Lab 2: Servidor de jugadores y preguntas

Nombres: Ever Hemerson Ceballos Rivera, Jimmy Sebastián Mora

Entregado: Julián Darío Barrero, Alexander López Parrado

Universidad del Quindío

**resumen.**

En este laboratorio aborda el desarrollo de un servidor para gestionar jugadores y preguntas, utilizando archivos y estructuras de programación en Python, para esto se suministró unos códigos base que no debe ser modificado **trivia\_server.py, trivia\_client.py, test\_trivia\_client.py,** y el objetivo es implementar funciones para registrar usuarios, actualizar los puntajes, verificar usuarios en línea y gestión de preguntas. Se evaluará la funcionalidad del servidor en un entorno de red local o en el mismo equipo, utilizando pruebas específicas implementando las funciones del código **users.py.**

***Palabras clave:***Servidor, Jugadores, Preguntas, Archivos, Gestión, Python.

1. **introducción.**

Para esta práctica de laboratorio se implementa la gestión de usuarios y preguntas en un servidor utilizando Python, el servidor permite registrar usuarios, gestionar sus sesiones, actualizar puntajes y proporcionar preguntas aleatorias a través de **test\_trivia\_client.py.** La correcta implementación de las funciones es esencial para el proyecto académico y estas se evalúa mediante pruebas de escritorio.

**Requisitos técnicos.**

* **Código bases:** trivia\_server.py, users.py, trivia\_client.py, test\_trivia\_client.py.
* **Módulo de Python:** requests.
* **Un equipo para pruebas.**

**Entrega.**

El informe y el código implementado se deben entregar a través de un repositorio en GitHub y compartirse en la plataforma Google Classroom.

1. **OBJETIVOS.**

**Objetivo General.**

Desarrollar un servidor en Python implementando funciones en el código base **users.py** para que el servidor gestione jugadores y preguntas mediante el uso de archivos, asegurando la correcta interacción entre clientes y servidor dentro de una red local.

**Objetivos Específicos.**

* Implementar funciones en Python para registrar nuevos jugadores, iniciar y cerrar sesiones, y verificar usuarios conectados utilizando archivos de texto.
* Desarrollar una interfaz que permita la interacción del jugador con las preguntas de trivia, asegurando la aleatoriedad de las mismas.
* Probar la correcta actualización de los puntajes en el servidor mediante la interacción con las respuestas proporcionadas por los usuarios.
* Evaluar el correcto funcionamiento del sistema a través de pruebas de escritorio que validen las interacciones entre el cliente y el servidor.
* Implementar un control de la lista de jugadores conectados en tiempo real para monitorear la actividad del juego.

1. **procedimiento y resultados.**

**PROCEDIMIENTOS Y PRUEBAS UNITARIAS.**

El sistema de gestión de usuarios y preguntas en Python permite registrar usuarios, iniciar y cerrar sesión, actualizar puntajes, consultar el puntaje actual, y listar usuarios conectados, todo gestionado mediante un archivo de texto. Las preguntas para el juego se obtienen de un archivo **preguntas.text,** se mezclan aleatoriamente, y se presentan al usuario, quien responde para acumular puntos por cada respuesta correcta. Las funciones implementadas abarcan desde el manejo de usuarios hasta la ejecución del juego trivia **“KingKizz”.** Esto se va a hacer con ayuda de **users.py y registerUseers.**

**CÓDIGO USER.PY**

from random import randint

import random

import requests

# Conjunto para llevar un registro de usuarios conectados

connected\_users = set()

# Función para registrar un nuevo usuario

def registerUser(name, password):

    name = name.strip().lower()

    password = password.strip()

    try:

        with open("usuarios.txt", "r") as file: #Se intenta abrir el archivo usuarios.txt en modo lectura r. Si el archivo no existe, se lanzará una excepción para más adelante

            for line in file: #Se itera sobre cada línea del archivo abierto.

                if line:  #verifica si la línea esta vacia

                    parts = line.split(",")  #Se divide la línea en partes utilizando la coma para separar

                    user = parts[0].strip().lower()  #se pasa el name a minusculas

                    if user == name:

