



JUEGO DEL AHORCADO

Por Iván García



PUNTOS DE LA PRESENTACIÓN

- 1 Imports con los que trabajaré
- 2 Funciones del juego
- 3 Interfaz
- 4 Base de datos
- 5 Entidad Relación
- 6 Modelo relacional





IMPORTS

```
1 import random  
2 import sqlite3  
3 import tkinter as tk  
4 from tkinter import messagebox
```

- Random para la generación aleatoria de la palabra que tendremos que adivinar
- He usado sqlite3 para hacer la conexión con la base de datos.
- Tkinter para trabajar con la interfaz
- Messagebox mostrará algunos mensajes al ganar o perder la partida o para informarte de algún error.

Aquí he hecho la conexión con la base de datos usando un cursor el sqlite3

```
def crear_base_datos(): 1 usage  ± Ivan
    conn = sqlite3.connect("ahorcado.db")
    cursor = conn.cursor()
        Ctrl+F to Chat, Ctrl+I to Command
    # Crear tabla palabras
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS palabras (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            tema TEXT NOT NULL,
            palabra TEXT NOT NULL
        )
    """)

    # Crear tabla partidas
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS partidas (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nombre_jugador TEXT NOT NULL,
            partidas_jugadas INTEGER DEFAULT 0,
            partidas_ganadas INTEGER DEFAULT 0
        )
    """)

    conn.commit()
    conn.close()
```

He hecho un diccionario de palabras divididas por tema y después las he añadido a la base de datos en la tabla “palabras” que cree anteriormente.

```
conn = sqlite3.connect("ahorcado.db")
cursor = conn.cursor()
# insertar en bd
for tema, lista_palabras in palabras.items():
    for palabra in lista_palabras:
        cursor.execute( sql: """
            INSERT OR IGNORE INTO palabras (tema, palabra)
            VALUES (?, ?)
        """, parameters: (tema, palabra))

conn.commit()
conn.close()
```

```
def insertar_palabras_iniciales(): 1 usage  ~ Ivan *
palabras = {
    "frutas": [
        "manzana",
        "platano",
        "naranja",
        "uva",
        "fresa",
        "melocoton",
        "pera",
        "mango",
        "kiwi",
        "sandia",
        "melon",
        "cereza",
        "ciruela",
        "higo",
        "granada",
        "papaya",
        "guayaba",
        "piña",
        "limon",
        "lima",
        "frambuesa",
        "arándano",
        "durazno",
        "nectarina",
        "mandarina",
        "coco",
        "pomelo",
        "tamarindo",
        "chirimoya",
        "caqui",
        "lichi"
    ],
    "informatica": [
        "python", "java", "código", "teclado", "monitor", "inteligencia", "servidor", "cliente", "software",
        "internet"
    ],
    "personas": [
        "maaaaaaaaaartin",
        "ivan",
        "andrea",
        "rafa",
        "antonio",
        "jusep",
        "niko",
        "rodri",
        "alberto",
        "victor",
        "lucas",
        "miguel",
        "patricia",
        "josema",
        "raul",
        "pablo",
        "dani",
    ]
},
```

Aquí hago una llamada a la base de datos haciendo un select de un tema que elegiremos posteriormente durante el juego.

Esto te dará un return con la palabra seleccionada

```
def cargar_palabras(tema):    1 usage  ± Ivan *
    conn = sqlite3.connect("ahorcado.db")
    cursor = conn.cursor()

    cursor.execute( sql: "SELECT palabra FROM palabras WHERE tema = ?",
                    parameters: (tema,))
    palabras = [row[0] for row in cursor.fetchall()]

    conn.close()
    return palabras
```

En este caso se llamará a la tabla “partidas” donde mediante un if se hará un update de los datos de un jugador existente o un insert si es la primera partida de ese jugador.

