

ANEXO

COMO UTILIZAR EL SISTEMA PARA PRUEBAS:

Configuración del entorno:	2
Registrar usuario nuevo:	2
.....	2
Hacer login y enviar un mensaje:	2
Hacer login y leer los mensajes:.....	2
Pruebas de sniffer:	3
Prueba modo inseguro:.....	3
Prueba modo seguro (con TLS):.....	3
Pruebas de estrés y conexión persistente:.....	3

COMO UTILIZAR EL SISTEMA PARA PRUEBAS:

Configuración del entorno:

Para el sistema se ha utilizado la base de datos PostgreSQL v17

Al ejecutar el archivo **serversocket.py** el sistema automáticamente crea una base de datos llamada *ssiidb* y crea dos usuarios de prueba/ejemplo.

Para ello es importante editar el archivo `serversocket.py` con las credenciales correctas de la base de datos PostgreSQL 17 (es la utilizada para el sistema)

Al INICIO DEL ARCHIVO `serversocket.py`:

```
# --- Configuración de conexión ---
# Usa variables de entorno o pon valores por defecto para desarrollo
DB_NAME = os.getenv("PGDATABASE", "ssiidb")
DB_USER = os.getenv("PGUSER", "postgres") # CAMBIAR POR USUARIO ADECUADO
DB_PASS = os.getenv("PGPASSWORD", "pua12398") # CAMBIAR POR CONTRASEÑA ADECUADA
DB_HOST = os.getenv("PGHOST", "127.0.0.1")
DB_PORT = os.getenv("PGPORT", "5432")
```

De este modo será capaz de acceder al sistema de PostgreSQL para crear la DATABASE si no está creada.

Registrar usuario nuevo:

- Carpeta **prueba-logs** → abrir el archivo **new_user.txt** se trata de un log del terminal del cliente, seguir los pasos y revisar la base de datos (PostgreSQL)
- Al registrar un usuario el sistema crea automáticamente una tabla llamada `usuarios`, donde se guardarán los usuarios que vayamos creando. Por ello, para comprobarlo, debemos introducir en el terminal de `psql` (SHELL), con el usuario que hemos configurado en el sistema:

```
SELECT * FROM usuarios; (por ejemplo)
```

```
ssiidb=# SELECT * FROM usuarios;
 username | password | cuenta
-----+-----+-----
 Angel    | $2b$12$nxadXT4WBDX60qLL71U1He46LFGfS/e0o3Rwpp/EJVg7e9IujDCjK | 1000
 Rafael   | $2b$12$BvcTF2sdoLWNOsf74DXdA0oztEqMznkt2jZ/0mJ3MCGLEzNL8Fgxy | 1000
```

Hacer login y enviar un mensaje:

- Revisar usuarios en la base de datos, por defecto, se crearán 2 usuarios:
 - o User: "Angel" pass: "Angel123?"
 - o User: "Rafael" pass: "Rafael123?"
- carpeta **prueba-logs** → abrir **login&message.txt**. Seguir el proceso del ejemplo.
- El sistema crea la tabla `mensajes` de forma automática al enviar el primer mensaje.
- Para comprobar el envío correcto de un mensaje:

```
ssiidb=# SELECT * FROM mensajes;
 id | emisor | destinatario | contenido | fecha | leido
-----+-----+-----+-----+-----+-----
 1 | Angel | Rafael | Hola Rafael, soy Angel | 2025-10-22 17:17:02.817798 | f
(1 fila)
```

Hacer login y leer los mensajes:

- Carpeta **pruebas-logs**: abrir archivo **login&viewmessages.txt** y seguir el proceso del ejemplo
- Nota: si el usuario loggeado no ha recibido ningún mensaje aparecerá no hay mensajes.

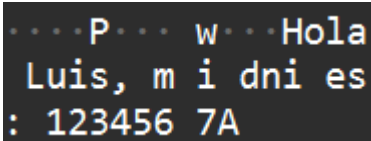
Pruebas de sniffer:

Se han realizado pruebas de sniffer para comprobar el correcto funcionamiento del protocolo SSL/TLS.

Prueba modo inseguro:

Pasos:

1. Lanzar `server_warning.py` y `client_warning.py`
 - a. Estos archivos son iguales en cuanto a funcionalidad que los archivos finales del sistema pero sin seguridad aplicada.
2. Enviar un mensaje:
 - a. Si capturamos el trafico con wireshark obtendremos una captura muy parecida a ***pruebas-logs/ captura_INSEGURO.pcapng***
3. Si comprobamos algunos de los mensajes intercambiados podemos ver perfectamente el mensaje



...P... w...Hola
Luis, m i dni es
: 123456 7A

Prueba modo seguro (con TLS):

Pasos:

1. Lanzar ***serversocket.py*** y ***clientsocket.py***
2. Enviar un mensaje:
 - a. Si capturamos el trafico con wireshark obtendremos una captura muy parecida a ***pruebas-logs/ captura_TLS1.3.pcapng***
3. Si comprobamos las tramas cada mensaje estará encapsulado con el protocolo TLS 1.3, siendo imposible saber su contenido mediante el sniff del tráfico (man in the middle)

Pruebas de estrés y conexión persistente:

Para realizar prueba de estrés y conexión persistente existe el archivo ***estres_test_client.py*** el cual simula el login de 300 usuarios. En el servidor esta implementado un hilo para la funcionalidad principal que se ejecuta con cada conexión.

Para realizar la prueba:

1. Lanzar `serversocket.py`
2. Lanzar ***estres_test_client.py*** (esperar, prepara los 300 clientes al inicio de la ejecución)