                        return "ya registrado"

    except FileNotFoundError:

        open("usuarios.txt", "w").close()

    with open("usuarios.txt", "a") as file: #abre el archivo txt

        file.write(f"{name},{password},0,no\_conectado\n")

    return "registrado"

def openCloseSession(name, password, flag):

    name = name.strip().lower()

    password = password.strip()

    lines = [] #para actualizar las líneas del archivo

    user\_found = False #se inicia así para verificar si el usuario está en el txt

    with open("usuarios.txt", "r") as file:

        for line in file:

            if line:

                try:

                    user, passw, user\_score, status = line.split(",") #se divide en comas el user,passw,etc

                    if user == name and passw == password: #verifica que si el usuario y la contraseña digitada esta en el txt

                        user\_found = True #si se encontro

                        status = "conectado" if flag else "desconectado" #se actualiza el estado a true que es conenctado,si no lo es pasa lo contrario

                    lines.append(f"{user},{passw},{user\_score},{status}") #agrega el nuevo estado conectado

                except ValueError:

                    continue

    if user\_found: #procede a actualizar el archivo

        with open("usuarios.txt", "w") as file:#sobrescribe lo que ya existía

            file.write("\n".join(lines) + "\n")#se escribe en el archivo el contenido de lines

        return "seccion iniciada" if flag else "Sseccion cerrada"

    return "eror de contrasa o usuario"

def updateScore(name, password, new\_score):

    name = name.strip().lower()

    password = password.strip()

    updated = False

    lines = [] #almacena cualquier línea del archivo

    with open("usuarios.txt", "r") as file:

        for line in file:

            if line:

                try:

                    user, passw, user\_score, status = line.split(",")

                    if user.lower() == name and passw == password:

                        user\_score = new\_score   #se asigna el nuevo dato

                        updated = True #si fue actualizado

                    lines.append(f"{user},{passw},{user\_score},{status}") #se agrega a la actualización a line

                except ValueError:

                    continue

    if updated:

        with open("usuarios.txt", "w") as file: #sobrescribe el dato "w"

            file.write("\n".join(lines) + "\n")

        return "Actualizado"

    return "eror de contra o usuario"

def getScore(name, password):

    name = name.strip().lower()

    password = password.strip()

    with open("usuarios.txt", "r") as file:

        for line in file:

            line = line.strip()

            if line:

                try:

                    user, passw, user\_score, status = line.split(",")

                    if user == name and passw == password:

                        return user\_score

                except ValueError:

                    continue

    return "error de contra o usuario"

def usersList(name=None, password=None):

    connected\_users = [] #guarda el nombre de los usuarios

    try:

        with open("usuarios.txt", "r") as file:

            for line in file:

                if line:

                    try:

                        parts = line.split(",")

                        #Se Comprueba que hay 4 datos de largo(longitud), se comprueba que el 4 dato es igual conectado

                        if len(parts) >= 4 and parts[3].strip().lower() == "conectado":

                            connected\_users.append(parts[0].strip())

                    except IndexError:

                        continue

    except FileNotFoundError:

        return "usuaruio no registrado"

    if connected\_users:

        return f"Usuarios conectados: {', '.join(connected\_users)}"

    else:

        return "No hay usuarios conectados."

def getQuestion(url, name, password, category):

            response = requests.get(url + '/question', data=f'name={name}&password={password}&cat={category}')

            if response.status\_code != 200:

                print("Error al obtener la pregunta del servidor.")

                return "Error"   #Devolver un mensaje de error si la solicitud falla

            questions = response.content.decode('utf-8').splitlines()  #Divide la respuesta en líneas

            random.shuffle(questions)  # Mezcla las preguntas

            return questions  # Devolver las preguntas mezcladas

def question():

    total\_points = 0  #Iniciar acumulado

    asked\_questions = []  #guardar preguntas que salgan

    user\_answers = []  #guardar respuestas del usuario

    question\_list = []  #iniciar la lista de preguntas

    try:

        with open("preguntas.txt", "r", encoding="utf-8") as file:  #para que funcione con la codificacion de caracteres (utf-8)

            questions = file.readlines()  #Lee todas las líneas del archivo

            if questions:

                question\_buffer = []

                for line in questions:

                    line = line.strip()

                    if line == "":  #Si es una línea vacía, significa que terminamos una pregunta

                        if question\_buffer:

                            #agarra la primera pregunta random

                            question\_text = question\_buffer[0]

