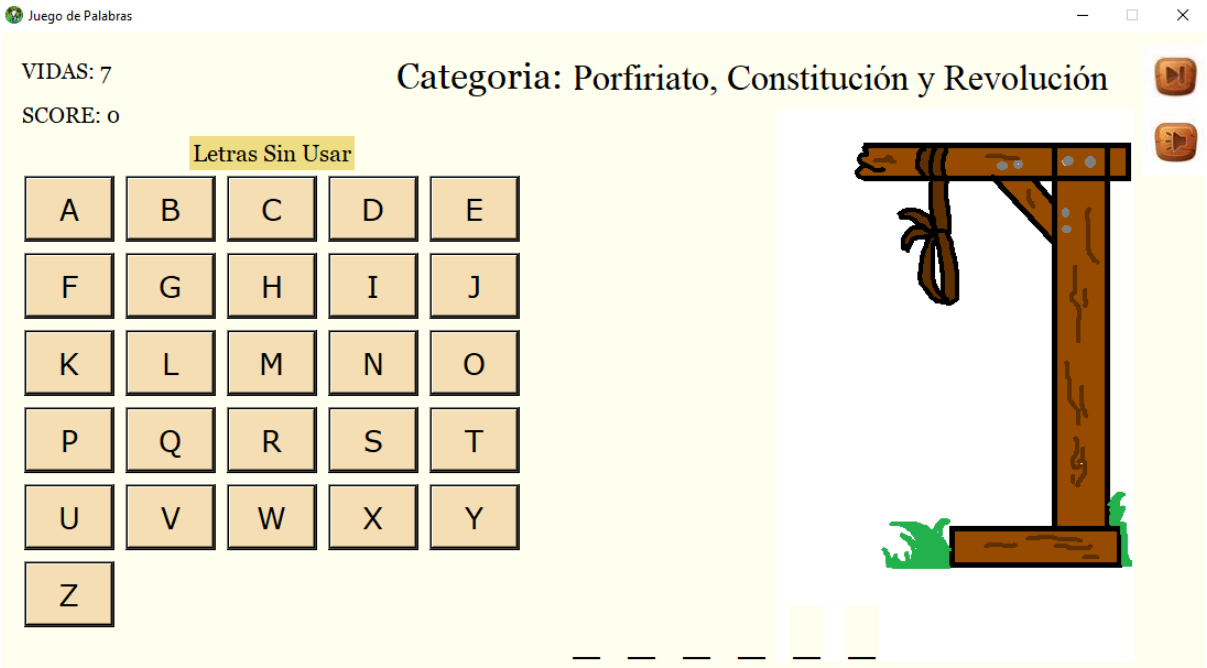
Universidad Autónoma de Baja California
Mexicali, Baja California, México

Facultad de Ingeniería

Reporte de actividad			
Materia:	Lenguaje de programación python		
Tema:	Juego final “Hanes”		
Alumno:	VIllalobos Ensaldo Luis Daniel, Melquicedec Luis Vicente	**Matricula:**	368617 369883
Docente:	Pedro Nunez Yepiz		
Fecha:	07/06/2023		

Pantallas


hanes
HANES
START
EXIT


Juego de Palabras
VIDAS: 7
SCORE: 0
Categoria: Porfiriato, Constitución y Revolución
Letras Sin Usar
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
Z

