Laboratorio de Sistemas Operativos y Redes

Diciembre 2025

CHAT EN LA MATRIX

Comunicación en red utilizando Matrix: Servidor Tuwunel y cliente FluffyChat

CONTENIDO

OBJETIVO DEL TP

El objetivo es instalar, configurar y utilizar un sistema de mensajería basado en el protocolo Matrix, implementando:

- Un servidor **Matrix** (**Tuwunel**), compilado desde el código fuente.
- Un cliente **Matrix** (**FluffyChat**), también compilado desde repositorio.
- Registro y autenticación de usuarios.
- Comunicación en red entre dos máquinas.

Andrés Mora y
Margarita Cortizas

Laboratorio de Sistemas Operativos y Redes

Diciembre 2025

MATRIX

Qué es Matrix?

- Protocolo abierto para mensajería descentralizada.
- Cada servidor es un homeserver.
- Los clientes se conectan a un servidor y crean salas, usuarios y mensajes.
- Permite comunicación segura, federada y moderna.

Laboratorio de Sistemas Operativos y Redes

Diciembre 2025

TUWUNEL

Qué es Tuwunel?

- Implementación de servidor Matrix escrita en Rust.
- Rápido, eficiente y liviano.
- Requiere compilarse manualmente.
- Guarda datos en RocksDB.
- Será nuestro servidor para el laboratorio.

Laboratorio de Sistemas Operativos y Redes

Diciembre 2025

FLUFFYCHAT

Qué es FluffyChat?

- Cliente moderno y liviano hecho en Flutter.
- Permite chatear, crear salas y enviar mensajes.
- Lo compilamos en Linux con soporte para escritorio.
- Interfaz amigable y fácil de usar.

Cliente Matrix

Laboratorio de Sistemas Operativos y Redes
Diciembre 2025

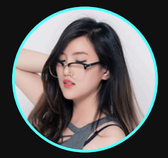
VAMOS A EMPEZAR

Listos?

SISTEMA OPERATIVO UTILIZADO:

UBUNTU DESKTOP 24.04 LTS

- VM VirtualBox v1.2 64 bits
- CPU Intel Core i5-7200U 2.50GHz (2 nucleos asignados a vm)
- 8GB RAM
- 128GB Almacenamiento
- Adaptador de red configurado en VM como "Adaptador Puente"



Software necesario para Tuwunel
Build Essentials: para compilar software en Linux
Clang: para compilar C/C++
RocksDB: motor de base de datos embebido
Git: necesario para clonar el repositorio
Rust: lenguaje en el que está desarrollado Tuwunel

Software necesario para FluffyChat
Build Essentials
Git
Clang
Flutter: framework creado por Google
Dart SDK: lenguaje que usa Flutter
SQLite: base de datos en memoria

INSTALACIÓN DEL SERVIDOR

- Actualización e instalación de paquetes (build-essential, clang, RocksDB, libssl-dev, etc.)
- Instalación de Rust con rustup.

```
curl https://sh.rustup.rs -sSf | sh
```

- Configuración del entorno con

```
source $HOME/.cargo/env.
```

1

Instalación de Tuwunel – Dependencias

```
sudo apt update
sudo apt install -y \
    build-essential \
    clang \
    pkg-config \
    libclang-dev \
    liburing-dev \
    libzstd-dev \
    librocksdb-dev \
    cmake \
    git \
    libssl-dev
```

INSTALACIÓN DEL SERVIDOR

- Elegimos la instalación estándar. Al finalizar, deberíamos ver algo como esto, y configurar según las instrucciones indicadas

```
Rust is installed now. Great!

To get started you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo's bin directory ($HOME/.cargo/bin).

To configure your current shell, you need to source
the corresponding env file under $HOME/.cargo.

This is usually done by running one of the following (note the leading DOT):
. "$HOME/.cargo/env"            # For sh/bash/zsh/ash/dash/pdksh
source "$HOME/.cargo/env.fish"  # For fish
source $"($nu.home-path)/.cargo/env.nu"  # For nushell
```

INSTALACIÓN DEL SERVIDOR

- Clonar el repo:

```
git clone https://github.com/matrix -
construct/tuwunel.git
cd tuwunel
```

- Compilar: `cargo build --release`
- Al finalizar, nos queda el ejecutable en:

```
target/release/tuwunel
```

