

Seminario Python - Presentacion Final

Simón Palacios - Matías Daglio

Evelyn Naveyra - Matías Hernan Pereyra

Contacto: palaciossimon6@gmail.com | matias.a.daglio@gmail.com

Agosto 2022

Índice

1	Introducción	2
2	Temas estudiados	2
2.1	Python	2
2.1.1	Entornos Virtuales	3
2.2	Biblioteca	3
2.2.1	Pandas	3
2.2.2	MatPlot	3
2.2.3	PySimpleGUI	4
3	Problemas y soluciones surgidas durante el desarrollo	4
4	Consideraciones éticas sobre el desarrollo	6
4.1	Licencias	6
4.1.1	¿Que es una licencia de software?	6
4.1.2	Tipos de licencias principales	6
4.1.3	Diferencias entre software libre y comercial	7
5	Conclusiones y trabajos futuros	7
5.1	Código repetitivo	7
6	Referencias	7
7	Anexo 1 guía de usuario	8
7.1	Menú principal	8
7.2	Pantalla de juego	9
7.3	Pantalla de perfiles	11
7.3.1	Pantalla de creación de perfiles	12
7.3.2	Pantalla de edición de perfiles	12
7.4	Pantalla de configuración	13
7.5	Pantalla de puntuación	13
8	Anexo 2 guía para el desarrollador	14
8.1	Ejecución de Loops	14
8.2	Los casos	15
8.3	Accesos a archivos	20

1. Introducción

Este trabajo presenta una aproximación a los conocimientos adquiridos durante el seminario de Python, con el objetivo de obtener las competencias genéricas tecnológicas¹ y su posterior implementación en el desarrollo del juego Figurace.

Esta presentación abarca un marco teórico compuesto por las librerías utilizadas, un enfoque funcional sobre el desglose del problema en subpartes y una capacidad comunicacional entre objetos instanciados de clases representando las interfaces del juego. Para la implementación se hizo uso de librerías como PYSimpleGUI, matplotlib, pandas, jupyter notebook y otras funcionalidades incluidas en el lenguaje.

También se aborda, en la presentación, consideraciones éticas del uso de licencias, los problemas surgidos durante el desarrollo y una guía tanto para usuarios como para desarrolladores.

2. Temas estudiados

2.1. Python

"Python es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave. Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science[...]."²

Python es lenguaje multiparadigma que permite la programación orientada a objetos, funcional e imperativa. Ofrece, por un lado, una multiplicidad de funcionalidades incorporadas para el procesamiento de datos como map, reduce, filter, decoradores, generadores y demás. Como también las posibilidades que brinda la programación orientada a objetos, clases, herencia, property, MRO, etc. Un acercamiento a estos métodos, funciones, conocimientos está en las referencias.³

2.1.1. Entornos Virtuales

”Un entorno virtual Python es un espacio de trabajo creado a partir de una instalación de Python existente en un sistema; al que es posible agregar o quitar paquetes y módulos sin riesgo de dañar la instalación principal. Utilizar este recurso es muy apropiado cuando se están desarrollando varios proyectos al mismo tiempo que utilizan módulos distintos o versiones diferentes de un mismo módulo.”⁴

2.2. Biblioteca

Algunas de las librerías que se usaron en el proyecto.

2.2.1. Pandas

”Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos. Las principales características de esta librería son:

- Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Realiza todas estas operaciones de manera muy eficiente.”⁵

2.2.2. MatPlot

Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones. Permite crear y personalizar los tipos de gráficos más comunes, entre ellos:

- Diagramas de barras
- Histograma

- Diagramas de sectores
- Diagramas de caja y bigotes
- Diagramas de violín
- Diagramas de dispersión o puntos
- Diagramas de líneas
- Diagramas de áreas
- Diagramas de contorno
- Mapas de color y combinaciones de todos ellos.”⁶

2.2.3. PySimpleGUI

PySimpleGUI es una librería para el desarrollo interfaces gráficas mediante el lenguaje python. Ésta está basada en las prestaciones que ofrecen las librerías tkinter, remi, Qt y WxPython.

