LABORATORIO 2 PROGRAMACION

CARLOS MANUEL DIAZ, ALVARO ANDRES GUZMAN

Resumen— Esta práctica de laboratorio tiene como propósito el desarrollo de un servidor mediante creación de funciones y prueba de ellas que hacen uso de archivos para la gestión de los jugadores y las preguntas, así como las diferentes estructuras estudiadas en la materia, se hace la creación de forma adecuada de las funciones implementando archivos para lograr un correcto funcionamiento del programa.

Palabras clave—CREACION, FUNCIONES, ARCHIVOS, PROGRAMA, FUNCIONAMIENTO, ESTRUCTURAS.

I.INTRODUCCIÓN

Desarrollar un servidor que tiene como propósito la gestión de preguntas y respuestas haciendo el uso de archivos para almacenar la información del usuario como el nombre, contraseña y archivos para el almacenamiento de las preguntas definidas en el programa , se crean funciones para controlar el inicio y cierre de sesión, actualizar puntaje, lista de usuarios que tienen iniciado sesión, función para las preguntas todas estas funciones se han creado en el código users.py y se hace pruebas de escritorio para el correcto funcionamiento de cada una de ellas para garantizar que el servidor funcione de manera correcta y eficiente.

OBJETIVOS GENERAL

Desarrollar un servidor cuya función sea gestionar preguntas y respuestas e información de jugadores que hayan iniciado sesión en el juego.

ESPECÍFICOS

- ***Crear funciones para gestionar la información***
- ***Hacer el uso de archivos dentro de las funciones para almacenar la información del jugador***
- ***Realizar pruebas de escritorio para verificar correcto funcionamiento de las funciones propuestas***

II. INSTRUMENTOS

II. Computador
II. Entornos de desarrollo como visual Studio
II. PYTHON
II. Lenguaje C++
III. Draw.io

III. **MÉTODOS**.

- La práctica de laboratorio contempla la creación y prueba de funciones que hacen uso de archivos para la gestión de los jugadores y las preguntas, así como las demás estructuras de programación y tipos de datos estudiados hasta el momento.

El servidor tendrá una serie de funciones para su funcionamiento las cuales son:

- Recibe nombre y contraseña del usuario y lo registra, si el usuario ya está registrado imprimirá el mensaje “Usuario ya registrado”.
- Inicio y cierre de sesión
- El usuario podrá salir del programa en el momento que lo desee
- El programa debe ser legible, correcto uso de recursos.
- Implementar estructuras vistas en clase y elegir la que mejor se adapte al programa es decir poco uso de memoria y legibilidad

Métodos:

1. **Registro de Usuarios**:
    - Se implementó un sistema de registro de usuarios en el cual los nombres y contraseñas se almacenan en un archivo de texto (usuarios.txt).
    - La función registerUser () verifica si un usuario ya está registrado. Si no lo está, se agrega al archivo junto con su contraseña, un puntaje inicial de 0.
2. **Iniciar/Cerrar Sesión**:
    - La función openCloseSession() permite que los usuarios puedan iniciar o cerrar sesión. Se verifica si el nombre de usuario y la contraseña coinciden con los demás datos almacenados en el archivo.
    - Si el usuario y contraseña son correctas, se actualiza el estado de conexión del usuario a "conectado" o "desconectado", dependiendo de la acción solicitada.
3. **Actualización y Consulta de Puntajes**:
    - La función updateScore() se utiliza para actualizar el puntaje de un usuario tras finalizar una partida del juego, mientras que getScore() permite obtener el puntaje actual del usuario.
    - Ambas funciones realizan la búsqueda del usuario por su nombre y contraseña en el archivo de texto y modifican o devuelven la información correspondiente.
4. **Listado de Usuarios Conectados**:
    - La función usersList() muestra una lista de todos los usuarios actualmente conectados, es decir, aquellos cuyo estado en el archivo es "conectado".
5. **Generación de Preguntas**:
    - Se utiliza una función llamada getQuestion() para obtener preguntas de un servidor.
    - Las preguntas locales se cargan desde archivos de texto (preguntas_categoria_1.txt y preguntas_categoria_2.txt), que contienen tanto las preguntas como las opciones de respuesta y la respuesta correcta.
6. **Desarrollo del Juego de Preguntas**:
    - Se implementó una función llamada question() para gestionar el juego. Las preguntas se seleccionan aleatoriamente de las categorías cargadas, y se muestran al usuario junto con las opciones de respuesta.
    - El usuario ingresa su respuesta, y el programa verifica si es correcta o incorrecta. El puntaje se incrementa en caso de una respuesta correcta.
    - El juego continúa hasta que el usuario decida finalizar la partida.
    - Al final del juego, se muestran los resultados, incluyendo el puntaje total, el número de respuestas correctas e incorrectas.