El contador de partidas aumentará siempre en 1, el de victorias solo si el jugador ganó.

```
def guardar_partida(nombre_jugador, gano): 2 usages ± Ivan +1
    conn = sqlite3.connect("ahorcado.db")
    cursor = conn.cursor()

    # Consultar si el jugador ya existe
    cursor.execute(sql: """
        SELECT id, partidas_jugadas, partidas_ganadas
        FROM partidas
        WHERE nombre_jugador = ?
    """, parameters: (nombre_jugador,))
    resultado = cursor.fetchone()

    if resultado:
        # El jugador ya existe, actualizamos sus estadísticas
        jugador_id, partidas_jugadas, partidas_ganadas = resultado
        partidas_jugadas += 1 # Aumenta el contador de partidas jugadas
        if gano:
            partidas_ganadas += 1 # Si ganas aumentan las partidas ganadas
        cursor.execute(sql: """
            UPDATE partidas
            SET partidas_jugadas = ?, partidas_ganadas = ?
            WHERE id = ?
        """, parameters: (partidas_jugadas, partidas_ganadas, jugador_id))
    else:
        # El jugador no existe, lo insertamos en la base de datos
        cursor.execute(sql: """
            INSERT INTO partidas (nombre_jugador, partidas_jugadas, partidas_ganadas)
            VALUES (?, ?, ?)
        """, parameters: (nombre_jugador, 1, 1 if gano else 0))

    conn.commit()
    conn.close()
```

CONSTRUCCIÓN DE LA CLASE AhorcadoApp

```
class AhorcadoApp: 1 usage ± 97ivangarcia +1 *  
Codeium: Refactor | Explain | Docstring | X  
def __init__(self, root): ± 97ivangarcia *  
    self.root = root #esto es la ventana principal  
    self.root.title("Juego del ahorcado")  
    self.nombre_jugador = ""  
    self.tema = ""  
    self.palabra_secreta = ""  
    self.letras_adivinadas = [] #array con las letras adivinadas  
    self.letras_incorrectas = [] #array con las letras incorrectas  
    self.error = 0  
    self.intentos_restantes = 6 #maximo de intentos  
  
    self.estilo_inicio()
```

CONFIGURACIÓN DE COLORES Y ESTILOS

```
fondo_principal = "#1e1e2f"
fondo_secundario = "#2d2d44"
color_texto = "#ffffff"
color_resaltado = "#ff79c6"
fuente_titulo = ("Helvetica", 20, "bold")
fuente_texto = ("Helvetica", 12)
```

APLICANDO COLORES Y ESTILOS (ventana de inicio)