PySimpleGUI ofrece la posibilidad de simplificar el código mediante estructuras prearmadas como, Window, Combos, Text, Input, etc. En dónde solo requieren algunas especificaciones para determinar aspectos estéticos, aparte del contenido necesario: el layout. El layout es una lista de listas en donde cada elemento ocupa una posición dentro del espacio geográfico del widget. Obviamente considerando los objetos que crean contenedores como Frame, Columns, Tables, etc. Para los Inputs, Text, Button este aspecto se reduce a un único ítem.

En conclusion, PySimpleGUI es un módulo que permite reducir el código en una forma simple de obtener resultados eficientes basado en el lenguaje Python y que no requiere necesariamente el uso de programación orientada a objetos, por lo que permite un mayor rango de implementación.⁷

3. Problemas y soluciones surgidas durante el desarrollo

Algunos de los problemas que tuvimos en la confección del proyecto se presentaron dada la ausencia de conocimiento en el uso de las herramientas, si bien una primera aproximación nos permitió gestionar un desarrollo sinusoidal en cuanto a la implementación de los métodos y

mecanismos de la programación funcional y orientada a objetos. Luego, en una posterior re-comprensión de las problemáticas, se volvió a emplear en un modo más "pythonico" el desarrollo.

Estos problemas fueron entorno a la modularización, al cómo contemplar los errores que podían aparecer, el manejo de archivos, el modelado de los datos, las importaciones relativas y absolutas, etc. Las soluciones a cada problema no fueron únicas en su forma, sino más bien mutables. De cada problema que aparecía le sucedían soluciones en base a las herramientas que aprendíamos. Por ejemplo, para el modelado de los datos nos tomó varias etapas obtener el resultado deseado. En una primera instancia se prepararon los dataset utilizando la librería "**csv**" de python sin mucha consideración sobre los valores "*Nan*" que podían tener las columnas. Luego se implementó la librería "**pandas**" para volver a modelar los datos. Finalmente, en el testeo de la jugabilidad, notamos que faltaba modificar los dataset para que en las pistas de la carta a adivinar no aparezcan los valores "*Nan*". Para los problemas surgidos durante el desarrollo se empleó esa forma, probar, observar, reformular y volver a probar. Una herramienta conceptual que se usó para la solución de determinados problemas fue la concepción del signo lingüístico saussuriano ⁸, en donde a determinada imagen se le asocia un concepto. Por ejemplo, sabíamos que el concepto de generar una pantalla de configuración era obtener la información necesaria para las preferencias del juego, tiempo, cantidad de rondas, cantidad de pistas, etc. Bajo ésta directiva, el desarrollo de la interfaz pretendía simular la imagen del signo que, a su vez, se dividía en sub signos. Estos sub signos podrían representarse como un input (imagen) para representar tiempo (concepto), otro input para representar cantidad de rondas (concepto). Cada concepto tiene su definición, por ejemplo en este caso ser valores numéricos. La unión de los sub signos producen el conjunto de imágenes del que se obtiene la resolución del problema principal: Crear una interfaz de configuración.

4. Consideraciones éticas sobre el desarrollo

4.1. Licencias

4.1.1. ¿Que es una licencia de software?

Es un contrato entre el usuario que usará el software y el propietario del mismo. Este contrato contiene una lista de términos y cláusulas las cuales el usuario final deberá cumplir para poder usar el software. Estos términos y cláusulas le dan al usuario ciertas libertades sobre el software variando en el tipo de licencia que se acuerde sobre el software (comercial, código abierto o software libre).

4.1.2. Tipos de licencias principales

- Código abierto

Existen algunas versiones para este tipo de licencia las cuales el usuario tiene total libertad sobre el software (utilizar el software, analizar y modificar el código fuente y los archivos binarios, y libre distribución del mismo o con algunas modificaciones) y hay otras licencias que pueden llegar a suponer restricciones mínimas como por ejemplo mantener el nombre del creador o incluso solo permitir distribución para uso exclusivamente personal.

- Software Libre

Este suele confundirse con el anterior. La diferencia radica en la gratuidad, ya que, el hecho de que se pueda leer y modificar el código fuente no garantiza la gratuidad del software.