Para realizar esta acción es necesario el código users y test client, ya que el users contiene la lógica necesaria para realizar gestionar usuarios y preguntas, mientras que el test client utiliza las funciones definidas en el users, así nos permite separar la lógica de manejo de usuarios, preguntas y puntuaciones de la interfaz de usuario, facilitando la organización del código

```python
import random
import requests


connected_users = {}

# Función para registrar un nuevo usuario
def registerUser(name, password):
    name = name.strip().lower()
    password = password.strip()

    try:
        file = open("usuarios.txt", "r")
        for line in file:
            if line.strip() and line.split(",")[0].strip().lower() == name:
                file.close()
                return "ya registrado"
        file.close()
    except FileNotFoundError:
        open("usuarios.txt", "w").close()

    file = open("usuarios.txt", "a")
    file.write(f"{name},{password},0,no_conectado\n")
    file.close()
    return "registrado"

def openCloseSession(name, password, flag):
    name = name.strip().lower()
    password = password.strip()
    lines, user_found = [], False

    file = open("usuarios.txt", "r")
    for line in file:
        if line.strip():
            user, passw, user_score, status = line.split(",")
            if user == name and passw == password:
                user_found = True
                user_found = True
                status = "conectado" if flag else "desconectado"
            lines.append(f"{user},{passw},{user_score},{status}")
    file.close()

    if user_found:
        file = open("usuarios.txt", "w")
        file.write("\n".join(lines) + "\n")
        file.close()
        return "sesión iniciada" if flag else "sesión cerrada"

    return "error"

def updateScore(name, password, new_score):
    name = name.strip().lower()
    password = password.strip()
    lines, updated = [], False

    file = open("usuarios.txt", "r")
    for line in file:
        if line.strip():
            user, passw, user_score, status = line.split(",")
            if user == name and passw == password:
                user_score = new_score
                updated = True
            lines.append(f"{user},{passw},{user_score},{status}")
    file.close()

    if updated:
        file = open("usuarios.txt", "w")
        file.write("\n".join(lines) + "\n")
        file.close()
        return "actualizado"

    return "error de usuario o contraseña"
```