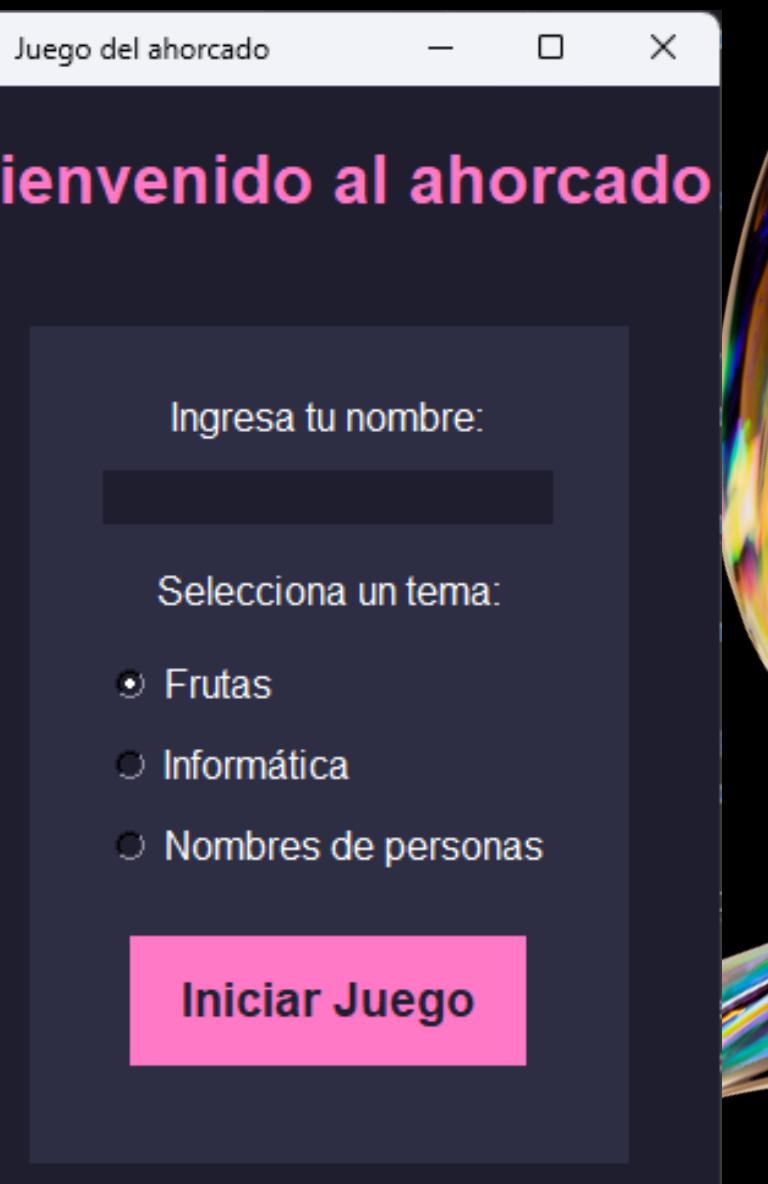
```
self.root.configure(bg=fondo_principal)

# Titulo
tk.Label(self.root, text="Bienvenido al ahorcado", font=fuente_titulo, bg=fondo_principal,
         fg=color_resaltado).pack(pady=20)

# datos jugador
frame = tk.Frame(self.root, bg=fondo_secundario, padx=20, pady=20, relief="flat")
frame.pack(pady=20)
# ingresar nombre
tk.Label(frame, text="Ingresa tu nombre:", font=fuente_texto, bg=fondo_secundario,
         fg=color_texto).pack(pady=5)
self.nombre_entry = tk.Entry(frame, font=fuente_texto, relief="flat", justify="center",
                             bg=fondo_principal, fg=color_texto, insertbackground=color_resaltado)
self.nombre_entry.pack(pady=5)

# elegir el tema
tk.Label(frame, text="Selecciona un tema:", font=fuente_texto, bg=fondo_secundario,
         fg=color_texto).pack(pady=10)
self.tema_var = tk.StringVar(value="frutas")
temas = [("Frutas", "frutas"), ("Informática", "informatica"), ("Nombres de personas", "personas")]
for texto, valor in temas:
    tk.Radiobutton(frame, text=texto, variable=self.tema_var, value=valor, font=fuente_texto,
                   bg=fondo_secundario, fg=color_texto, selectcolor=fondo_principal,
                   activebackground=fondo_secundario,
                   activeforeground=color_resaltado).pack(anchor="w", padx=10, pady=2)

# boton inicio
tk.Button(frame, text="Iniciar Juego", command=self.iniciar_juego, font=("Helvetica", 14, "bold"),
          bg=color_resaltado, fg=fondo_principal, bd=0, padx=15, pady=10, activebackground="#ff92d0",
          activeforeground=fondo_principal).pack(pady=20)
```



Alertas

```
def iniciar_juego(self): 1usage ± 97ivangarcia +1
    self.nombre_jugador = self.nombre_entry.get().strip()
    self.tema = self.tema_var.get()
    if not self.nombre_jugador:
        messagebox.showerror(title= "Error", message= "Por favor, ingrese su nombre.")
        return

    palabras = cargar_palabras(self.tema)
    if not palabras:
        messagebox.showerror(title= "Error", message= "No hay palabras disponibles para este tema.")
        return

    self.palabra_secreta = random.choice(palabras).lower()
    self.letras_adivinadas = []
    self.letras_incorrectas = []
    self.errorres = 0

    self.estilo_juego()
```

Aquí he hecho uso del messagebox que mostrará una alerta si se cumple una condición.