- Copyleft

Esta licencia es la que respalda los software de código libre y software libre. Este tiene

- GNU GPL

La Licencia Pública General GNU (GNU General Public License) es la licencia que acompaña los paquetes distribuidos por el Proyecto GNU. Además se deben incluir una gran variedad de software que incluye el núcleo del sistema operativo Linux.

4.1.3. Diferencias entre software libre y comercial

Se diferencian principalmente en las libertades que ofrece sobre el mismo. Los software libre se caracteriza principalmente por tener la libertad poder usar, analizar el código y modificarlo a placer y tener distribución libre. Algunos ejemplos de este son: Linux, 7-Zip, NotePad++ y FreeCad. Por otra parte el comercial tiene que ser contratado por una suscripción mensual o pagar una sola vez, pero este solo permite el uso del software, por ejemplo Windows (Microsoft), MAC (Apple) o AutoCAD

5. Conclusiones y trabajos futuros

5.1. Código repetitivo

Una conclusión notable a la que llegamos fue que en la construcción de la interfaz la forma adoptada para desarrollar cada pantalla no era de las mejores. Se uso un módulo por separado para cada pantalla, lo cual producía una repetición de código excesivo. Para esto una propuesta de trabajo futuro sería obtener de las definiciones repetitivas una función unificada y reducir la cantidad de veces que se repiten. Por ejemplo, la mayoría de las interfaces están formadas dentro de *frames*, que a su vez contienen columnas. Esa forma se repite en casi todas las pantalla. Lo ideal sería unificarla, creando una clase llamada Interfaz que permita instanciar distintos objetos con característica especiales de cada uno. Otro patrón identificable son los usos de verificación de datos ingresados en los *inputs*, si bien los tipos de datos a verificar pueden ser distintos la forma de respuesta ante un dato no permitido es la misma. Con esta consideración, se podría crear una única función que realice el recorrido de una lista de *inputs* para los cuales se le aplica una lista de *tests* que validen la información contenida.

6. Referencias

- ¹. [↑ Artículo sobre el seminario](#)
- ². [↑ ¿Qué es Python?](#)

³. [↑ Bibliografía:](#)

”Python para todos” de Raúl González Duque

Aprende con Alf

Python para Impacientes

”Python for Everybody” Dr. Charles R. Severance

⁴. [↑ Entornos Virtuales](#)

⁵. [↑ Pandas](#)

⁶. [↑ MatPlot](#)

⁷. [↑ PySimpleGUI](#)

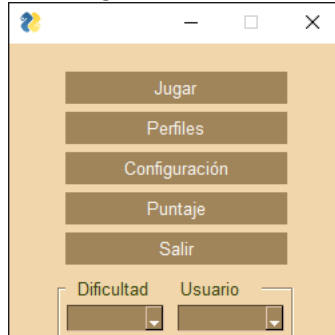
⁸. [↑ Signo Lingüístico](#)

7. Anexo 1 guía de usuario

7.1. Menú principal

En la Figura 1 está la pantalla principal del software donde se presentan varias opciones que incluyen iniciar partida, perfiles, configuración, puntajes y salir, luego, hay dos listas desplegables para usuarios y niveles de dificultad. Para jugar se requiere elegir un usuario y nivel. En la pantalla de perfiles se puede crear y editar usuarios. La pantalla de configuración permite visualizar las dificultades y editarlas. Y por último, la pantalla de puntajes muestra los puntajes individuales de los usuarios por nivel y su promedio entre todas las partidas.

Figura 1: Menú



7.2. Pantalla de juego

En la Figura 2 se encuentra la pantalla de juego de la cual se distinguen cuatro partes principales. Arriba información sobre el jugador, nivel y temática. A la izquierda los puntajes por respuesta, en el centro el tiempo restante de la ronda y a la derecha las pistas y las opciones para adivinar el volcán, lago o jugador de FIFA. El jugador debe elegir una opción y dependiendo si es correcta o no en la siguiente ronda se verá reflejada en la zona de respuestas el puntaje.