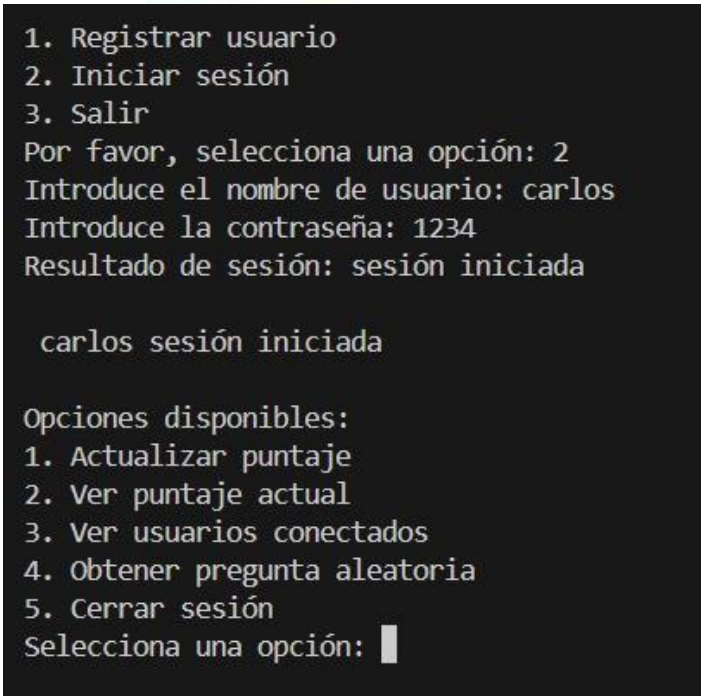
```python
    return "error de usuario o contraseña"

def getScore(name, password):
    name = name.strip().lower()
    password = password.strip()

    file = open("usuarios.txt", "r")
    for line in file:
        if line.strip():
            user, passw, user_score, status = line.split(",")
            if user == name and passw == password:
                file.close()
                return user_score
    file.close()
    return "error de usuario o contraseña"

def usersList():
    connected_users = []

    try:
        file = open("usuarios.txt", "r")
        for line in file:
            if line.strip():
                parts = line.split(",")
                if len(parts) >= 4 and parts[3].strip().lower() == "conectado":
                    connected_users.append(parts[0].strip())
        file.close()
    except FileNotFoundError:
        return "usuario no registrado"

    return f"Usuarios conectados: {', '.join(connected_users)}" if connected_users else "No hay usuarios conectados."

def question():
    total_points = 0
    asked_questions, user_answers = [], []
```

```python
def question():
    total_points = 0
    asked_questions, user_answers = [], []
    question_list = []

    for selected_file in ["preguntas_categoria_1.txt", "preguntas_categoria_2.txt"]:
        try:
            file = open(selected_file, "r", encoding="utf-8")
            question_buffer = []
            for line in file:
                line = line.strip()
                if line:
                    question_buffer.append(line)
                else:
                    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})
                    question_buffer = []
            file.close()
        except FileNotFoundError:
            print(f"Error: El archivo {selected_file} no existe.")
            return 0

    if not question_list:
        print("No se encontraron preguntas en los archivos.")
        return 0

    while question_list:
        question_data = random.choice(question_list)
        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()
        while user_answer not in ['A', 'B', 'C', 'D']:
            print("Por favor, ingresa una opción válida (A, B, C o D).")
            user_answer = input("Escribe tu respuesta (A, B, C o D): ").strip().upper()

        asked_questions.append(question_data['question'])
        user_answers.append(user_answer)

        if user_answer == question_data['answer']:
            total_points += 1
            print("¡Correcto!")
        else:
            print("Incorrecto.")

        question_list.remove(question_data)

        if input("¿Quieres continuar? (si/no): ").strip().lower() != 'si':
            break

    total_questions = len(asked_questions)
    correct_answers = total_points
    incorrect_answers = total_questions - correct_answers

    print("\nFin del juego.")
    print(f"Tu puntaje total es: {total_points}")
    print(f"Preguntas correctas: {correct_answers}")
    print(f"Preguntas incorrectas: {incorrect_answers}")

    return total_points
```

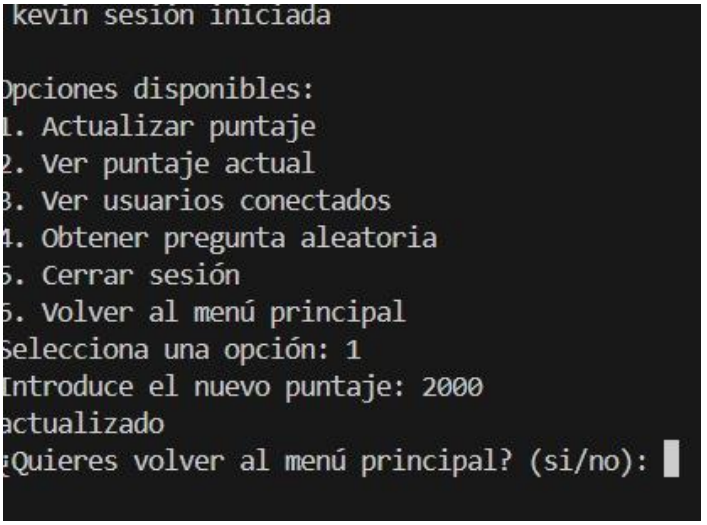
Resultados

```
1. Registrar usuario
2. Iniciar sesión
3. Salir
Por favor, selecciona una opción: 1
Introduce el nombre de usuario: Carlos
Introduce la contraseña: 1234
registrado
```

