

Laboratorio No 2. Servidor de Jugadores y Preguntas

Integrantes

Catalina Arcila Grisales

Angela María Bolaños Moncayo

Juan Camilo Poveda Delgado

Ingeniería electrónica

Programación

Profesor

Alexander López – Julián Barrero

24 de septiembre del 2024

Laboratorio 2, Servidor de Jugadores y Preguntas

Resumen- En el siguiente documento se proporciona un conjunto de archivos de los cuales solo uno debe ser modificado para crear el registro de los usuarios, inicio de sesión y actualización de puntajes, además de solicitar preguntas aleatorias de una categoría determinada, los demás archivos no deben ser modificados y se usan como prueba para validar el funcionamiento del servidor. Finalmente, se analizan los conceptos aprendidos y aplicados, como el uso de estructuras de datos, programación orientada a objetos, manejo de archivos, y la importancia funciones óptimas que permitan el acceso adecuado de los datos.

Palabras clave - Diccionarios, sentencias condicionales, listas, memoria, Python.

I. INTRODUCCIÓN

En este documento se demuestra el uso de los conocimientos y habilidades adquiridas durante el proceso de aprendizaje en programación con el lenguaje Python, con el propósito de aplicarlos en un nuevo reto, haciendo uso de archivos los cuales se usan para guardar y actualizar información haciendo referencia al nombre, contraseña y puntaje de cualquier usuario establecido, además de obtener la lista de usuarios conectados y solicitar alguna pregunta aleatoria suministrada de uno de los archivos creados para esta función. Cabe destacar el uso de diccionarios en este laboratorio ya que es una estructura que permite almacenar y organizar información de manera eficiente.

A continuación, se presenta el reto que consiste en complementar el archivo *user.py* siguiendo las instrucciones indicadas en los comentarios.

II. OBJETIVOS

Objetivo general:

Analizar el problema estipulado por el docente para hallar la solución más eficaz mediante conocimientos anteriormente adquiridos, además de aprender a manejarlos junto a nuevas funciones como archivos de texto.

Objetivos específicos:

- Optimizar recursos computacionales.
- Optimizar memoria en CPU.

- Comprender el funcionamiento de diccionarios y archivos en Python.
- Comprender la programación orientada a objetos.

III. MÉTODOS E INSTRUMENTOS

Se suministra el código base del servidor en el archivo *trivia_sever.py* el cual contiene toda la funcionalidad para que éste opere dentro de una red de área local o en el mismo equipo de prueba. [1]

Para este laboratorio se pide completar un código (*users.py*), un añadiendo un servidor y otras funciones usando archivos como JSON [2] que representan diccionarios. Inicialmente, se propone el siguiente código en Python:

```
1 from json import load, dump, JSONDecodeError
2 from random import choice
3
4 def registerUser(name,password):
5     try:
6         with open("users.txt", "r") as file:
7             usuarios = load(file)
8     except (FileNotFoundError, JSONDecodeError):
9         usuarios = []
10
11     name = input("Nombre: ")
12
13     # Verificar si el usuario existe
14     user_exist = False
15     for usuario in usuarios:
16         if usuario["name"] == name:
17             user_exist = True
18             break
19
20     if user_exist:
21         print("User already registered.")
22     else:
23         password = input("Clave: ")
24
25         # Crea un nuevo usuario
26         nuevo_usuario = {
27             "name": name,
28             "password": password,
29             "active": 0,
30             "score": 0
31         }
32
33     # Agrega al nuevo usuario al archivo
34     usuarios.append(nuevo_usuario)
35
36     with open("users.txt", "w") as file:
37         dump(usuarios, file, indent=4)
38     print("User successfully registered.")
39
40 def openCloseSession(name,password,flag):
41     try:
42         with open("users.txt", "r") as file:
43             usuarios = load(file)
44     except (FileNotFoundError, JSONDecodeError):
45         return "error"
46
47     # Buscar el usuario
48     for usuario in usuarios:
49         if usuario["name"] == name and usuario["password"] == password:
50             # Verificar el estado de la sesión según flag
51             if flag == 1:
52                 usuario["active"] = 1
53                 session = "Session was successfully opened"
54             elif flag == 0:
55                 usuario["active"] = 0
56                 session = "Session was successfully closed"
57             else:
58                 return "error"
59
60     # Actualizar el estado
61     with open("users.txt", "w") as file:
62         dump(usuarios, file, indent=4)
```

```

40 def openCloseSession(name,password,flag):
41
42     return session
43
44     return "error"
45
46 def updateScore(name,password,score):
47     try:
48         with open("users.txt", "r") as file:
49             usuarios = load(file)
50     except (FileNotFoundError, JSONDecodeError):
51         return "error"
52
53     for usuario in usuarios:
54         if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
55             usuario["score"] += score
56
57     # Actualizar el archivo
58     with open("users.txt", "w") as file:
59         dump(usuarios, file, indent=4)
60
61     return "Score was successfully updated"
62
63 return "error"
64
65 def getScore(name,password):
66     try:
67         with open("users.txt", "r") as file:
68             usuarios = load(file)
69     except (FileNotFoundError, JSONDecodeError):
70         return "error"
71
72