Figura 2: Pantalla Juego

Lagos **Dificultad: Normal** **Player: Sai**

0/10

Ubicación:
1. Chubut

Superficie (km²):
2. 10

Profundidad máxima (m):
3. 57.0

Profundidad media (m):
4. 25.0

Coordenadas:
5. -43.26,-71.35

Lago Epuyén

Lago Futalaufquen

Lago Gutiérrez

Lago Chollila

Lago Rosario

Pasar Abandonar

Respuestas

1:

2:

3:

4:

5:

6:

7:

8:

9:

10:

Tiempo

28

En la parte inferior derecha de la pantalla aparecen las opciones para pasar de pregunta o abandonar la partida. En caso de abandonar la partida saldrá un mensaje como el de la Figura 3.

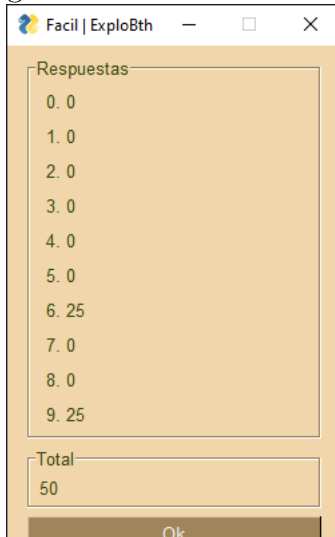
Figura 3: Partida Abandonada

Partida Abandonada :(

OK

Al finalizar la partida, sin abandonar, se podrá ver el puntaje obtenido de la forma en la que se muestra en la Figura 4. Los puntos por cada opción elegida correspondientes al nivel de dificultad, y el total acumulado.

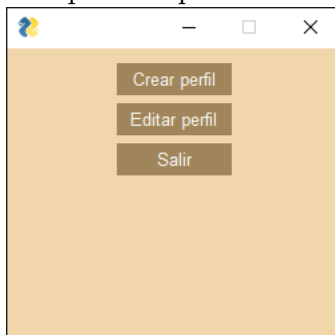
Figura 4: Partida Finalizada



7.3. Pantalla de perfiles

En ésta pantalla se puede crear y editar usuarios del software. En la Figura 5 se puede ver la primera de las pantalla de perfiles en la cual se puede elegir entre crear un nuevo perfil, editar uno existente o volver al menú principal.

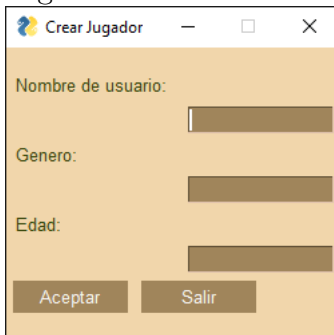
Figura 5: Opciones pantalla de perfiles



7.3.1. Pantalla de creación de perfiles

En la figura 6 se muestra la pantalla de creación de usuarios. Para crear un usuario se debe ingresar un nombre que no esté registrado, un género y una edad. Para el género solo están permitidos los valores alfabéticos y para la edad valores numéricos.

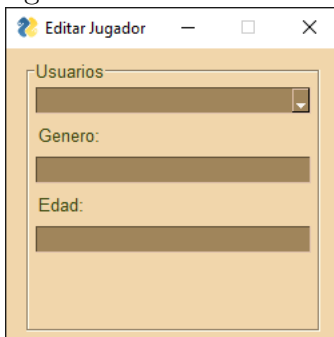
Figura 6: Crear Usuario



7.3.2. Pantalla de edición de perfiles

En la Figura 7 se ve la pantalla de edición de perfil. Para editar un jugador se elige el nombre del usuario, esto mostrará los valores con los que está registrado. Género y edad, son los único editable.