Juego de Palabras

VIDAS: 7

SCORE: 1

Categoria: Porfiriato, Constitución y Revolución

Letras Sin Usar

A B C D

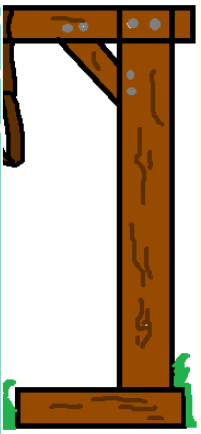
F G H I

K L M N

P Q R S

U V W X

Z



Juego de Palabras

VIDAS: 0

SCORE: 0

Categoria: Porfiriato, Constitución y Revolución

Letras Sin Usar

A B C D

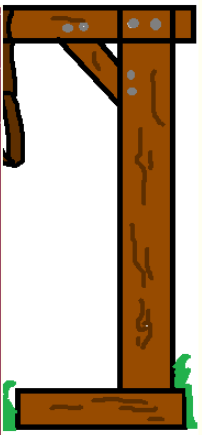
F G H I

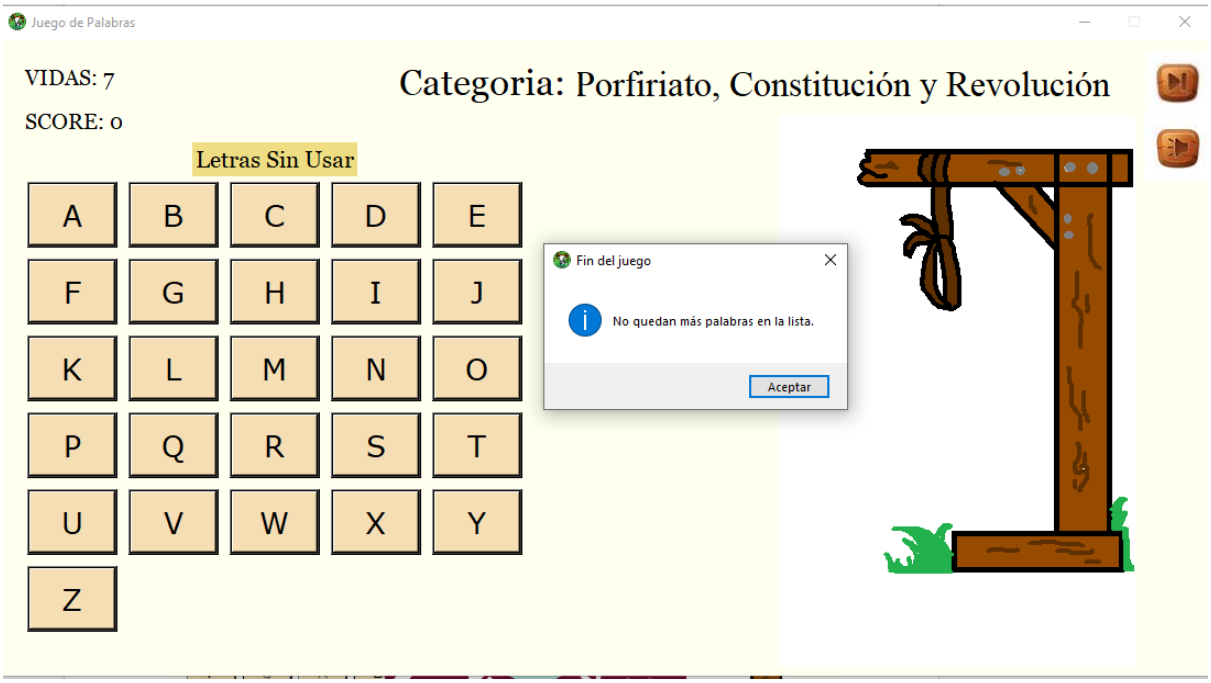
K L M N

P Q R S

U V W X

Z





Código:

```
import tkinter as tk
from tkinter import *
from random import randint
from tkinter.messagebox import showwarning, showinfo
from PIL import ImageTk, Image
from functools import partial
import csv
import pygame
pygame.init()
conjuntoPalabras = []
palabras_adivinadas = 0

def mostrar_juego(ventana_principal):
    pygame.mixer.music.stop()
    ventana_principal.destroy()
    ventana_juego = tk.Tk()
    ventana_juego.title("Juego de Palabras")
    ventana_juego.geometry("1100x580")
    ventana_juego.iconbitmap("icono.ico")
    ventana_juego.configure(bg="ivory")
    ventana_juego.resizable(0, 0)
    letrasUsadas = set()
    vidas = 7
    letrasAcertadas = 0
    imagen_victoria = ""
```

```
    imagen_derrota = ""

    pygame.mixer.music.load("sonido_juego.mp3")
    pygame.mixer.music.play(-1)
    estado_musica_juego = True

    def toggle_musica_juego():
        nonlocal estado_musica_juego
        if estado_musica_juego:
            pygame.mixer.music.pause()
            boton_musica_juego.config(image=imagen_musica_off)
            estado_musica_juego = False
        else:
            pygame.mixer.music.unpause()
            boton_musica_juego.config(image=imagen_musica_on)
            estado_musica_juego = True

    pygame.mixer.pre_init(frequency=44100, size=-16, channels=8,
buffer=44100)


    def cargar_palabras():
        with open("palabras.csv", "r") as archivo_csv:
            lector_csv = csv.reader(archivo_csv)
            for linea in lector_csv:
                palabra = linea[0].lower()
                imagen_victoria = linea[1]
                imagen_derrota = linea[2]
                conjuntoPalabras.append((palabra, imagen_victoria,
imagen_derrota))

    def colocarLetras():
        Label(canvas_letras, text="Letras Sin Usar", font=("Georgia",
16), bg="lightgoldenrod").grid(row=0, column=0, columnspan=5)
        for i in range(26):
            letra = chr(i + 65)
            funcion_parcial = partial(probar_letra, letra)
            frame_letra = Frame(canvas_letras, relief="raised",
borderwidth=2, bg="black")
            frame_letra.grid(row=1 + i // 5, column=i % 5, padx=5,
pady=5)
            letrasBoton[i] = Button(frame_letra, text=letra,
font=("verdana", 20), width=4, command=funcion_parcial, bg="wheat")
```

```
            letrasBoton[i].pack(fill="both", expand=True)
            if letra.lower() in letrasUsadas:
                letrasBoton[i].config(state="disabled")

    def probar_letra(letra):
        nonlocal vidas, letrasAcertadas, imagen_victoria
