Presentación del Proyecto Final

Seminario de Lenguajes - Python | 2022

Presentadores:

Aberastain Marzorati, Álvaro 19950/8
Beraún Centineo, Julián Jorge 19962/3

Temas a tratar

- Módulo de manejo de datos
- Pantalla de juego
- Funcionamiento de las dificultades

▶ ¿Por qué estos temas?

Temas a tratar

- Módulo de manejo de datos
  - El “corazón” de la comunicación entre módulos
  - Contiene todas las funciones para el tratado de archivos
- Pantalla de juego
  - La pantalla más dinámica para el usuario
  - Varios mecanismos para las distintas funcionalidades
    - Opciones, características, temporizador, pasar ronda, muestra de puntaje...
- Funcionamiento de las dificultades
  - Se presenta un conflicto entre dos aspectos
    - Dificultades por defecto vs Configuración
  - Distintas soluciones al problema

Módulo para manejar datos

- La mayoría de los métodos definidos en este módulo respetan la estructura mostrada tanto para obtener como para guardar la información.

- Funcionan de tal forma que, si el archivo no existe o se encuentra vacío, lo crea/abre en modo escritura.

Archivos ↔ manejar_datos ↔ Otros módulos

Fragmentos del módulo manejar_datos.py

```
def obtener_perfiles():
    try:
        with open("perfiles.json", "r") as perfiles:
            jugadores = json.load(perfiles)
    except:
        with open("perfiles.json", "w") as perfiles:
            jugadores = []
    return jugadores

def guardar_perfiles(jugadores):
    with open("perfiles.json", "w") as perfiles:
        json.dump(jugadores, perfiles)
```

El fragmento de código fue simplificado para la presentación

Pantalla de juego

- Los módulos se comunican principalmente para compartir información.
    - Esto sucede varias veces en el módulo menu_juego.py

Fragmentos del módulo menu_juego.py

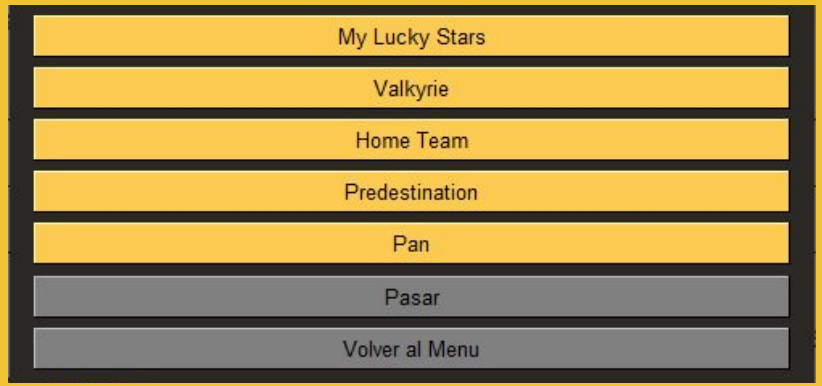
```
from . import manejar_datos as md

def iniciar_pantalla_juego():
    config = md.obtener_config()
    dataset = md.obtener_dataset(config["dataset"])
    . . .
def pasar_ronda(...):
    if ronda_actual == cant_rondas:
        generar_evento(...)
        id = md.obtener_id_ultima_partida()
        md.guardar_partidas(eventos, id)
        if (estado == "finalizado"):
            md.guardar_puntajes(cant_puntos,...)
    . . .
```

El fragmento de código fue simplificado para la presentación

Pantalla de juego

- Recibe el dataset (lista de listas).
- Selecciona 5 elementos al azar.
- Toma la primera como correcta y el resto como incorrecta.
- Las mezcla y descarta de la variable “dataset”.



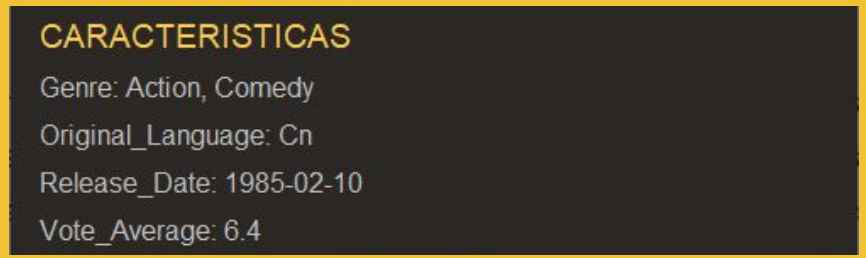
Función generar_opciones()

```
def generar_opciones(dataset):
    lineas = choices(dataset, k=5)
    linea_correcta = lineas[0]
    opcion_correcta = [sg.Button(linea_correcta,
    key="-OPCION CORRECTA-")]
    dataset.remove(linea_correcta)
    opciones = [opcion_correcta]
    for i in range(1,5):
        boton_incorrecto =
[sg.Button(lineas[i][5],
        key="-OPCION INCORRECTA-")]
        opciones.append(boton_incorrecto)
    shuffle(opciones)
    return opciones, linea_correcta
```

El fragmento de código fue simplificado para la presentación

Pantalla de juego

- Recibe la configuración, el encabezado del dataset, y la línea correcta.
- Itera sobre las características del elemento correcto en base a la cantidad que dice la configuración.
- Las va almacenando en un arreglo de textos y lo retorna.