```

```

87 def getScore(name,password):
88     for usuario in usuarios:
89         if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
90             return usuario["score"]
91
92     return "error"
93
94 def usersList(name,password):
95     try:
96         with open("users.txt", "r") as file:
97             usuarios = load(file)
98     except (FileNotFoundError, JSONDecodeError):
99         print("error")
100     else:
101         active_user = False
102
103         # Revisar si está activo
104         for usuario in usuarios:
105             if usuario["active"] == 1:
106                 print(f"Nombre: {usuario['name']}, Puntaje: {usuario['score']}")
107                 active_user = True
108
109         # Si no se encontró ningún usuario activo
110         if not active_user:
111             return "error"
112
113 def question(name,password,cat):
114     # Verificar si el usuario tiene sesión activa
115     try:
116         with open("users.txt", "r", encoding="utf-8") as file:
117             usuarios = load(file)
118     except (FileNotFoundError, JSONDecodeError):
119

```

```

119 def question(name,password,cat):
120     return "error: archivo de usuarios no encontrado"
121
122     user_found = False
123     for usuario in usuarios:
124         if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
125             user_found = True
126             puntuacion = 0
127
128     todas_preguntas = []
129
130     # Cargar preguntas de historia
131     try:
132         with open("question_history.txt", "r", encoding="utf-8") as file:
133             preguntas_historia = load(file)
134             for pregunta in preguntas_historia:
135                 pregunta["categoria"] = 'Historia'
136                 todas_preguntas.append(pregunta)
137     except (FileNotFoundError, JSONDecodeError):
138         print("No se pudo cargar el archivo de historia")
139
140     # Cargar preguntas de ciencia
141     try:
142         with open("question_science.txt", "r", encoding="utf-8") as file:
143             preguntas_ciencia = load(file)
144             for pregunta in preguntas_ciencia:
145                 pregunta["categoria"] = 'Ciencia'
146                 todas_preguntas.append(pregunta)
147     except (FileNotFoundError, JSONDecodeError):
148         print("No se pudo cargar el archivo de ciencia")
149
150     if not todas_preguntas:

```

```

150 def question(name,password,cat):
151     return "error"
152
153     # Mostrar las preguntas al usuario eligiendo categoría al azar
154     while todas_preguntas:
155         # Seleccionar una pregunta al azar
156         pregunta = random.choice(todas_preguntas)
157
158         print(f"\nCategoría: {pregunta['categoria']}")
159         print(f"Pregunta: {pregunta['pregunta']}")
160
161         for opcion, texto in pregunta["opciones"].items():
162             print(f"({opcion}): {texto}")
163
164         while True:
165             respuesta = input("Selecciona una opción (A, B, C, D) o presiona 'F' para salir del juego: ")
166             if respuesta in ['A', 'B', 'C', 'D', 'F']:
167                 break
168             print("Opción no válida. Inténtalo de nuevo.")
169
170         if respuesta == 'F':
171             print("Juego terminado.")
172             break
173
174         # Verificar la respuesta correcta
175         if respuesta == pregunta["respuesta correcta"]:
176             print("Correcto")
177             puntuacion += 2
178         else:
179             print("Incorrecto")
180             puntuacion -= 1
181
182     print(f"Puntuación actual: {puntuacion}")
183
184     # Remover la pregunta actual de la lista
185     todas_preguntas.remove(pregunta)
186
187     # Informar cuando haya respondido todas las preguntas
188     if not todas_preguntas:
189         print("\nHas respondido todas las preguntas.")
190         print(f"Puntuación final: {puntuacion}")
191         print("Preguntas completadas")
192
193     return "error"

```

Fig. 1. Código en Python.

Para la primera función (Línea 4), (registerUser) Su funcionamiento se basa en que ingrese un usuario usando los parámetro name y password con los cuales se le pedirá usuario y contraseña con los cuales verifica en un archivo previamente creado tipo JSON [2], el cual guarda la información de los usuarios registrados, estos datos los compara con el archivo para al final retornar nos posibles opciones, le dirá al usuario que no está registrado mostrando un error usando la función try except , o por otro lado lo aceptara como un usuario ya registrado y lo dejara ingresar

Segunda función (Línea 40), (openCloseSession), Tiene la función de revisar si un usuario ya ha sido registrado anteriormente, logra esto haciéndolo de la siguiente manera, usando los parámetros name password se verifica si el usuario está registrado en el archivo JSON el cual se lograra leer por medio de la función load de esta manera se corrobora que si existe el usuario, si no esta se retornará un error controlado usando try except, de lo contrario usará la función flag el cual cambiará su valor a 1 si se inició sesión correctamente o 0 cuando se termina o cierra la sección, o en el caso que el valor sea diferente de 0 y 1 retornará un mensaje de error

Tercera función (Línea 68), (updateScore), La función se ejecutará satisfactoriamente si se cumplen tres condiciones, que el usuario, contraseña, y sección activa (1) sean correctas para después leer la información en el archivo users verificando algún usuario con las condiciones iguales, si esto cumple se actualizará score sumándose al puntaje actual que tenía previamente guardado, de ser lo contrario ya sea que no se encuentra el usuario, o no esta bien estructura el documento, devolverá un error

Cuarta función (Línea 87), (getScore), Esta función se ejecutará cuando se verifique que se cumplan tres valores, name, password y flag o en este caso sección activa, si esto es así le retornara al usuario su puntaje actual en el juego, o de lo contrario si faltara o estuviera incorrecto alguno de los valores o en debido caso que estuviera mal estructurado el archivo JSON o presentara alguna falla, va a retornar como un error

Quinta función (Línea 100), (usersList), La siguiente función se ejecutará la de la forma deseada cuando la función active_user se ejecuta en el programa buscando jugadores o usuario activos(flag=1) para que de esta manera pueda crear una tabla de ranking de los jugadores que están jugando en vivo, pero si al finalizar de escanear el documento users no encuentra ningún jugador con la sección activada devolverá un error ya que no tendrá información para construir la tabla de ranking

Por último, la sexta función (Línea 119), (question), Para que la siguiente función se lleve a acabo el usuario tiene que colocar correctamente su name y password si esto es correcto se dará inicio al programa en el cual se le proporcionará al jugador preguntas provenientes de dos listas llamas ciencia y historia, las cuales se unirán en una sola lista desde la cual se proporcionan preguntas aleatorias teniendo en cuenta de cuál categoría han sido suministradas, si el usuario responde correctamente la pregunta se le se aumenta 2 puntos a su puntuación, de lo contrario se le restará 1 punto, se cerrará la sección cuando el jugador termine de responder todas las preguntas otorgadas, o presionando la tecla f con la cual saldrá del juego y se cerrará la sección dando así fin al juego.

