Reporte práctica 1

Integrantes del equipo:
- Alexia Rodríguez Miranda
- Mariana Nava Córdova

Protocolo de comunicación:

Mensajes enviados del cliente al servidor, con sus posibles contenidos:

- Para Registrarse:
  {
    username: nombre de usuario
    password: clave de acceso
    tipo: 0 Indica que el mensaje es para registrarse
  }

- Para hacer sign_in:
  {
    cookie_id: Sería la cookie que regresa el servidor para autenticar al cliente
    tipo: 1 Indica que el mensaje es para hacer sign in
  }

- Para hacer una jugada:
  {
    letra: Letra que el usuario quiere probar en la palabra
    tipo: 2 Indica que el mensaje es para ver si la letra se encuentra en la palabra
  }

Para la parte de los sockets, mientras la sesión del usuario identificado con la ip xxx.xxx.xxx.xxx esté activa entonces mantendremos los registros de los jugadores que inician sesión.

Mensajes enviados del servidor al cliente, con sus posibles contenidos:

{
  "juego": {
  "jugador1": {
    "username": username,
    "estado": ganador/perdedor/continua,
    "progreso": La palabra adivinada hasta el momento,
    "errores": Número que representa la cantidad de errores que lleva,donde los posibles valores van del 0 al 6.
  },
  "jugador2": {
    "username": username,
    "estado": ganador/perdedor/continua,
    "progreso": La palabra adivinada hasta el momento,
    "errores": Número que representa la cantidad de errores que lleva, donde los posibles valores van del 0 al 6.
  },

```
   "siguiente_jugador": El jugador con el tiro siguiente,
   "estado_partida": finalizada, continua o no disponible
   }
}
```

Ejemplo:

```
{
   "juego": {
   "jugador1": {
      "username": "juan",
      "estado": "continua",
      "progreso": "p _ a _ _",
      "errores": 3
   },
   "jugador2": {
      "username": "alicia",
      "estado": "continua",
      "progreso": "_ i a _ o",
      "errores": 2
   },
   "siguiente_jugador": 1,
   "estado_partida": "continua"
   }
}
```

Autenticación y persistencia de la sesión

Para autenticar e iniciar sesión se usan las funciones ‘sign_up’ y ‘sign_in’ del archivo ‘api.py’.

Para ‘sign_up’ se revisa que el usuario no exista previamente antes de registrarlo, se le asigna un identificador y se genera una cookie, la cual se le envía de regreso al cliente para inicio de sesión en el futuro.

Para ‘sign_in’ revisamos que la cookie enviada por el jugador esté almacenada en el servidor, de lo contrario enviamos un mensaje de error “Not Found User”. Si está guardado, entonces generamos una nueva jugada y lo agregamos como jugador de ella, enviando un mensaje de confirmación “Successful login”.

Manejo del estado de la partida y cómo se sincronizan los clientes con el servidor

Intentamos mantener la persistencia del estado de la partida teniendo siempre un archivo de texto desde donde podemos leer la información de la partida hasta el momento. Además también agregamos funciones y mecanismos para leer y recuperar la información en caso de que sea necesario, manteniendo el estado lógico de la jugada.

En cuanto a la sincronización de los clientes con el servidor, lo que hacemos es actualizar el estado cada vez que uno de los jugadores hace un movimiento o ingresa datos para inicio de sesión. Se tiene guardada la información de ambos jugadores en un mismo archivo (‘solucion.txt’), por lo que basta modificar este archivo para actualizar la información del

servidor. Además, por cada acción de una petición se envía dicho mensaje de regreso al cliente después de actualizar.

Demostración de ejecución:
Primero ejecutamos en una terminal el archivo socket_servidor.py

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_servidor.py
El servidor está listo
```

Mientras que en otra terminal ejecutamos el archivo socket_cliente.py e ingresamos datos para iniciar una sesión del jugador.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_cliente.py
Bienvenido al juego de ahorcado :)
Introduce tu username: jugador1234
Introduce tu contraseña: 123456
```

Una vez que enviamos esta información al servidor, entonces recibimos un mensaje de registro y login exitoso.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_cliente.py
Bienvenido al juego de ahorcado :)
Introduce tu username: jugador1234
Introduce tu contraseña: 123456
User registered successfully
Successful login
Dame una letra:
```

Del lado del servidor podemos observar los registros de las conexiones que se han hecho con este, viendo que el identificador asignado al primer usuario es 48665.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_servidor.py
El servidor está listo
Se conectó ('127.0.0.1', 48665)
Se conectó ('127.0.0.1', 48665)
Se conectó ('127.0.0.1', 48665)
```

Enviamos una letra para empezar una partida.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_cliente.py
Bienvenido al juego de ahorcado :)
Introduce tu username: jugador1234
Introduce tu contraseña: 123456
User registered successfully
Successful login
Dame una letra: a
Dame una letra:
```

Como aún no hay suficientes jugadores, entonces los datos de la partida que se almacenan en el archivo ‘solucion.txt’ indican que aún no hay datos guardados y que el estado de la partida aún es no disponible.

solucion.txt


```
{'juego': {'jugador1': {'username': '', 'estado': '', 'progreso': '', 'errores': 0}, 'jugador2': {'username': '', 'estado': '', 'progreso': '', 'errores': 0}, 'siguiente_jugador': '', 'estado_partida': 'no disponible'}, 'secreto': ''}
```


Necesitamos dos jugadores para jugar nuestro juego de ahorcado, por lo que en una nueva terminal registramos a un segundo jugador de la misma manera en la que se indicó previamente.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_cliente.py
Bienvenido al juego de ahorcado :)
Introduce tu username: holaaa
Introduce tu contraseña: abcd
User registered successfully
Successful login
Dame una letra:
```

Notamos que se muestran mensajes nuevos con un id distinto al primero en las conexiones realizadas con el servidor.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_servidor.py
El servidor está listo
Se conectó ('127.0.0.1', 48665)
Se conectó ('127.0.0.1', 48665)
Se conectó ('127.0.0.1', 48665)
Se conectó ('127.0.0.1', 57908)
Se conectó ('127.0.0.1', 57908)
```

Al enviar una letra desde el jugador que se registró más tarde, notamos que ha sido guardado como Jugador 2, pero desafortunadamente el programa se detiene en este caso, interrumpiendo el juego y deteniendo la comunicación para darle continuidad a la partida.

```
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$ python socket_cliente.py
Bienvenido al juego de ahorcado :)
Introduce tu username: holaaa
Introduce tu contraseña: abcd
User registered successfully
Successful login
Dame una letra: b
Jugador 2
(base) mnava@debian12:~/QuintoSemestre/Redes/Practica1Redes$
```

Sin embargo, vemos que el archivo ‘solucion.txt’ sí ha registrado un cambio en la información almacenada de la partida. Teniendo asignados a los jugadores y los estados en los que se encuentran.

solucion.txt


```
{'juego': {'jugador1': {'username': 'jugador1234', 'estado': 'continua', 'progreso': '_ _ _ _ _', 'errores': 0}, 'jugador2': {'username': 'holaaa', 'estado': 'continua', 'progreso': '_ _ _ _ _', 'errores': 0}, 'siguiente_jugador': 'jugador1234', 'estado_partida': 'continua'}}
```