                            options = question\_buffer[1:5]

                            correct\_answer = question\_buffer[-1].split(":")[1].strip()

                            question\_list.append({

                                "question": question\_text,

                                "options": options,

                                "answer": correct\_answer

                            })

                            #limpiador de pregunta

                            question\_buffer = []

                    else:

                        question\_buffer.append(line)

                #Mezclador de preguntas

                random.shuffle(question\_list)

                if not question\_list:  #verifica si ya no hay preguntas en el archivo txt

                    print("No se encontraron preguntas en el archivo.")

                    return 0

                for question\_data in question\_list:

                    # muestra la pregunta

                    print(f"\nPregunta: {question\_data['question']}")

                    for option in question\_data['options']:

                        print(option)

                    user\_answer = input("Escribe tu respuesta (A, B, C o D): ").strip().upper()

                    #almacena la pregunta y respuesta del ususario

                    asked\_questions.append(question\_data['question'])

                    user\_answers.append(user\_answer)

                    #verifica si la respuesta es correcta

                    if user\_answer == question\_data['answer']:

                        total\_points += 1

                        print("¡Correcto!")

                    else:

                        print("Incorrecto.")

                    #Esto ya es interfaz del juego

                    #Pregunta si desea continuar

                    continue\_game = input("¿Quieres continuar? (s/n): ").strip().lower()

                    if continue\_game != 's':

                        break

                #Mostrar los resultados finales

                print("\n\*"+ "\n\*")

                print("\nFin del juego.")

                print(f"Tu puntaje final es:==>> {total\_points}")

                print("\nPreguntas y tus respuestas:")

                for i in range(len(asked\_questions)):

                    print(f"Pregunta: {asked\_questions[i]}")

                    print(f"Tu respuesta: {user\_answers[i]}")

                    #Muestra la respuesta correcta

                    print(f"Respuesta correcta: {question\_list[i]['answer']}\n")

                return total\_points

            else:

                print("No hay preguntas disponibles en el archivo.")

                return 0

    except FileNotFoundError:

        print("Error: El archivo de preguntas no existe.")

        return 0

**CÓDIGO REGISTERUSSER.PY:**

from users import registerUser, openCloseSession, updateScore, getScore, usersList,question

import random

def main():

    current\_user = None

    current\_password = None

    while True:

        print("\n1. Registrar usuario")

        print("2. Iniciar sesión")

        print("3. Salir")

        option = input("Selecciona una opción: ")

        if option == "1":

            name = input("Introduce el nombre de usuario: ")

            password = input("Introduce la contraseña: ")

            result = registerUser(name, password)

            print(result)

        elif option == "2":

            name = input("Introduce el nombre de usuario: ")

            password = input("Introduce la contraseña: ")

            session\_result = openCloseSession(name, password, True)

            if session\_result == "seccion iniciada":

                current\_user = name

                current\_password = password

                print(f"\n¡Bienvenido, {name}! {session\_result}")

                while True:

                    print("\nOpciones disponibles:")

                    print("1. Actualizar puntaje")

                    print("2. Ver puntaje actual")

                    print("3. Ver usuarios conectados")

                    print("4. Obtener pregunta aleatoria")

                    print("5. Cerrar sesión")

                    print("6. Volver al menú principal")

                    logged\_option = input("Selecciona una opción: ")

                    if logged\_option == "1":

                        new\_score = int(input("Introduce el nuevo puntaje: "))

                        result = updateScore(current\_user, current\_password, new\_score)

                        print(result)

                    elif logged\_option == "2":

                        score = getScore(current\_user, current\_password)

                        print("Tu puntaje es:", score)

                    elif logged\_option == "3":

                        connected\_users\_info = usersList()  #Ver usuarios conectados

                        print(connected\_users\_info)

                    elif logged\_option == '4':

                           question()

                    elif logged\_option == "5":

                        session\_result = openCloseSession(current\_user, current\_password, False)

                        current\_user = None

                        current\_password = None

                        print("Sesión cerrada.")