A continuación, se muestran los archivos utilizados para el funcionamiento del código:

```

1 users.txt
2 {
3   {
4     "name": "catalina",
5     "password": "00000",
6     "active": 1,
7     "score": 100
8   },
9   {
10    "name": "isabella",
11    "password": "1010",
12    "active": 0,
13    "score": 0
14  },
15  {
16    "name": "maíno",
17    "password": "0007",
18    "active": 0,
19    "score": 0
20  },
21  {
22    "name": "lilo",
23    "password": "4545",
24    "active": 1,
25    "score": 0
26  },
27  {
28    "name": "julian",
29    "password": "32345",
30    "active": 0,
31    "score": 0
32  }
33 }

```

Fig. 2. Archivo (users.txt) que almacena al usuario.

```

1 question_history
2 {
3   "pregunta": "¿Cómo se les llamaba a los antiguos gobernantes de Egipto?", "opciones": ["A": "Farosones", "B": "Reyes", "C": "Emperadores", "D": "Reinas"], "respuesta correcta": "B", "pregunta": "Según las leyendas, ¿quién fue el fundador de Roma?", "opciones": ["A": "César y Pompeyo", "B": "Rómulo y Remo", "C": "Néstor y Odiseo", "D": "Hércules y Atenea"], "respuesta correcta": "B", "pregunta": "¿Cuál fue la primera civilización conocida?", "opciones": ["A": "Sumerios", "B": "Sotegios", "C": "Romanos", "D": "Egipcios"], "respuesta correcta": "A", "pregunta": "¿Qué imperio fue conocido por su red de caminos y sistema de correo?", "opciones": ["A": "Mongol", "B": "Bizantino", "C": "Mao", "D": "Mao"], "respuesta correcta": "A", "pregunta": "¿Cuál de los siguientes fue un filósofo griego famoso?", "opciones": ["A": "Confucio", "B": "Sócrates", "C": "Platón", "D": "Buda"], "respuesta correcta": "B", "pregunta": "¿En qué continente se desarrolló la civilización del antiguo Egipto?", "opciones": ["A": "Asia", "B": "Europa", "C": "Asia", "D": "Asia"], "respuesta correcta": "A", "pregunta": "¿Qué dinastía unificó a China en el siglo III a.C?", "opciones": ["A": "Dinastía Tang", "B": "Dinastía Qin", "C": "Dinastía Han", "D": "Dinastía Ming"], "respuesta correcta": "B", "pregunta": "¿Quién fue el líder de la India en su lucha por la independencia?", "opciones": ["A": "Jawaharlal Nehru", "B": "Mahatma Gandhi", "C": "Sardar Patel", "D": "Bhaskar Rana"], "respuesta correcta": "B", "pregunta": "¿Qué civilización es famosa por sus ruinas en Machu Picchu?", "opciones": ["A": "Incas", "B": "Aztecas", "C": "Mayas", "D": "Incas"], "respuesta correcta": "A", "pregunta": "¿Qué fue la ruta de la seda?", "opciones": ["A": "Un sistema de defensa", "B": "Un tratado de paz", "C": "Una ruta comercial", "D": "Una ruta marítima"], "respuesta correcta": "C"
4 }

