ANEXO

[COMO UTILIZAR EL SISTEMA PARA PRUEBAS:](#_Toc212050314)

[Configuración del entorno: 1](#_Toc212050315)

[Registrar usuario nuevo: 1](#_Toc212050316)

[1](#_Toc212050317)

[Hacer login y enviar un mensaje: 1](#_Toc212050318)

[Hacer login y leer los mensajes: 2](#_Toc212050319)

[Pruebas de sniffer: 2](#_Toc212050320)

[Prueba modo inseguro: 2](#_Toc212050321)

[Prueba modo seguro (con TLS): 2](#_Toc212050322)

[Pruebas de estrés y conexión persistente: 2](#_Toc212050323)

# 

# 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:

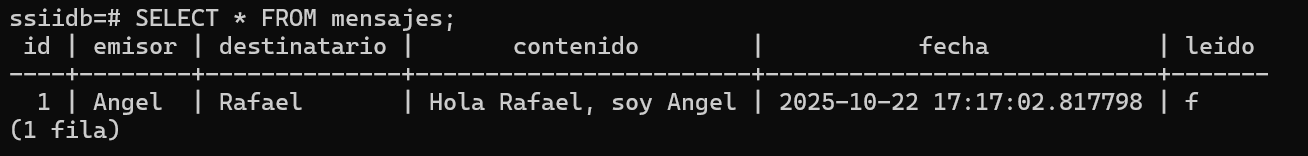
* Carpeta ***prueba-logs*** 🡪 abrir el archivo ***new\_user.txt*** se trata de un log del terminal del cliente, seguir los pasos y revisar la base de datos (PostgreSQL)
* Al registrar un usuario el sistema crea automáticamente una tabla llamada usuarios, donde se guardarán los usuarios que vayamos creando. 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)

## Texto El contenido generado por IA puede ser incorrecto.

## Hacer login y enviar un mensaje:

* Revisar usuarios en la base de datos, por defecto, se crearán 2 usuarios:
  + User: “Angel” pass: “Angel123?”
  + 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:



## 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
   1. Estos archivos son iguales en cuanto a funcionalidad que los archivos finales del sistema pero sin seguridad aplicada.
2. Enviar un mensaje:
   1. 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

Texto

El contenido generado por IA puede ser incorrecto.

### Prueba modo seguro (con TLS):

Pasos:

1. Lanzar ***serversocket.py*** y ***clientsocket.py***
2. Enviar un mensaje:
   1. 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)