        global palabras_adivinadas

        if letra.lower() in letrasUsadas:
            showwarning(title="Letra Repetida", message="Ya has
utilizado esta letra.")
            return

        letrasUsadas.add(letra.lower())

        if letra.lower() in palabra:
            for i in range(len(palabra)):
                if palabra[i] == letra.lower():
                    guiones[i].config(text=letra)
                    letrasAcertadas += 1

            if letrasAcertadas == len(palabra):
                mostrar_imagen_victoria(imagen_victoria)
                palabras_adivinadas += 1
                palabras_label.config(text="SCORE:
{}".format(palabras_adivinadas))

        else:
            vidas -= 1
            vidas_label.config(text="VIDAS: {}".format(vidas),
font=("Georgia", 15))

            canvas_juego.itemconfigure(imagenId,
image=imagenes_derrota[vidas - 1])
            if vidas == 0:
                mostrar_imagen_derrota(imagen_derrota)
                palabras_adivinadas = 0
                palabras_label.config(text="SCORE:
{}".format(palabras_adivinadas), font=("Georgia", 15))

        letrasBoton[ord(letra.lower()) - 97].config(state="disabled")

        for i in range(26):
```

```
            if chr(i + 65).lower() in letrasUsadas:
                letrasBoton[i].config(state="disabled")

    def mostrar_imagen_victoria(imagen_palabra):
        imagen_victoria = Image.open(imagen_palabra)
        imagen_victoria = imagen_victoria.resize((500, 500),
Image.ANTIALIAS)
        imagen_victoria = ImageTk.PhotoImage(imagen_victoria)
        imagen_label.config(image=imagen_victoria)
        imagen_label.image = imagen_victoria
        sonido_victoria = pygame.mixer.Sound("sonido_victoria.mp3")
        sonido_victoria_channel = pygame.mixer.Channel(1)
        sonido_victoria_channel.play(sonido_victoria)


    def mostrar_imagen_derrota(imagen_palabra):
        imagen_derrota = Image.open(imagen_palabra)
        imagen_derrota = imagen_derrota.resize((500, 500),
Image.ANTIALIAS)
        imagen_derrota = ImageTk.PhotoImage(imagen_derrota)
        imagen_label.config(image=imagen_derrota)
        imagen_label.image = imagen_derrota
        sonido_derrota = pygame.mixer.Sound("sonido_derrota.mp3")
        sonido_derrota_channel = pygame.mixer.Channel(2)
        sonido_derrota_channel.play(sonido_derrota)

    def siguiente_palabra():
        nonlocal palabra, vidas, letrasUsadas, letrasAcertadas,
guiones, imagen_victoria, imagen_derrota, imagenId
        letrasUsadas.clear()
        vidas = 7
        letrasAcertadas = 0

        limpiar_pantalla()
        limpiar_imagen()

        if len(conjuntoPalabras) > 0:
            index_palabra = randint(0, len(conjuntoPalabras) - 1)
            palabra_imagen = conjuntoPalabras.pop(index_palabra)
            if isinstance(palabra_imagen, tuple) and
len(palabra_imagen) == 3:
                palabra, imagen_victoria, imagen_derrota =