Mas colores y estilos (ventana de juego)

```
def estilo_juego(self): 2 usages ± 97ivangarcia +1 *
    for widget in self.root.winfo_children():
        widget.destroy()

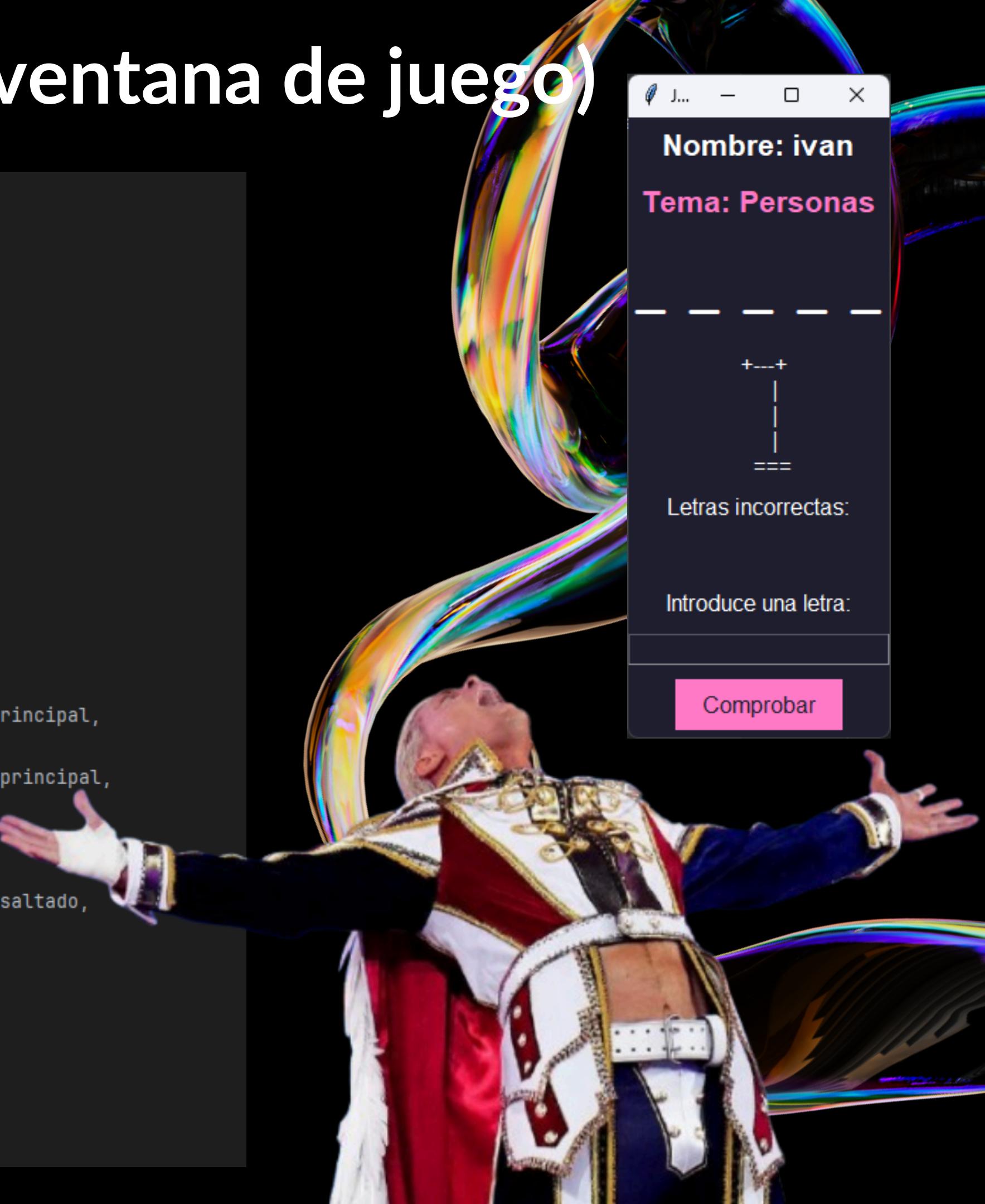
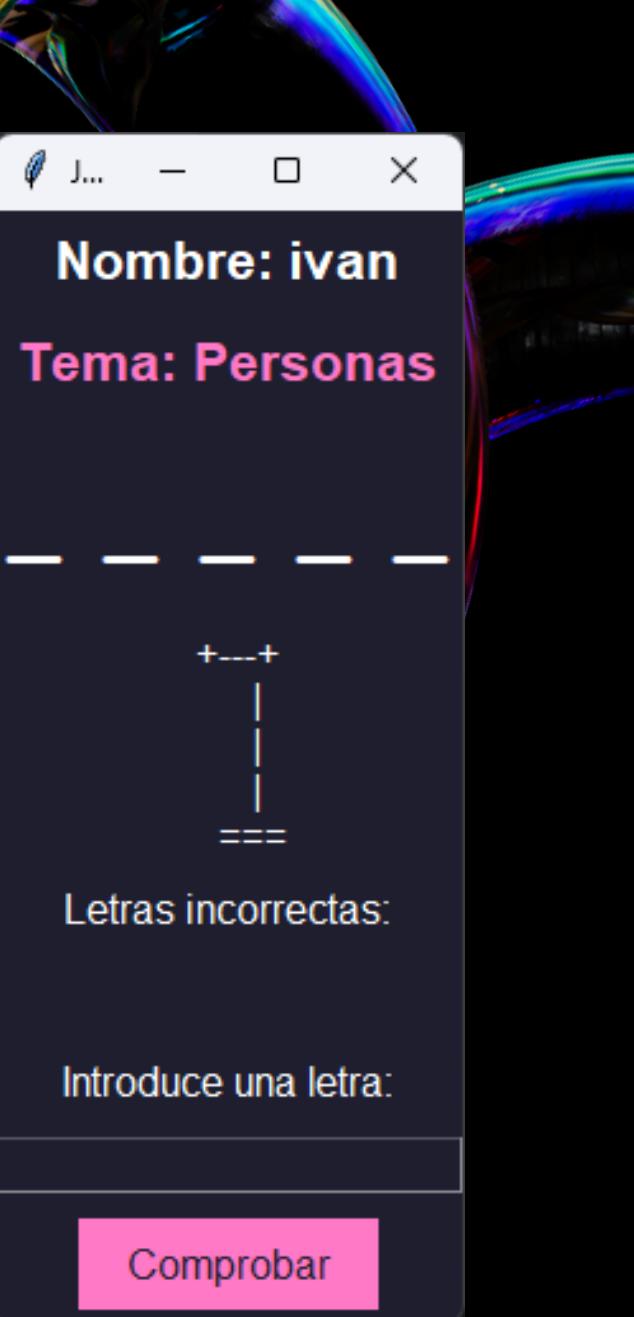
    # configuracion colores y estilo
    fondo_principal = "#1e1e2f"
    fondo_secundario = "#2d2d44"
    color_texto = "#ffffff"
    color_resaltado = "#ff79c6"
    fuente_titulo = ("Helvetica", 16, "bold")
    fuente_texto = ("Helvetica", 12)
    fuente_resaltado = ("Courier New", 24, "bold")
    borde_redondeado = 10

    self.root.configure(bg=fondo_principal)

    # datos jugador y tema
    tk.Label(self.root, text=f"Nombre: {self.nombre_jugador}", font=fuente_titulo, bg=fondo_principal,
             fg=color_texto).pack(pady=5)
    tk.Label(self.root, text=f"Tema: {self.tema.capitalize()}", font=fuente_titulo, bg=fondo_principal,
             fg=color_resaltado).pack(pady=5)

    # palabra secreta
    self.palabra_label = tk.Label(self.root, text=self.get_palabra_mostrada(), font=fuente_resaltado,
                                  bg=fondo_principal, fg=color_texto)
    self.palabra_label.pack(pady=20)

    # dibujo
    self.dibujo_label = tk.Label(self.root, text=self.dibujar_ahorcado(), font=fuente_texto,
                                 bg=fondo_principal, fg=color_texto)
    self.dibujo_label.pack()
```