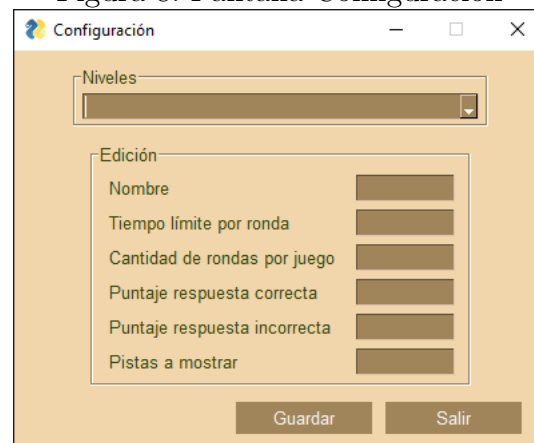
Figura 7: Edición Usuario



7.4. Pantalla de configuración

En la Figura 8 se encuentra la pantalla de configuración. En ésta pantalla se visualiza la información referida a cada nivel. Tiempo de la ronda, cantidad de rondas, puntos por respuesta correcta, puntos por respuesta incorrecta y cantidad de pistas. Los niveles por defecto son *Fcil*, *Normal* y *Difcil*. Existe la posibilidad de modificar la configuración de un único nivel, el *Custom*.

Figura 8: Pantalla Configuración

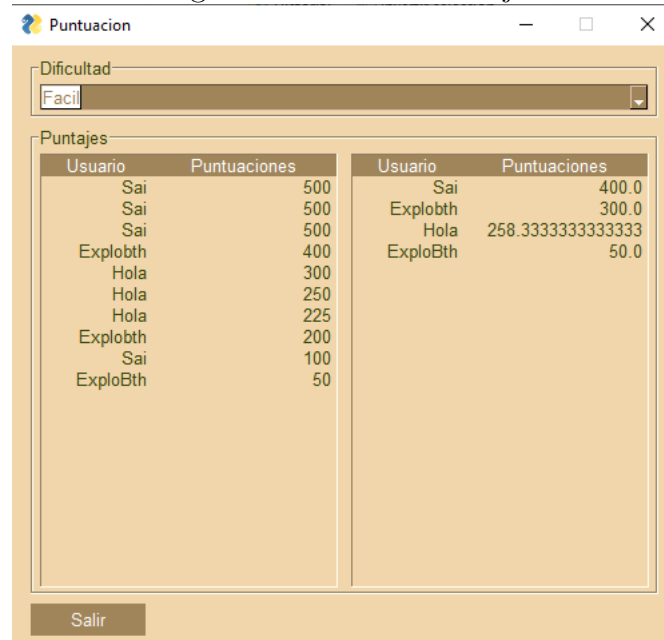


The image shows a software window titled "Configuración" with standard window controls (minimize, maximize, close). Inside the window, there is a dropdown menu labeled "Niveles". Below this is a section titled "Edición" which contains six rows, each with a label and a corresponding text input field: "Nombre", "Tiempo límite por ronda", "Cantidad de rondas por juego", "Puntaje respuesta correcta", "Puntaje respuesta incorrecta", and "Pistas a mostrar". At the bottom of the window, there are two buttons: "Guardar" and "Salir".

7.5. Pantalla de puntuación

En la Figura 9 se encuentra la pantalla de puntuación. En esta pantalla se puede ver los mejores 20 puntajes y los mejores 20 promedios de los usuarios según la dificultad seleccionada.