palabra_imagen
```

```
                vidas_label.config(text="VIDAS: {}".format(vidas),
font=("Georgia", 15))
                palabras_label.config(text="SCORE:
{}".format(palabras_adivinadas), font=("Georgia", 15))

                # Reiniciar los guiones para la nueva palabra
                inicialX = 0
                for i in range(len(palabra)):
                    guion = Label(canvas_juego, text="_",
font=("verdana", 30), bg="ivory")
                    guion.place(x=inicialX, y=500)
                    guiones.append(guion)
                    inicialX += 50

                # Reiniciar las imágenes del ahorcado
                canvas_juego.delete(imagenId)
                imagenId = canvas_juego.create_image(350, 300,
image=imagenes_derrota[6])

                for boton in letrasBoton:
                    boton.config(state="normal")

        else:
            showinfo(title="Fin del juego", message="No quedan más
palabras en la lista.")

    def limpiar_imagen():
        nonlocal imagen_victoria
        imagen_label.config(image="")
        imagen_victoria = None

    def limpiar_pantalla():
        for guion in guiones:
            guion.destroy()
        guiones.clear()

        imagen_label.config(image="")


    cargar_palabras()
    palabra, imagen_victoria, imagen_derrota =
conjuntoPalabras.pop(randint(0, len(conjuntoPalabras) - 1))
```

```
    canvas_juego = tk.Canvas(ventana_juego, width=700, height=600,
bg="ivory", borderwidth=0, highlightthickness=0)
    canvas_juego.grid(row=0, column=1, padx=20, pady=20)

    canvas_letras = tk.Canvas(ventana_juego, width=300, height=600,
bg="ivory", borderwidth=0, highlightthickness=0)
    canvas_letras.grid(row=0, column=0, padx=20, pady=20)

    imagenes_derrota = [
        ImageTk.PhotoImage(file="1.png"),
        ImageTk.PhotoImage(file="2.png"),
        ImageTk.PhotoImage(file="3.png"),
        ImageTk.PhotoImage(file="4.png"),
        ImageTk.PhotoImage(file="5.png"),
        ImageTk.PhotoImage(file="6.png"),
        ImageTk.PhotoImage(file="7.png"),
    ]

    imagenId = canvas_juego.create_image(350, 300,
image=imagenes_derrota[6])

    letrasBoton = [None] * 26
    colocarLetras()

    guiones = []
    inicialX = 0
    for i in range(len(palabra)):
        guion = Label(canvas_juego, text="_", font=("verdana", 30),
bg="ivory")
        guion.place(x=inicialX, y=500)
        guiones.append(guion)
        inicialX += 50

    canvas_juego.config(bg="ivory")

    imagen_siguiente = ImageTk.PhotoImage(file="boton_siguiente.jpg")

    boton_siguiente = tk.Button(ventana_juego, image=imagen_siguiente,
relief="flat", borderwidth=5, command=siguiente_palabra)
    boton_siguiente.place(x=1040, y=10, width=60, height=60)
    boton_siguiente.config(bg="ivory")
```

```
    vidas_label = tk.Label(ventana_juego, text="VIDAS:
{}".format(vidas), font=("Georgia", 15), bg="ivory")
    vidas_label.place(x=20, y=20)

    palabras_label = tk.Label(ventana_juego, text="SCORE:
{}".format(palabras_adivinadas), font=("Georgia", 15), bg="ivory")
    palabras_label.place(x=20, y=60)

    catego = tk.Label(ventana_juego, text="Categoria:
".format(palabras_adivinadas), font=("Georgia", 25), bg="ivory")
    catego.place(x=360, y=17)

    frame_titulo = Frame(canvas_juego, bg="ivory")
    frame_titulo.place(x=0, y=0)
    titulo_juego = Label(frame_titulo, text="Porfiriato, Constitución y
Revolución", font=("Times", 25), bg="ivory")
    titulo_juego.pack()

    imagen_musica_on = ImageTk.PhotoImage(file="musica_on.jpg")
    imagen_musica_off = ImageTk.PhotoImage(file="musica_off.jpg")

    boton_musica_juego = tk.Button(ventana_juego,
image=imagen_musica_on, relief="flat", borderwidth=5,
command=toggle_musica_juego)
    boton_musica_juego.place(x=1040, y=70, width=60, height=60)

    imagen_label = tk.Label(ventana_juego, bg="ivory")
    imagen_label.place(x=350, y=100)

    ventana_juego.deiconify()
    ventana_juego.mainloop()



def mostrar_ventana_principal():
    ventana_principal = tk.Tk()
    ventana_principal.title("hanes")
    ventana_principal.geometry("800x384")
    ventana_principal.iconbitmap("icono.ico")
    ventana_principal.resizable(0, 0)
    ventana_principal.configure(bg="white", highlightthickness=0)
```

```
    etiqueta_titulo = Label(ventana_principal, text="HANES",
font=("Times", 40, "bold"), bg="white", highlightthickness=0)
    etiqueta_titulo.pack(pady=10)

    imagen_start = ImageTk.PhotoImage(file="start.png")
    imagen_salir = ImageTk.PhotoImage(file="salir.png")
    fondo = ImageTk.PhotoImage(file="portada.jpg")
    imagen_musica_on = ImageTk.PhotoImage(file="musica_on.jpg")
    imagen_musica_off = ImageTk.PhotoImage(file="musica_off.jpg")

    etiqueta_fondo = tk.Label(ventana_principal, image=fondo)
    etiqueta_fondo.pack()

    pygame.mixer.music.load("sonido_principal.mp3")
    pygame.mixer.music.play(-1)

    boton_iniciar = tk.Button(ventana_principal, image=imagen_start,
bg="ivory", relief="flat", borderwidth=5, command=lambda:
mostrar_juego(ventana_principal))
    boton_iniciar.place(x=300, y=80, width=200, height=65)

    boton_salir = tk.Button(ventana_principal, image=imagen_salir,
bg="ivory", relief="flat", borderwidth=5, command=quit)
    boton_salir.place(x=300, y=145, width=200, height=65)



    estado_musica = True

    def toggle_musica():
        nonlocal estado_musica
        if estado_musica:
            pygame.mixer.music.pause()
            boton_musica.config(image=imagen_musica_off)
            estado_musica = False
        else:
            pygame.mixer.music.unpause()
            boton_musica.config(image=imagen_musica_on)
            estado_musica = True

    boton_musica = tk.Button(ventana_principal, image=imagen_musica_on,
relief="flat", borderwidth=5, command=toggle_musica)
    boton_musica.place(x=740, y=0, width=60, height=60)
```

```
    ventana_principal.mainloop()


mostrar_ventana_principal()
```