- Matrix necesita un directorio donde Tuwunel guarda la DB y la config:

```
mkdir ~/tuwunel-data
Crear nano config.toml
```

- Ejecutar servidor:

```
~/tuwunel/target/release/tuwunel -c ~/tuwunel-
data/config.toml
```

1. Compilación y configuración de Tuwunel

2. Generamos archivo de configuración mínima:

```
[global]
server_name = "labo-unq"
database_path =
"/home/{usuario}/tuwunel-
data/db"
address = ["0.0.0.0"]
port = 8008
allow_registration = true
registration_token = "token123"
```

INSTALACIÓN DEL SERVIDOR

- Listo! Si todo salió bien deberíamos ver el servidor corriendo:

```
andres@ubuntu-labo:~/tuwunel-data$ ~/tuwunel/target/release/tuwunel -c ~/tuwunel-data/config.toml
2025-12-06T00:11:47.616791Z  INFO tuwunel::server: 1.4.7 server_name=labo-unq database_path="/home/andres/tuwunel-data/db"
 log_levels=info
2025-12-06T00:11:47.798689Z  INFO main:start:open: tuwunel_database::engine::open: Opened database. columns=95 sequence=0
time=158.632847ms
2025-12-06T00:11:47.804035Z  WARN main:start: tuwunel_service::migrations: Created new RocksDB database with version 17
2025-12-06T00:11:47.807141Z  INFO tuwunel_router::serve::plain: Listening on [0.0.0.0:8008]
```

INSTALACIÓN DEL CLIENTE

- Instalación de dependencias de Flutter y GTK.
- Clonar Flutter y agregarlo al PATH.

```
cd ~
git clone https://github.com/flutter/flutter.git
-b stable
echo 'export PATH="$PATH:$HOME/flutter/bin"' >>
~/.bashrc
source ~/.bashrc

flutter --version
```

1

Instalación de FluffyChat – Dependencias

```
sudo apt update

sudo apt install -y \
  git curl unzip xz-utils build-
essential \
  libgtk-3-dev liblzma-dev clang
cmake ninja-build pkg-config
libsqlite3-dev \
  libwebkit2gtk-4.1-dev
libsecret-1-dev libjsoncpp-dev
```

INSTALACIÓN DEL CLIENTE

- Deberíamos ver este mensaje:

```
andres@ubuntu-labo:~$ flutter --version
Downloading Linux x64 Dart SDK from Flutter engine a5cb96369ef86c7e85abf5d662a1ca5d89775053...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  200M  100  200M    0     0  24.1M      0  0:00:08  0:00:08 --:--:-- 28.4M
Building flutter tool...
Resolving dependencies...
Downloading packages... (3.2s)
Got dependencies.
Flutter 3.38.4 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 66dd93f9a2 (2 days ago) • 2025-12-03 14:56:10 -0800
Engine • hash 360877569ab6632a564a0a8a815b2e0fe5ae294a (revision a5cb96369e) (46
hours ago) • 2025-12-03 20:56:06.000Z
Tools • Dart 3.10.3 • DevTools 2.51.1
```

INSTALACIÓN DEL CLIENTE

- Activar soporte Linux Desktop.
- Debería encontrar Linux como dispositivo:

```
flutter config --enable-linux-desktop
flutter devices
```

```
andres@ubuntu-labo:~$ flutter devices
Downloading Material fonts...                                   491ms
Downloading Gradle Wrapper...                                   176ms
Downloading package sky_engine...                               124ms
Downloading package flutter_gpu...                               21ms
Downloading flutter_patched_sdk tools...                        268ms
Downloading flutter_patched_sdk_product tools...                259ms
Downloading linux-x64 tools...                                  3.1s
Downloading linux-x64/font-subset tools...                      304ms
Found 1 connected device:
  Linux (desktop) • linux • linux-x64 • Ubuntu 24.04.3 LTS 6.14.0-36-generic
```

INSTALACIÓN DEL CLIENTE

- Clonar repo oficial:

```
cd ~
git clone https://github.com/krille-chan/fluffychat.git
cd fluffychat
```

- Obtener dependencias:

```
flutter pub get
```

- Compilar:

```
flutter build linux --release
```

- Esto genera los archivos binarios disponibles para poder correr la aplicación.
- Ejecutar el binario desde `.build/linux/x64/release/bundle/fluffychat`