                        break  #Salir del menú de sesión

                    elif logged\_option == "6":

                        break  #Volver al menú principal

                    else:

                        print("Opción inválida. Inténtalo de nuevo.")

            else:

                print(session\_result)

        elif option == "3":

            print("Saliendo del programa...")

            break

        else:

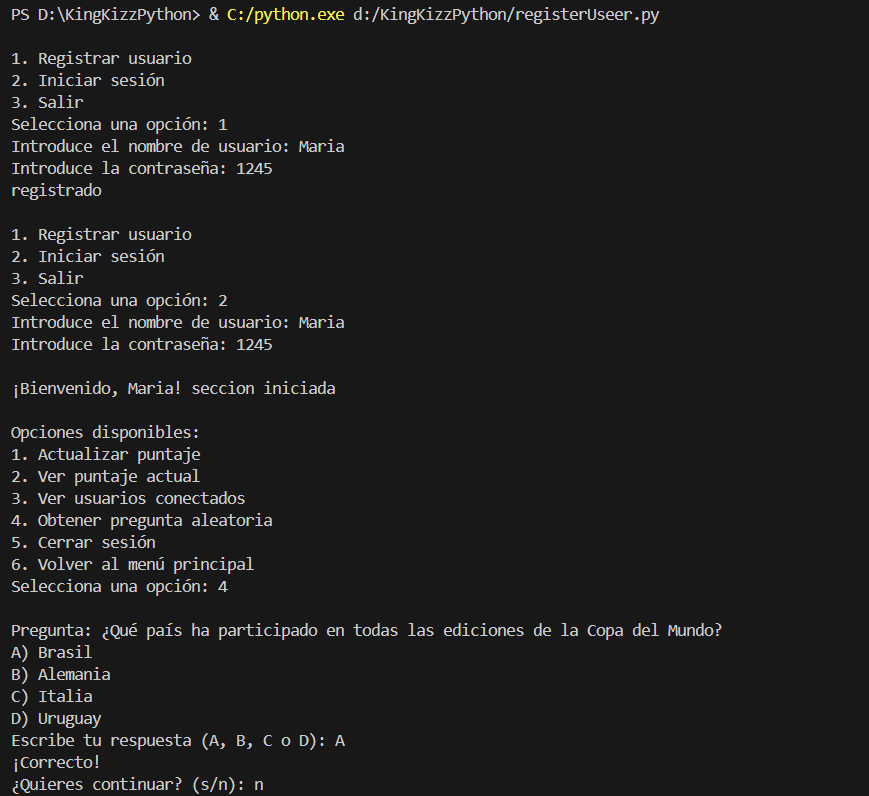
            print("Opción inválida. Inténtalo de nuevo.")

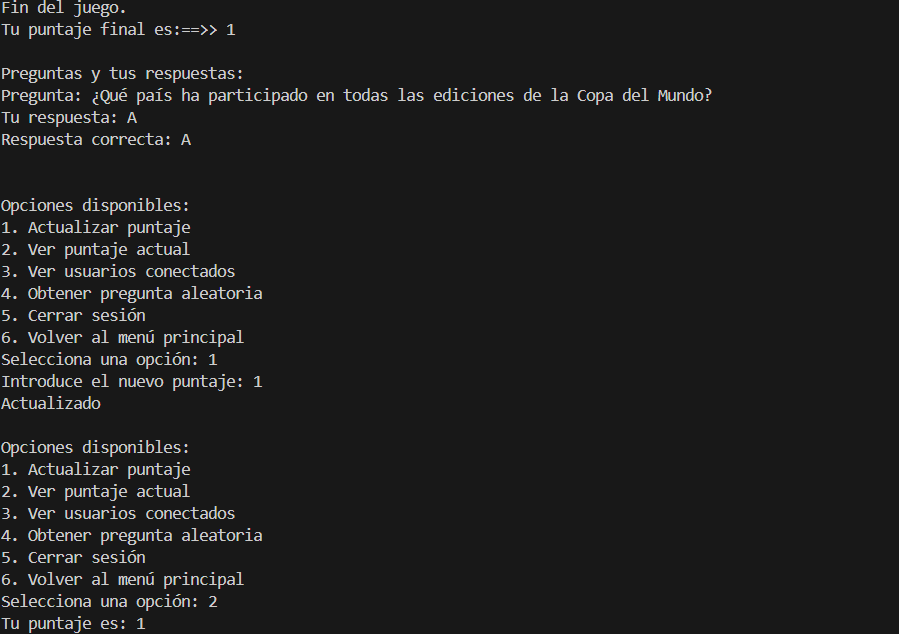
if \_\_name\_\_ == "\_\_main\_\_":