```

```

1 question_science
2 {
3   "pregunta": "¿Cuántos huesos hay en el cuerpo humano?", "opciones": ["A": "200", "B": "262", "C": "340", "D": "380"], "respuesta correcta": "B", "pregunta": "¿Qué misión lunar Apollo fue la primera en llevar un vehículo lunar?", "opciones": ["A": "Misión Apollo II", "B": "Misión Apollo I", "C": "Misión Apollo III", "D": "Misión Apollo IV"], "respuesta correcta": "A", "pregunta": "¿A qué temperatura el agua se evapora?", "opciones": ["A": "37 grados", "B": "100 grados", "C": "131 grados", "D": "78 grados"], "respuesta correcta": "B", "pregunta": "En qué planeta se encuentra la montaña volcánica Olympus Mons?", "opciones": ["A": "Júpiter", "B": "Saturno", "C": "Marte", "D": "Venus"], "respuesta correcta": "C", "pregunta": "¿Cuál es el proceso mediante el cual las plantas convierten la energía luminosa en energía química?", "opciones": ["A": "Fotosíntesis", "B": "Respiración celular", "C": "Fermentación", "D": "Oxidación"], "respuesta correcta": "A", "pregunta": "¿Cuál es el órgano más grande del cuerpo humano?", "opciones": ["A": "Piel", "B": "Intestino grueso", "C": "Intestino delgado", "D": "Hígado"], "respuesta correcta": "A", "pregunta": "¿Cuál es el símbolo químico del oro?", "opciones": ["A": "O", "B": "Au", "C": "Ag", "D": "Fe"], "respuesta correcta": "B", "pregunta": "¿Cómo se llama el planeta más pequeño de nuestro sistema solar?", "opciones": ["A": "Júpiter", "B": "Mercurio", "C": "Venus", "D": "Marte"], "respuesta correcta": "B", "pregunta": "¿Cómo se llama el océano más grande del mundo?", "opciones": ["A": "Atlántico", "B": "Ártico", "C": "Antártico", "D": "Pacífico"], "respuesta correcta": "A", "pregunta": "¿La Tierra gira sobre su eje una vez cada?", "opciones": ["A": "364 días", "B": "24 horas", "C": "12 horas", "D": "120 días"], "respuesta correcta": "B"
4 }

```

Fig. 2.1. Archivo con preguntas de historia (question history) y ciencia (question science).

Por otro lado, se presentan las pruebas unitarias de cada función, siguiendo el orden de las funciones del código (Observar Fig. 1):

```

PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Nombre: catalina
User already registered.
PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Ingresar el nombre para iniciar/cerrar sesión: catalina
Ingresar la contraseña: 00000
Ingresar el nombre de usuario: catalina
Ingresar la contraseña: 00000
Session was successfully opened
PS C:\Users\Vegela\Documents>

```

```

PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Nombre: catalina, Puntaje: 100
PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Ingresar el nombre de usuario: catalina
Ingresar la contraseña: 00000
Categoría: Historia
Pregunta: ¿Quién fue el líder de la India en su lucha por la independencia?
A: Jawaharlal Nehru
B: Mahatma Gandhi
C: Sardar Patel
D: Bhaskar Rana
Selecciona una opción (A, B, C, D) o presiona 'f' para salir del juego:

```

```

PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Ingresar el nombre de usuario: catalina
Ingresar la contraseña: 00000
100
PS C:\Users\Vegela\Documents> python3.12.exe "C:\Users\Vegela\AppData\Local\Microsoft\WindowsApps\python3.12.exe" "C:\Users\Vegela\Documents\trivia_demo\demo.py"
Nombre: catalina, Puntaje: 100
Nombre: lilo, Puntaje: 0

```

Fig. 3. Pruebas unitarias de cada función.

Uso del espacio en memoria: En cuanto al uso de memoria, el uso del archivo que contenía las preguntas definidas por cada categoría, era el principal consumidor de recursos ya que además de almacenar la lista de preguntas llamaba a la función que contenía la información de cada usuario con su nombre contraseña, actualización de sesión y puntaje.

CONCLUSIONES

1- Implementar un sistema sobre gestión de usuarios, incluyendo registro, autenticación y actualización de cada uno de ellos, sumado a esto se trabaja con un código base existente que permite verificar la funcionalidad del programa.

2-El manejo de archivos y el uso de estructuras de datos como diccionarios fueron útiles para almacenar y organizar la información requerida, además de comprender mejor los conceptos de programación orientada a objetos.

REFERENCIAS

- [1] “JSON introduction”, *W3schools.com*. [En línea].
Disponible en:
https://www.w3schools.com/js/js_json_intro.asp.
[Consultado: 26-oct-2024].

- [2] A. López-Parrado, *Lab2*. .