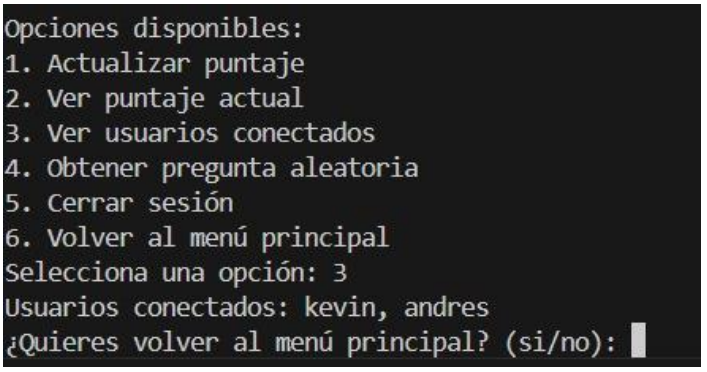
El sistema presenta un menú con las opciones que puede elegir el usuario, se selecciona la opción 1 para registrar nuevo usuario, el sistema solicita el nombre de usuario y contraseña, se verifica si el usuario ya esta registrado, como no está registrado, se añade un nuevo usuario, el sistema retorna el mensaje “registrado” para indicar que el usuario se añadió correctamente



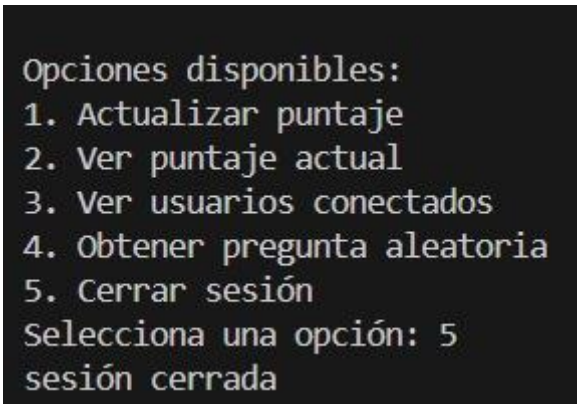
En este caso se presenta el menú de opciones, se selecciona la opción 2 para que luego el sistema solicite el nombre y contraseña, el sistema busca en el archivo, como el usuario y contraseña coinciden cambia el estado a conectado y devuelve el mensaje sesión iniciada, después de esto despliega las demás opciones



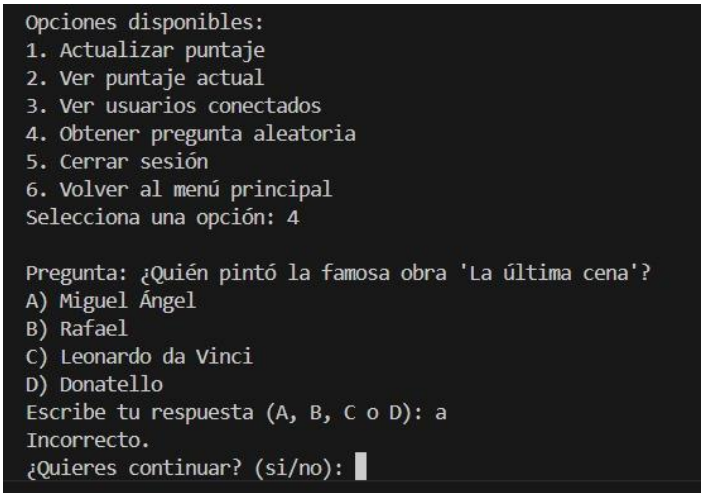
En este caso la persona desea actualizar el puntaje, el sistema le solicita que ingrese el nuevo puntaje y actualiza el puntaje de usuario en el archivo y el sistema envía el mensaje "puntaje actualizado"



