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

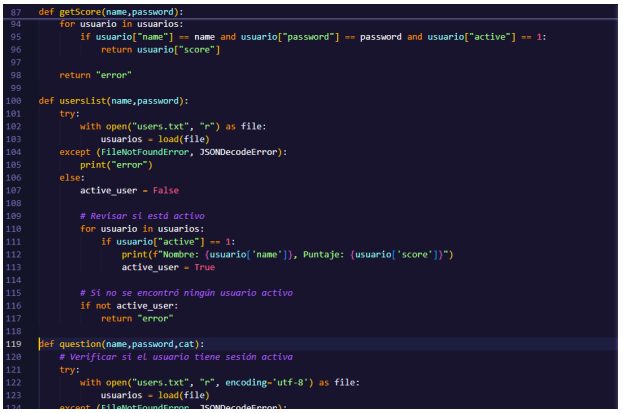
Se suministra el código base del servidor en el archivo *trivia_sever.py* el cual contiene toda la funcionalidad para que éste opere dentro de una red de área local o en el mismo equipo de prueba. [1]

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:

```
C > Users > Angela > Documents > trivia > 🐍 users.py > ⓕ question
from json import load, dump, JSONDecodeError
from random import choice

def registerUser(name,password):
    try:
        with open("users.txt", "r") as file:
            usuarios = load(file)
    except (FileNotFoundError, JSONDecodeError):
        usuarios = []

    name = input("Nombre: ")

    # Verificar si el usuario existe
    user_exist = False
    for usuario in usuarios:
        if usuario["name"] == name:
            user_exist = True
            break

    if user_exist:
        print("User already registered.")
    else:
        password = input("Clave: ")

        # Crea un nuevo usuario
        nuevo_usuario = {
            "name": name,
            "password": password,
            "active": 0,
            "score": 0
        }
```

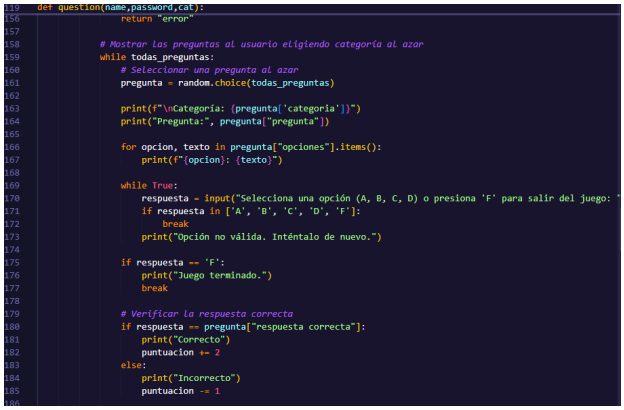
```
def registerUser(name,password):
        # Agrega al nuevo usuario al archivo
        usuarios.append(nuevo_usuario)

        with open("users.txt", "w") as file:
            dump(usuarios, file, indent=4)
        print("User successfully registered.")

def openCloseSession(name,password,flag):
    try:
        with open("users.txt", "r") as file:
            usuarios = load(file)
    except (FileNotFoundError, JSONDecodeError):
        return "error"

    # Buscar el usuario
    for usuario in usuarios:
        if usuario["name"] == name and usuario["password"] == password:
            # Verificar el estado de la sesión según flag
            if flag == 1:
                usuario["active"] = 1
                session = "Session was successfully opened"
            elif flag == 0:
                usuario["active"] = 0
                session = "Session was successfully closed"
            else:
                return "error"

            # Actualizar el estado
            with open("users.txt", "w") as file:
                dump(usuarios, file, indent=4)
```

```
def openCloseSession(name,password,flag):

            return session

    return "error"

def updateScore(name,password,score):
    try:
        with open("users.txt", "r") as file:
            usuarios = load(file)
    except (FileNotFoundError, JSONDecodeError):
        return "error"

    for usuario in usuarios:
        if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
            usuario["score"] += score

            # Actualizar el archivo
            with open("users.txt", "w") as file:
                dump(usuarios, file, indent=4)

            return "Score was successfully updated"

    return "error"

def getScore(name,password):
    try:
        with open("users.txt", "r") as file:
            usuarios = load(file)
    except (FileNotFoundError, JSONDecodeError):
        return "error"
```