Función obtener_características()

```
def obtener_caracteristicas(config,
    encabezado,linea_correcta):
    caracteristicas = [[sg.Text('CARACTERISTICAS')]]
    for i in range(config["cant_carac"]):
        nueva_carac = [sg.Text(encabezado[i]+
        linea_correcta[i])]
        caracteristicas.append(nueva_carac)

    return caracteristicas
```

El fragmento de código fue simplificado para la presentación

Dificultades vs. Configuración

- Hay dificultades por defecto, con los ajustes definidos por el grupo.
- El menú de configuración se dispone para que el jugador modifique las características.
  ►¿Qué pasa si el usuario elige una dificultad y luego cambia alguna configuración?
  - Se mantiene la dificultad seleccionada?
    - Inconsistencia en la tabla de puntajes
    - No tendrían sentido las estadísticas

Nuestra solución → Dificultad “personalizada”

Funcionamiento de las dificultades

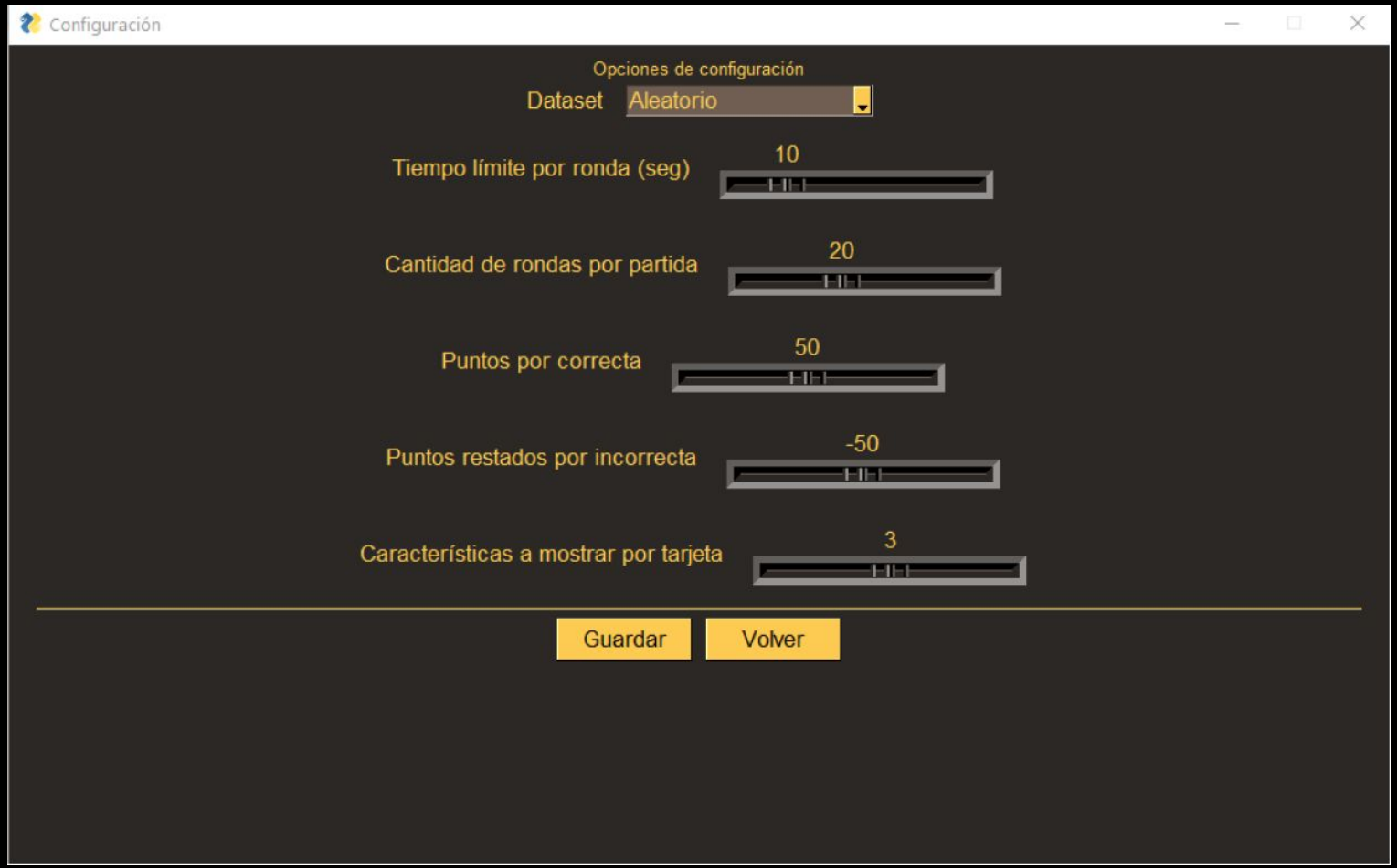
- Verifica constantemente si se modificaron los controles deslizadores para activar una flag.
- Si se guardan los cambios, y la flag fue activada, se guarda la dificultad como “Personalizada”.
- No se activa la flag si sólo se cambia el dataset

Fragmento de menu_config.py

```
flag_sliders = False
while True:
      sliders = ["-CONFIG_TIEMPO-”, “-CONFIG_...”]
      if (...)
      elif event in sliders:
        flag_sliders = True

      elif event == "-CONFIG_GUARDAR-":
            . . .
            if (flag_sliders):
                config["dificultad"] = "Personalizada"
            md.guardar_config(config)
       . . .
```

El fragmento de código fue simplificado para la presentación


Configuración
Opciones de configuración
Dataset
Aleatorio
Tiempo límite por ronda (seg)
10
Cantidad de rondas por partida
20
Puntos por correcta
50
Puntos restados por incorrecta
-50
Características a mostrar por tarjeta
3
Guardar
Volver

Gracias por su atención

Nos vemos en Proyecto de Software.