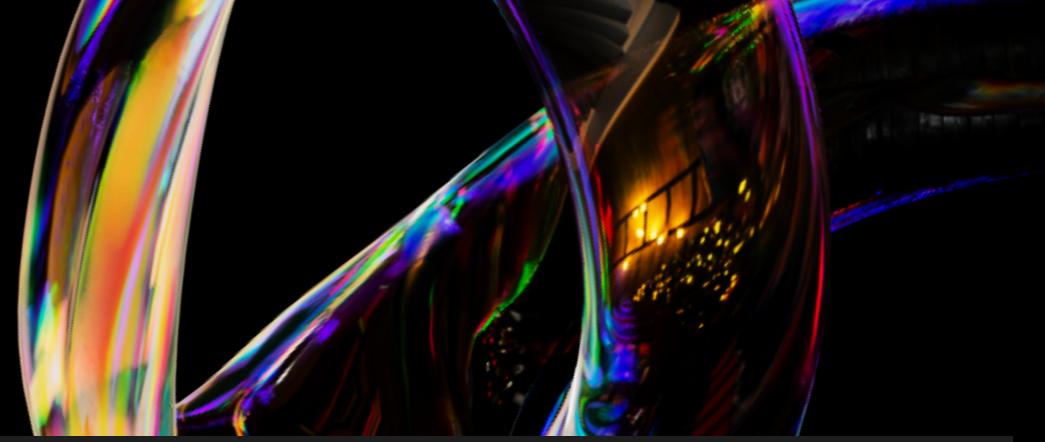


Dibujo ahorcado

```
def dibujar_ahorcado(self): 1 usage ± 97ivangarcia +1
    ahorcado = [
        "  +---+\n        |\n        |\n        |\n        ===",
        "  +---+\n        0   |\n        |\n        |\n        ===",
        "  +---+\n        0   |\n        |   |\n        |\n        ===",
        "  +---+\n        0   |\n        |   |\n        |\n        /|\n        ===",
        "  +---+\n        0   |\n        |   |\n        |\n        /|\n        /\n        ===",
        "  +---+\n        0   |\n        |   |\n        |\n        /|\n        /\n        /\n        ===",
        "  +---+\n        0   |\n        |   |\n        |\n        /|\n        /\n        /\n        /\n        ===",
    ]
    return ahorcado[self.erros]
```

Esto es el dibujo del ahorcado, cada fallo en el juego imprimirá una nueva linea

Comprobación de letras



```
def comprobar_letra(self): 1 usage ▾ 97ivangarcia +1 *
    letra = self.letra_entry.get().strip().lower()
    if len(letra) != 1 or not letra.isalpha():
        messagebox.showerror(title= "Error", message= "Por favor, ingresa solo una letra.")
        return

    if letra in self.letras_adivinadas or letra in self.letras_incorrectas:
        messagebox.showinfo(title= "Atención", message= "Ya has adivinado esta letra.")
        return

    if letra in self.palabra_secreta:
        self.letras_adivinadas.append(letra)
        if all(letra in self.letras_adivinadas for letra in self.palabra_secreta):
            messagebox.showinfo(title= "¡OLEEEE OLEEEE LOS CARACOLEEEEEE!",
                                message= f"; Felicidades shurra, ganaste! La palabra era {self.palabra_secreta}.")
            guardar_partida(self.nombre_jugador, gano= True)
            self.estilo_inicio()

    else:
        self.letras_incorrectas.append(letra)
        self.errores += 1
        if self.errores == 6:
            messagebox.showinfo(title= "Perdiste", message= f"Perdiste. La palabra era {self.palabra_secreta}.")
            guardar_partida(self.nombre_jugador, gano= False)
            self.estilo_inicio()

    self.estilo_juego()
```

Otra vez mediante messagebox compruebo si hay errores como letras duplicadas o si el jugador ha ingresado mas de una.