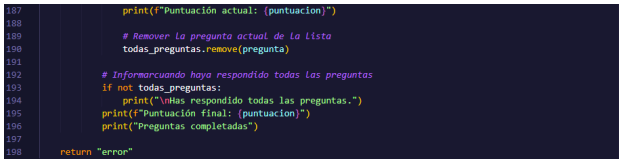


```
def getScore(name,password):
    for usuario in usuarios:
        if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
            return usuario["score"]

    return "error"

def usersList(name,password):
    try:
        with open("users.txt", "r") as file:
            usuarios = load(file)
    except (FileNotFoundError, JSONDecodeError):
        print("error")
    else:
        active_user = False

        # Revisar si está activo
        for usuario in usuarios:
            if usuario["active"] == 1:
                print(f"Nombre: {usuario['name']}, Puntaje: {usuario['score']}")
                active_user = True

        # Si no se encontró ningún usuario activo
        if not active_user:
            return "error"

def question(name,password,cat):
    # Verificar si el usuario tiene sesión activa
    try:
        with open("users.txt", "r", encoding='utf-8') as file:
            usuarios = load(file)
```

```
def question(name,password,cat):
        return "error: archivo de usuarios no encontrado"

    user_found = False
    for usuario in usuarios:
        if usuario["name"] == name and usuario["password"] == password and usuario["active"] == 1:
            user_found = True
            puntuacion = 0

            todas_preguntas = []

            # Cargar preguntas de historia
            try:
                with open("question_history.txt", "r", encoding='utf-8') as file:
                    preguntas_historia = load(file)
                    for pregunta in preguntas_historia:
                        pregunta['categoria'] = 'Historia'
                        todas_preguntas.append(pregunta)
            except (FileNotFoundError, JSONDecodeError):
                print("No se pudo cargar el archivo de historia")

            # Cargar preguntas de ciencia
            try:
                with open("question_science.txt", "r", encoding='utf-8') as file:
                    preguntas_ciencia = load(file)
                    for pregunta in preguntas_ciencia:
                        pregunta['categoria'] = 'Ciencia'
                        todas_preguntas.append(pregunta)
            except (FileNotFoundError, JSONDecodeError):
                print("No se pudo cargar el archivo de ciencia")

            if not todas_preguntas:
```



```
def question(name,password,cat):
                return "error"

            # Mostrar las preguntas al usuario eligiendo categoría al azar
            while todas_preguntas:
                # Seleccionar una pregunta al azar
                pregunta = random.choice(todas_preguntas)

                print(f"\nCategoría: {pregunta['categoria']}")
                print("Pregunta:", pregunta["pregunta"])

                for opcion, texto in pregunta["opciones"].items():
                    print(f"{opcion}: {texto}")

                while True:
                    respuesta = input("Selecciona una opción (A, B, C, D) o presiona 'F' para salir del juego: "
                    if respuesta in ['A', 'B', 'C', 'D', 'F']:
                        break
                    print("Opción no válida. Inténtalo de nuevo.")

                if respuesta == 'F':
                    print("Juego terminado.")
                    break

                # Verificar la respuesta correcta
                if respuesta == pregunta["respuesta correcta"]:
                    print("Correcto")
                    puntuacion += 2
                else:
                    print("Incorrecto")
                    puntuacion -= 1
```



```
                print(f"Puntuación actual: {puntuacion}")

                # Remover la pregunta actual de la lista
                todas_preguntas.remove(pregunta)

            # Informarcuando haya respondido todas las preguntas
            if not todas_preguntas:
                print("\nHas respondido todas las preguntas.")
            print(f"Puntuación final: {puntuacion}")
            print("Preguntas completadas")

    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 sí 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:

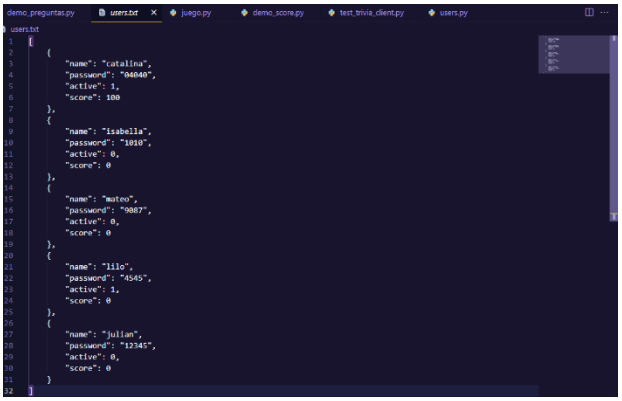


Fig. 2. Archivo (users.txt) que almacena al usuario.

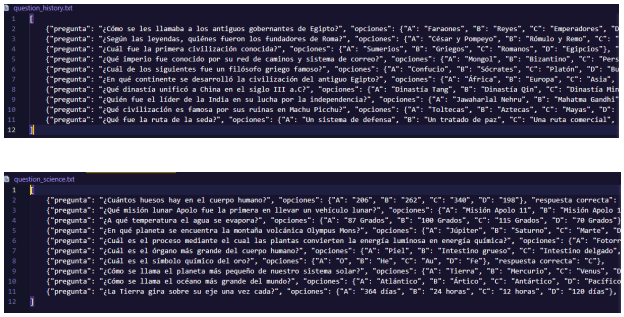


Fig. 2.1. Archivo con preguntas de historia (question history) y ciencia (question science).

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

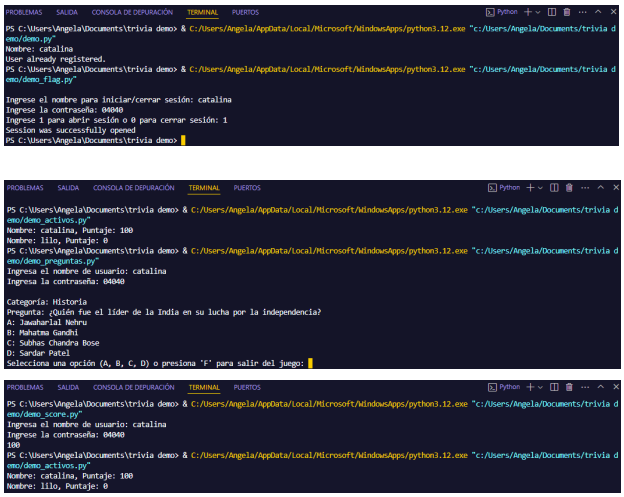


Fig. 3. Pruebas unitarias de cada función.

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*. .