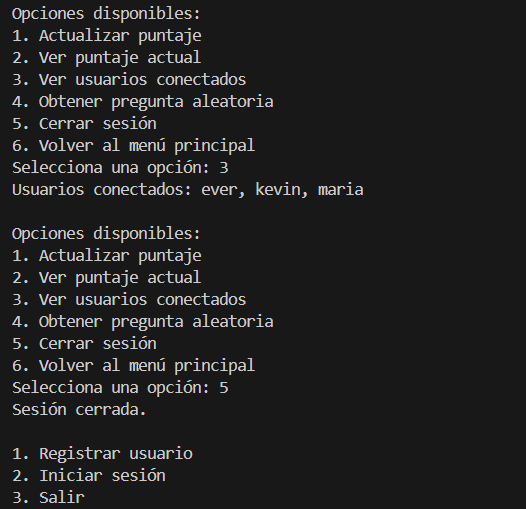
    main()

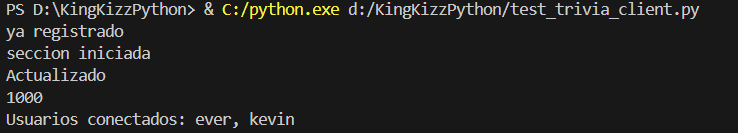
**RESULTADOS:**

Durante las pruebas, los usuarios se registraron exitosamente, validando que el sistema no permite registros duplicados, las sesiones de usuario se abrieron y cerraron correctamente, reflejando el estado de conexión en el archivo, se actualizaron los puntajes de acuerdo con las respuestas correctas e incorrectas en el juego y los usuarios pudieron consultar su puntaje actual sin problemas. La lista de usuarios conectados mostró los usuarios activos, y las preguntas del juego fueron presentadas de manera aleatoria, con un puntaje final calculado con precisión. Para las pruebas unitarias se hizo desde una nueva terminal, ejecutando **trivia\_server.py,** por consiguiente, se abre una nueva terminal y se ejecuta **registerUseer.py** para hacer el registro, inicio de sesión, actualización del puntaje y verificación de éste, una pregunta aleatoria y cierre de sesión. Finalmente se vuelve a ejecutar en una nueva terminal **trivia\_server.py,** junto con **test\_trivia\_client.py** para comprobar el cliente trivia, verificando la conexión, el registro de usuarios y la lista de usuarios activos.









1. **PRUEBAS DE ESCRITORIO:** Para las pruebas de escritorio se realizó un análisis exhaustivo del código, evaluando detalladamente el funcionamiento de cada una de las funciones presentes en el archivo **users.py**; además, se complementó el proceso con un código adicional desarrollado en **registerUser.py**, el cuál apoya las funciones del archivo **users.py**, ambos archivos fueron probados minuciosamente, y los resultados demostraron un rendimiento sólido y una correcta interacción entre las funcionalidades principales para el sistema de gestión de usuarios y preguntas.

**PRUEBAS CON USER.PY Y REGISTERUSSER.PY**

**CÓDIGO REGISTERUSSER:**

from users import registerUser, openCloseSession, updateScore, getScore, usersList,question

import random

def main():

    current\_user = None

    current\_password = None

    while True:

        print("\n1. Registrar usuario")

        print("2. Iniciar sesión")

        print("3. Salir")

        option = input("Selecciona una opción: ")

        if option == "1":

            name = input("Introduce el nombre de usuario: ")

            password = input("Introduce la contraseña: ")

            result = registerUser(name, password)

            print(result)

        elif option == "2":

            name = input("Introduce el nombre de usuario: ")

            password = input("Introduce la contraseña: ")

            session\_result = openCloseSession(name, password, True)

            if session\_result == "seccion iniciada":

                current\_user = name

                current\_password = password

                print(f"\n¡Bienvenido, {name}! {session\_result}")

                while True:

                    print("\nOpciones disponibles:")

                    print("1. Actualizar puntaje")

                    print("2. Ver puntaje actual")

                    print("3. Ver usuarios conectados")

                    print("4. Obtener pregunta aleatoria")

                    print("5. Cerrar sesión")

                    print("6. Volver al menú principal")

                    logged\_option = input("Selecciona una opción: ")

                    if logged\_option == "1":