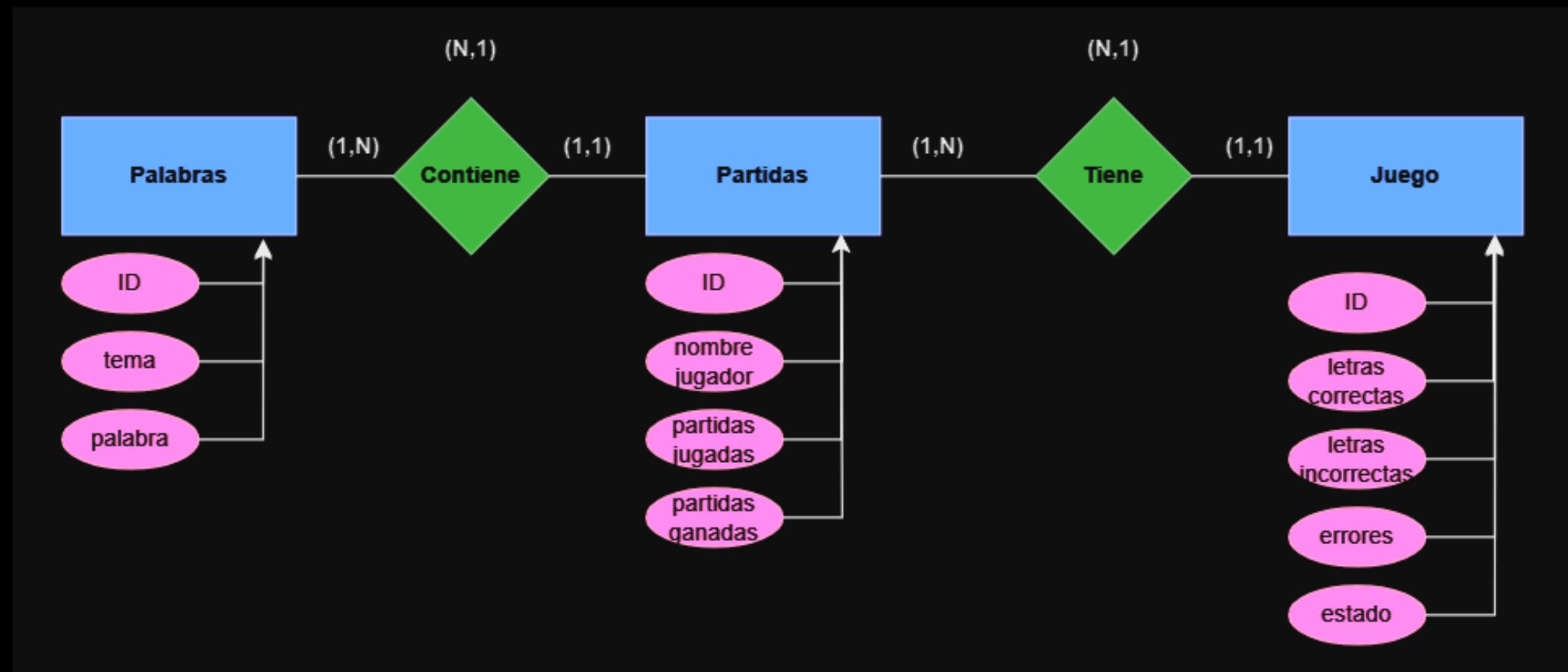
También te notificara al ganar o perder la partida mostrandote cual era la palabra secreta.

Ejemplo base de datos

Table: partidas Page: 0

id	nombr...	partida...	partida...
1	sas	1	1
2	ivan	4	3
3	pepe	1	1
4	manolito	1	1

ENTIDAD RELACIÓN



Contiene: Una palabra puede estar en muchas partidas, pero cada partida contiene muchas palabras (relación (1,N))

Tiene: Un juego puede tener muchas partidas, pero cada partida pertenece a un único juego (relación (1,N))

MEET THE TEAM

Thank you for your time! Reach out to us for questions.



IVÁN
GARCÍA

Chief Executive Officer



IVÁN
GARCÍA

Marketing



IVÁN
GARCÍA

App developer



IVÁN
GARCÍA

Director





Ilerna City, 2024

GRACIAS

por su atención

Presentación hecha por Iván García