Aquí, el usuario selecciona la opción 3 para ver los usuarios conectados en este caso muestra en un mensaje afirmando que Kevin y Andrés están conectados en ese momento



El usuario selecciona la opción 5 para cerrar sesión el sistema actualiza el usuario a desconectado y devuelve el mensaje "sesión cerrada" indicando que se cerró sesión correctamente



En esta prueba el usuario selecciona la opción 4 "obtener

pregunta aleatoria", el sistema carga la pregunta disponible en el archivo, luego solicita al usuario ingresar la respuesta correcta, luego de evaluarla determina si fue correcta o incorrecta y le pregunta al usuario si desea continuar.

Códigos trivia_client.py y trivia_server.py

Estos códigos cuentan con un diseño especifico, el cual garantiza la operabilidad interna entre el cliente y el servidor, si se llega a modificar puede ocasionar errores o perdidas de conexión, poniendo en riesgo la operabilidad del programa, estos 2 códigos no se pueden modificar bajo ninguna circunstancia

Trivia_server.py

```python
import argparse
from urllib.parse import urlparse
from urllib.parse import parse_qs
from http.server import BaseHTTPRequestHandler, HTTPServer
from json import dumps
import users

class Server(HTTPServer):
    def __init__(self, address, request_handler):
        super().__init__(address, request_handler)


class RequestHandler(BaseHTTPRequestHandler):
    def __init__(self, request, client_address, server_class):
        self.server_class = server_class
        super().__init__(request, client_address, server_class)

    def do_GET(self):
        resource=urlparse(self.path).path
        match resource:

            case '/score':
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=str(users.getScore(params_dict['name'][0],params_dict['password'][0]))

            case '/list':
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=str(users.userslist(params_dict['name'][0],params_dict['password'][0]))

            case '/question':
                content_len = int(self.headers.get('Content-Length'))
            post_body = self.rfile.read(content_len)
            params_dict=parse_qs(post_body.decode(), strict_parsing=True)
            response=str(users.question(params_dict['name'][0],params_dict['password'][0],params_dict['cat'][0]))


        #     case '/list':



        self.send_response(200)
        self.send_header("Content-type", "application/json")
        self.send_header("Content-Length", str(len(response)))
        self.end_headers()
        self.wfile.write(str(response).encode('utf8'))
        self.wfile.flush()



    def do_PUT(self):
        resource=urlparse(self.path).path
        print(resource)
        match resource:
            case "/score":
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=users.updateScore(params_dict['name'][0],params_dict['password'][0],params_dict['score'][0])
            case "/login":
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=users.openCloseSession(params_dict['name'][0],params_dict['password'][0],True)
            case "/logout":
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=users.openCloseSession(params_dict['name'][0],params_dict['password'][0],False)


        self.send_response(200)
        self.send_header("Content-type", "application/json")
        self.send_header("Content-Length", str(len(response)))
        self.end_headers()
        self.wfile.write(str(response).encode('utf8'))
        self.wfile.flush()



    def do_POST(self):
        resource=urlparse(self.path).path
        print(resource)
        match resource:
            case "/register":
                content_len = int(self.headers.get('Content-Length'))
                post_body = self.rfile.read(content_len)
                params_dict=parse_qs(post_body.decode(), strict_parsing=True)
                response=users.registerUser(params_dict['name'][0],params_dict['password'][0])




        self.send_response(200)
        self.send_header("Content-type", "application/json")
        self.send_header("Content-Length", str(len(response)))
        self.end_headers()
            self.wfile.write(str(response).encode('utf8'))
            self.wfile.flush()

    def start_server(addr, port, server_class=Server, handler_class=RequestHandler):
        server_address = (addr, port)
        http_server = server_class(server_address, handler_class)
        print(f"Starting server on {addr}:{port}")
        http_server.serve_forever()


    def main():
        parser = argparse.ArgumentParser(description="Run a simple HTTP server.")
        parser.add_argument(
            "-l",
            "--listen",
            default="0.0.0.0",
            help="Specify the IP address which server should listen",
        )
        parser.add_argument(
            "-p",
            "--port",
            type=int,
            default=80,
            help="Specify the port which server should listen",
        )
        args = parser.parse_args()
        start_server(addr=args.listen, port=args.port)


    if __name__ == "__main__":
        main()
```