                        new\_score = int(input("Introduce el nuevo puntaje: "))

                        result = updateScore(current\_user, current\_password, new\_score)

                        print(result)

                    elif logged\_option == "2":

                        score = getScore(current\_user, current\_password)

                        print("Tu puntaje es:", score)

                    elif logged\_option == "3":

                        connected\_users\_info = usersList()  #Ver usuarios conectados

                        print(connected\_users\_info)

                    elif logged\_option == '4':

                           question()

                    elif logged\_option == "5":

                        session\_result = openCloseSession(current\_user, current\_password, False)

                        current\_user = None

                        current\_password = None

                        print("Sesión cerrada.")

                        break  #Salir del menú de sesión

                    elif logged\_option == "6":

                        break  #Volver al menú principal

                    else:

                        print("Opción inválida. Inténtalo de nuevo.")

            else:

                print(session\_result)

        elif option == "3":

            print("Saliendo del programa...")

            break

        else:

            print("Opción inválida. Inténtalo de nuevo.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. **Registro de usuarios (registerUser)**

* Esta función permite registrar a un nuevo usuario en el archivo **usuarios.txt.**
* **Entrada:** recibe dos parámetros: **name** (nombre del usuario) y **password** (contraseña).
* **Proceso:**
* El nombre y la contraseña se limpian de espacios y se convierten a minúsculas.
* Se abre el archivo **usuarios.txt** en modo de lectura. Si no existe, se crea uno vacío.
* Se busca si el usuario ya está registrado (buscando el nombre en el archivo).
* Si el usuario ya está registrado, la función retorna **"ya registrado".**
* Si no está registrado, se abre el archivo en modo de escritura, se añade la nueva línea con el usuario, contraseña, puntaje **(inicialmente 0),** y estado **"no conectado".**
* **Salida:** devuelve **"registrado"** si el usuario se añade correctamente, o **"ya registrado"** si el usuario ya existe.

1. **Iniciar o cerrar sesión (openCloseSession)**

Esta función permite abrir o cerrar la sesión de un usuario.

**Entrada:** tres parámetros:

* **name** (nombre del usuario)
* **password** (contraseña)
* **flag** (booleano, donde True es para iniciar sesión y False para cerrarla).

**Proceso:**

* Se limpia el nombre y la contraseña de espacios y se convierten a minúsculas.
* El archivo **usuarios.txt** se abre en modo de lectura para verificar si el usuario y contraseña son correctos.
* Si se encuentra el usuario, se cambia su estado a **"conectado"** si el flag es True, o **"no\_conectado"** si es **False.**
* El archivo se vuelve a escribir con las líneas actualizadas.
* **Salida:** retorna **"sesión iniciada"** o **"sesión cerrada",** dependiendo de la operación realizada. Si el usuario o contraseña no coinciden, retorna **"error de contraseña o usuario".**

1. **Actualizar el puntaje (updateScore)**

* Esta función permite actualizar el puntaje de un usuario.
* **Entrada:** tres parámetros: **name, password, y new\_score (nuevo puntaje).**
* **Proceso:**
* Se limpia el nombre y la contraseña de espacios y se convierten a minúsculas.
* Se abre el archivo usuarios.txt para encontrar al usuario.
* Si se encuentra, se actualiza el puntaje con el nuevo valor proporcionado y se reescribe el archivo.
* **Salida:** devuelve **"Actualizado"** si el puntaje se actualiza correctamente, o **"error de contraseña o usuario"** si no coincide.

1. **Obtener puntaje (getScore)**

* Esta función permite obtener el puntaje actual de un usuario.
* **Entrada:** dos parámetros: **name y password.**
* **Proceso:**
* El archivo se abre para leer y buscar al usuario.
* Si el usuario y la contraseña son correctos, devuelve el puntaje.
* **Salida:** devuelve el puntaje del usuario si lo encuentra, o **"error de contraseña o usuario"** si no lo encuentra.