Figura 9: Pantalla Puntaje



8. Anexo 2 guía para el desarrollador

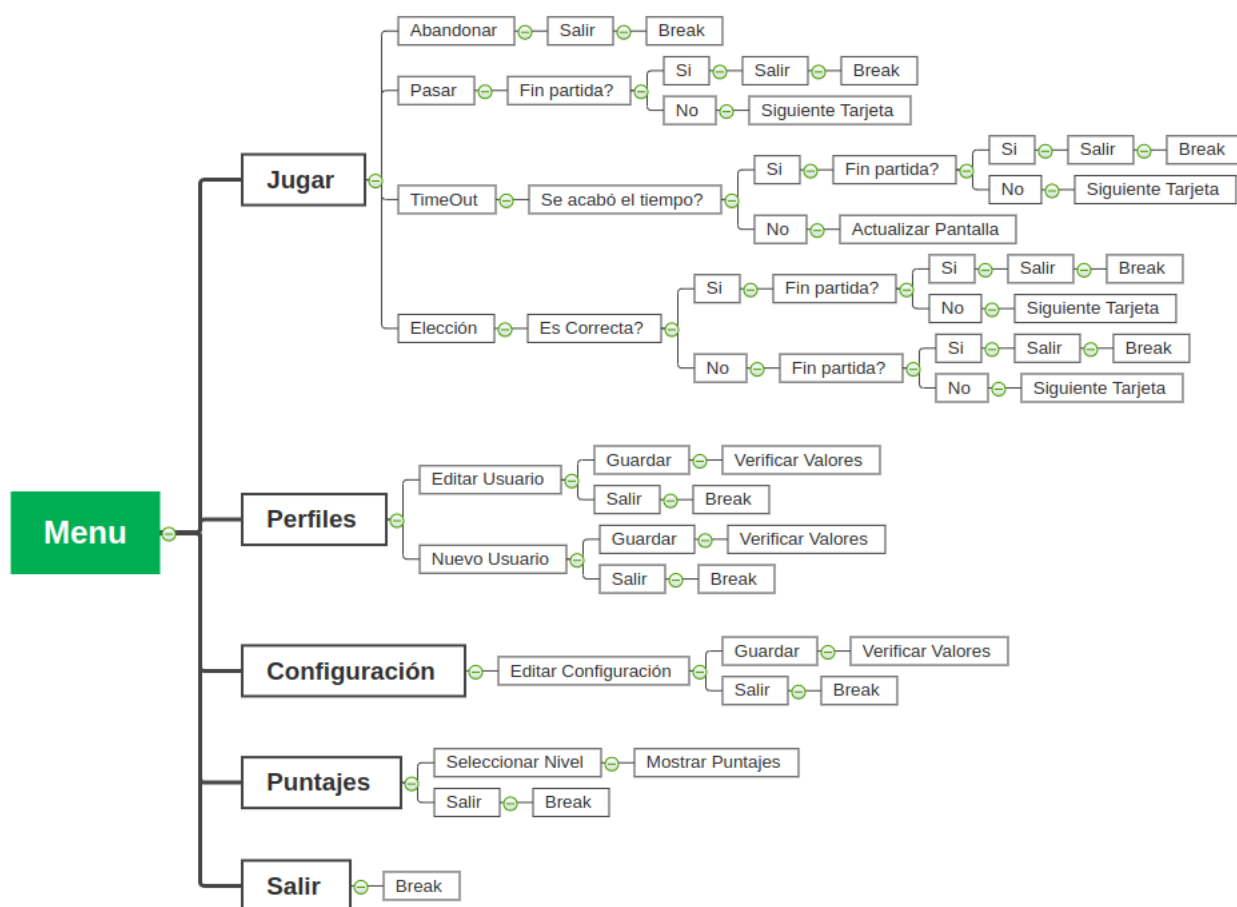
8.1. Ejecución de Loops

La ejecución de loops está basada en la lectura de eventos dentro de un *while True* que focalizando en un match-case los casos posibles actúa según la coincidencia. Para determinado evento, hay una respuesta. Por ejemplo, la ejecución del loop de la pantalla principal establece un case dónde cada posibilidad contemplada representa los botones que el usuario tiene disponible para interactuar. Jugar, Perfiles, Configuración, Puntaje y Salir. Dependiendo del evento generado por la función incorporada, en el objeto Window de PySimpleGUI, *read()* la coincidencia generaría ejecutar un determinado loop. Este mecanismo se

utiliza para todas las ventanas. Un *while True* que está a la espera de algún evento en la pantalla. Para el caso de la pantalla Perfil, los casos son distintos pero la metodología es la misma. Un *while True* que se mantiene a la espera de algún evento en la pantalla.

8.2. Los casos

Figura 10: Casos



Como se ve en la Figura 9 cada Pantalla tienen casos determinados que comprenden las posibilidades del usuario.