Trivia_client.py

```python
# Instalar el módulo requests haciendo desde una terminal: pip install requests
import requests


# Función para registrar usuario
def registerUser(url,name,password):
    response=requests.post(url+'/register',data=f'name={name}&password={password}')
    return response.content.decode('utf-8')

# Funciónn para abrir una sesión
def openSession(url,name,password):
    response=requests.put(url+'/login',data=f'name={name}&password={password}')
    return response.content.decode('utf-8')

# Función para cerrar una sesión
def closeSession(url,name,password):
    response=requests.put(url+'/logout',data=f'name={name}&password={password}')
    return response.content.decode('utf-8')

# Función para actualizar el puntaje
def updateScore(url,name,password,score):
    response=requests.put(url+'/score',f'name={name}&password={password}&score={score}')
    return response.content.decode('utf-8')

# Función para obtener el puntaje
def getScore(url,name,password):
    response=requests.get(url+'/score',data=f'name={name}&password={password}')
    return response.content.decode('utf-8')

# Función para obtener la lista de usuarios conectados
def getList(url,name,password):
    response=requests.get(url+'/list',data=f'name={name}&password={password}')
    return response.content.decode('utf-8')

# Función para obtener una pregunta aleatoria
def getQuestion(url,name,password,cat):
    response=requests.get(url+'/question',data=f'name={name}&password={password}&cat={cat}')
```

```
def getQuestion(url,name,password,cat):
    response=requests.get(url+'/question',data=f'name={name}&password={password}&cat={cat}')
    return response.content.decode('utf-8')
```

CONCLUSIONES

- La separación de los códigos (como users.py y test client.py) mejora la manejabilidad y legibilidad del código logrando reutilizar funciones en diferentes circunstancias sin necesidad de duplicar el código
- El sistema guía al usuario con opciones claras facilitando el uso de las funciones como lo es registrar, iniciar sección, actualizar puntaje entre otras
- El programa logra identificar bien errores, nombres de usuarios repetidos evitando que el programa se detenga
- Al usar el archivo de texto para guardar usuarios y puntuaciones se simplifica la forma en la que se guarda la información entre sesiones

VI. REFERENCIAS

[1] "Manejo de archivos en Python," YouTube, [Enlace: https://www.youtube.com/watch?v=go52Qbt9Gyk]. Accedido el 18 de octubre de 2024.

[2] "Archivos en Python ," PythonEs.net. [Enlace: https://pythones.net/archivos-en-python-3/]. Accedido el 22 de octubre de 2024.

[3] "Crear archivos en PYTHON," YouTube, [Enlace: https://www.youtube.com/watch?v=QjrAEJZVuJw]. Accedido el 15 de octubre de 2024.

septiembre de 2024, de https://www.youtube.com/watch?v=pFEs4dO63y4&t=462s

- "Juego de preguntas en Python," YouTube, 2024. [Online]. Available: https://www.youtube.com/watch?v=aBL2lhuU2fI. [Accessed: 24-Sep-2024].

Enlace GitHub

https://github.com/AG-Program/laboratorio-2-programacion-.git