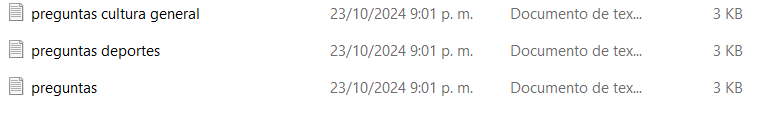
1. **Obtener lista de usuarios conectados (usersList)**

* Esta función obtiene una lista de los usuarios que están **"conectados".**
* **Entrada:** Ninguna (los parámetros son opcionales, pero no se usan en este caso).
* **Proceso:**
* Se abre el archivo **usuarios.txt** y se busca a los usuarios cuyo estado sea "conectado".
* **Salida:** devuelve una lista de usuarios conectados o **"No hay usuarios conectados"** si no hay ninguno.

1. **Lectura de archivo**: Se lee el archivo **preguntas.txt** para extraer las preguntas, opciones y respuestas correctas.

* **Presentación de preguntas**:
* Se muestra una pregunta con 4 opciones.
* El usuario elige una respuesta (A, B, C o D).
* **Verificación de respuestas**:
* Se compara la respuesta del usuario con la respuesta correcta.
* Si es correcta, se incrementa el puntaje.
* **Control del juego**:
* El usuario decide si continúa respondiendo preguntas o si termina el juego.
* **Resultados**:
* Al final del juego, se muestra el puntaje total.
* Se presentan las preguntas respondidas, las respuestas del usuario y las respuestas correctas.

**NOTA:** Para los archivos de las preguntas se utilizó un solo archivo debido a que consume 3KB y si utilizo dos o más, cada archivo marca 3 KB y así ocupa más espacio en la memoria.



**RESULTADOS PRUEBAS DE ESCRITORIO**

**1. Obtener lista de usuarios conectados (usersList)**

* **registerUser** ("Mariana", "6784"):
* Busca en **usuarios.txt.** Como el archivo está vacío, se añade:

*mariana,6784,5, desconectado*

* **Salida:** "registrado".

**2. Iniciar sesión con** "juan" y "1234"

* **openCloseSession** ("Ever", "1245", True):
* Se encuentra al usuario y se actualiza su estado a "conectado":

*ever,1245,0, conectado*

* **Salida:** "sesión iniciada".

**3. Actualizar el puntaje de "María" a 1**

* updateScore ("María", "1245", 1):
* Se encuentra al usuario y se actualiza su puntaje:

*maría,1245,1, desconectado*

* **Salida:** "Actualizado".

**4. Obtener lista de usuarios conectados o desconectados**

* usersList ():
* Encuentra a "Kevin" con estado "conectado" y a “María” con estado “desconectado”
* **Salida:** "Usuarios conectados: Kevin"

*kevin, pass,1000, conectado*

*maría,1245,1, desconectado*

5. **Resultado de prueba con las preguntas:**

* **Primera pregunta**:
* Pregunta: "¿Qué tratado puso fin a la Guerra de los Cien Años entre Inglaterra y Francia?”
* Respuesta del usuario: **B** (correcta).
* **Puntaje acumulado**: 1.

1. **Segunda pregunta**:

* Pregunta: “¿Qué país ganó la primera Copa del Mundo de Rugby en 1987?”
* Respuesta del usuario: **A** (incorrecta).
* **Puntaje acumulado**: 1.
* **Fin del juego**:
* **Puntaje final**: 1.
* **Preguntas y respuestas**:
* "¿Qué tratado puso fin a la Guerra de los Cien Años entre Inglaterra y Francia?”

Respuesta correcta: C.

* “¿Qué país ganó la primera Copa del Mundo de Rugby en 1987?”

Respuesta correcta: B.

1. **CONCLUSIONES**

* El sistema permitió un manejo adecuado de registros, sesiones y usuarios conectados mediante archivos de texto, funcionando correctamente en las pruebas.
* Los puntajes de los jugadores se actualizaron de manera correcta según sus respuestas, sin embargo, los jugadores manejan su puntaje de forma manual, actualizando su puntaje según sus respuestas contestadas.
* Se logró una presentación aleatoria de preguntas, lo que mejoró la dinámica y mantuvo el interés de los jugadores.
* Las pruebas de escritorio validaron que el sistema funciona adecuadamente en la interacción entre cliente y servidor.

1. **REFERENCIAS.**

* *Requests*. (s/f). PyPI. Recuperado el 24 de octubre de 2024, de <https://pypi.org/project/requests/>
* López-Parrado, A. (s/f). *Lab2*.