■ Jugar:

- A todas las acciones durante la partida le corresponden una escritura en el archivo de registro. Cada escritura en el registro tiene el formato:

timestamp|id|evento|usuario|estado|textoingresado|respuesta|nivel

Dado que el nivel, el usuario y el id son siempre los mismo se "setean" al principio de la partida y sólo el resto tienen valores dinámicos. A su vez los valores de timestamp no dependen de la interacción directa entre el usuario y el juego como si lo es elegir una opción. Por lo que solo se va a especificar sobre evento, estado, texto ingresado y respuesta.

- Evento Pasar:

Al presionar el botón **pasar** el evento coincide con el caso 'pasar' y establece el registro con los valores [*intento| paso|-|correcta*] para luego actualizar la pantalla, si es que quedan rondas por jugar, con la siguiente carta a adivinar.

- Evento Abandonar:

Al presionar el botón **abandonar** el evento coincide con el caso abandonar, por lo que se establece la lista de respuestas como None, se escribe en el registro [*"fin", "abandono", "-", "-"*] y se cierra la ventana. Al tener la lista de respuestas en None se evita el calculo del puntaje total y la creación de la ventana *puntajetotal*.

- Evento TimeOut:

El evento time out está relacionado con la cantidad de segundos disponibles para adivinar la carta según la dificultad. Éste evento tiene dos posibilidades que se halla terminando el tiempo para adivinar la carta actual o no. En el caso de no haber llegado a cero, se actualiza la pantalla. Por el otro lado se escribe en el registro [*"intento", "timeout", "-", "correcta"*] y si quedan rondas por adivinar se actualiza la pantalla con

una nueva carta. En caso contrario, se calcula el puntaje total y se crea una ventana con el puntaje por respuesta y el total.

- Perfiles:

Esta pantalla se subdivide en 2 pantallas mas ya que está destinada para la creación/edición de usuario. La primer pantalla tiene 3 tipos de eventos: *–Create–* | *–Edit–* | *–Exit–*

- Evento *–Create–*:

Este evento genera la pantalla de creación de usuarios. Para la creación de usuario se requiere que el usuario ingrese el nombre, género y edad de forma correcta (por ejemplo en edad no se debe ingresar ninguna letra o carácter especial). Esta pantalla tiene 2 eventos: *–Accept–* | *–Exit–*

- Evento *–Accept–* en la pantalla crear:

El evento *–Accept–* genera un nuevo usuario el archivo *json* de usuarios. Antes de eso realiza un llamamiento al módulo verificar que verifica que en la casilla de edad no se haya ingresado ningún carácter que no sea un valor numérico entero o que se haya dejado la casilla vacía, que en genero no se haya ingresado ningún valor numérico o se haya dejado vacía y por ultimo que el nombre ingresado por el usuario no exista ya en el archivo de usuarios. En caso que se puede almacenar aparece una pantalla con la confirmación que se pudo agregar el usuario de forma correcta.

- Evento *–Exit–* en la pantalla crear:

El evento *–Exit–* cierra la pantalla crear perfil y vuelve a la pantalla de perfiles

- Evento *–Edit–*:

Este evento genera la pantalla edición de usuarios. En esta pantalla se muestra un *combo* donde se muestran los usuarios existentes para seleccionar uno y editarlo. Luego de seleccionar el usuario para editar se actualizan los *inputText* de edad y genero con la información que se tiene del mismo para la edición. Esta pantalla tiene 2 eventos: *–Accept–* | *–seleccion–* | *–Exit–*

- Evento *–Accept–* en la pantalla editar:
El evento *–Accept–* realiza la misma verificación que el de la pantalla crear pero, a diferencia de la pantalla crear, actualiza el usuario que se eligió para editar sobre escribiendo la información ya almacenada del usuario
- Evento *–seleccion–* en la pantalla editar:
El evento *–seleccion–* se activa cuando el usuario selecciona un perfil en el *combo* y este actualiza los *inputText* con la información almacenada del perfil seleccionado en el archivo de perfiles
- Evento *–Exit–* en la pantalla editar:
El evento *–Exit–* realiza las mismas acciones que el de la pantalla crear

■ Configuración:

Esta pantalla permite ver las características de las diferentes dificultades que existe en el juego (fácil, normal, difícil y *custom*) en las cuales muestran la información, de cada característica, en un *inputText* de la dificultad seleccionada en el combo por el usuario. Esta pantalla maneja 3 tipos de eventos: *–done–* | *–list–* | *–save–*

- Evento *–done–*:
Este evento cierra la pantalla configuración. Se utiliza cuando el usuario quiere salir de la pantalla.
- Evento *–list–*:
Este evento sucede cuando un usuario selecciona una dificultad en el combo, entonces se actualizan los *inputText* con la información de la dificultad seleccionada, esta información esta almacenada en el *json* de las dificultades.
- Evento *–save–*:
Este evento guarda la información editada de la dificultad *custom*, para guardar esta información primero verifica que dificultad esta seleccionada en el *combo*, ya que la única que se puede editar es la de *custom*, además verifica que en todos los *inputText* haya información valida (ya que si o si se

deben ingresar caracteres numéricos enteros y tampoco puede quedar vacía la casilla), al intentar guardar la información pueden suceder tres tipos de casos, el primero es que la información no sea válida saldrá el mensaje "Valor no valido {val} entrada: {self._labels[key]}", el valor *val* indica el valor ingresado en la casilla y {self._labels[key]} indica en que casilla se ingresó. El segundo caso es que se haya intentado editar otro nivel que no sea el *custom*, en cuyo caso saldrá el mensaje "Nivel no editable". El tercer y último caso es cuando la información se ingresó correctamente y se intentó editar el nivel *custom* se guardara de forma correcta y se quedara en la pantalla configuración hasta que ocurra el evento *—done—*

- Puntajes:

Esta pantalla muestra las mejores 20 (veinte) puntuaciones y promedios de las puntuaciones de los usuarios según la dificultad elegida. La dificultad se selecciona en un *combo*, representando los valores en dos *table* (uno para las puntuaciones y el otro para los promedios), estas tablas están divididas cada una por *Usuario* y *Puntaje*. La pantalla presenta dos tipos de eventos: *—choise—* | *—Exit—*

- Evento *—choise—*:

Este evento toma del *json* de *scores* todos los puntajes de los usuarios en esa dificultad para luego filtrar los mejores veinte puntajes mas altos y actualiza el *table* de los puntajes y también hace el promedio de todos los puntajes de cada usuario de la dificultad seleccionada en el *combo* y devuelve los mejores veinte de los promedio y actualiza el *table* de los promedios.

- Evento *—Exit—*:

Este evento sale de la pantalla de puntajes.

- Salir:

El uso del botón Salir en la pantalla de Menú implica el break del *while True* cerrando la aplicación y dando por terminada la ejecución de la aplicación.

8.3. Accesos a archivos

El acceso de los archivos está logrado por dos módulos "genéricos". Por un lado el *FilesGetter* se ocupa de obtener las rutas de los archivos retornándolos como diccionario dónde la clave es el nombre del archivo y el valor la ruta a éste. Y, por el otro, el módulo *DAO* (Data Access Objects) se ocupa de abrir un archivo según la petición que se hace y retornar el contenido. La clase *FilesGetter* del módulo rutas tiene una función *get_directory()*, ésta recibe por parámetro dos argumentos; tipo de archivo (su terminación "csv" o "json") y la carpeta en la que está. Desde el módulo *DAO* se pueden invocar tres objetos distintos según el tipo de archivo. Archivos *.json* a través de la clase *JsonsFiles*, los "datasets" a través de la clase *CsvFiles* y el archivo de registro del juego a través de la clase *Registro*. Tanto el *CsvFiles* como el *JsonsFiles* reciben por parámetro el nombre del archivo que se desea abrir, por ejemplo para los jsons las posibilidades son "config". "scores" o "users". Puede pasar que los archivos no estén en la carpeta, pero para esta posibilidad se crean archivos con la estructura por defecto y vacíos. Cómo la implementación depende del desarrollador, está presupuesto que no se van a modificar las lineas por opciones de archivos que no existen en la carpeta. Pero si sucediera, la irrupción del programa se produciría por un *KeyError*.