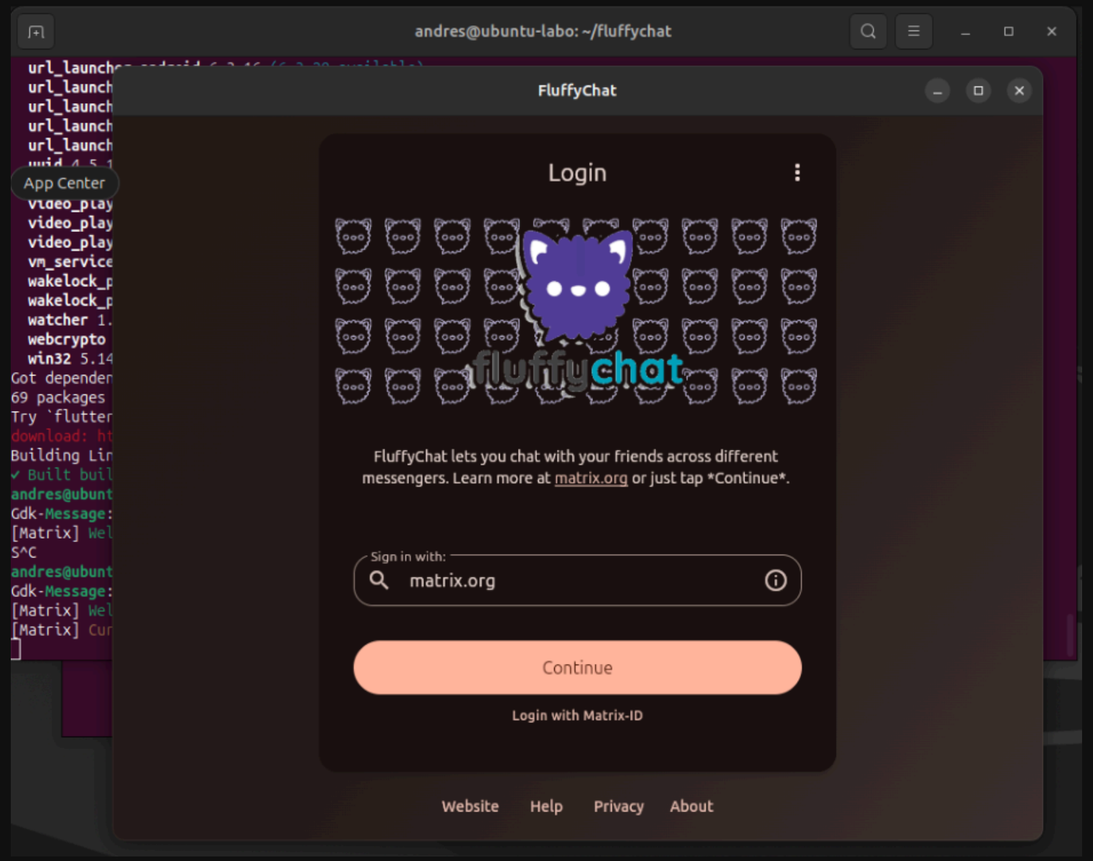
1

Compilación de FluffyChat

```
Built build/linux/x64/release/bundle/fluffychat
```

INSTALACIÓN DEL CLIENTE

Listo! Ya tenemos nuestro cliente desktop funcionando:



Registro de Usuarios

REGISTRO Y AUTENTICACIÓN

- Para el laboratorio, vamos a hacer uso del servicio que acabamos de configurar y usarlo como chat en red. Debemos registrarnos como usuarios desde el cliente para poder conectarse y chatear.
- Acá hay **dos opciones**:
- La primera es a través de una api pública expuesta por el mismo servicio, así:

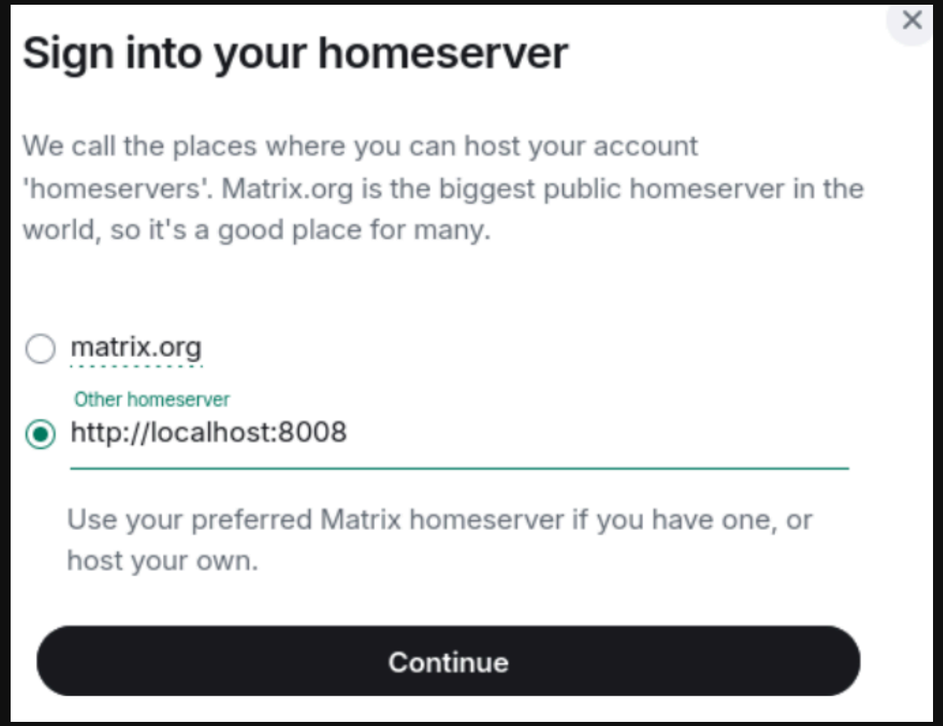
```
curl -XPOST -d '{
  "username": "andres",
  "password": "123456",
  "auth": {
    "type":
"m.login.registration_token",
    "token": "token123"
  }
}'
"http://localhost:8008/_matrix/c
lient/v3/register"
```

Importante: el token debe ser el mismo que se configuró en el servidor

Registro de Usuarios

REGISTRO Y AUTENTICACIÓN

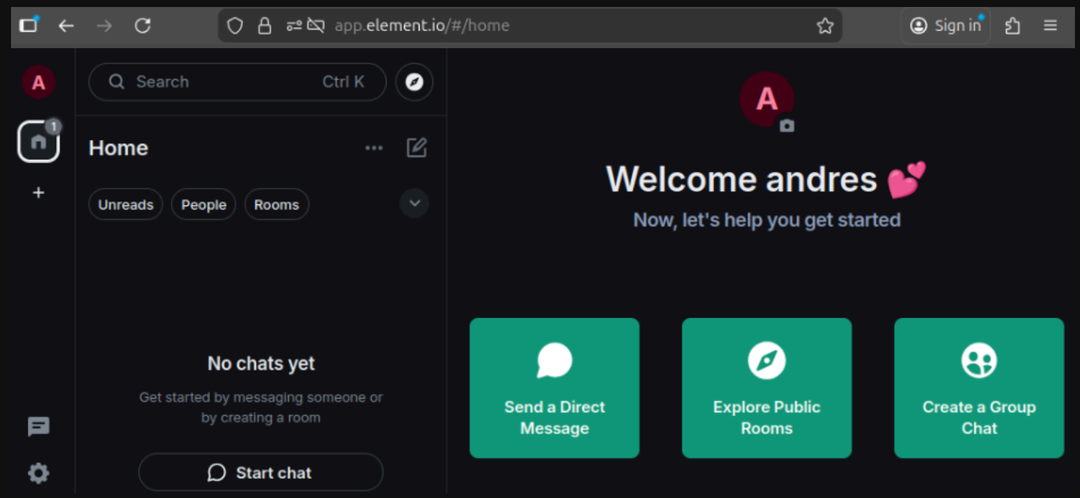
- La otra opción es utilizar algún cliente que tenga la opción de registro. El que usamos de muestra (FluffyChat) no lo tiene integrado, pero podemos usar otro, como por ejemplo Element.
- Accedemos desde la web: https://app.element.io/
- Elegimos editar el servidor, seleccionamos otro homeserver y escribimos http://localhost:8008 (o el puerto que se haya configurado en el archivo config.toml)



Registro de Usuarios

REGISTRO Y AUTENTICACIÓN

- Elegimos la opción para crear una nueva cuenta, ingresamos el **usuario** que queramos, **password** y confirmamos.
- Luego, nos aparece otra pantalla para ingresar el **token**. Lo escribimos, listo!
- Ya tenemos acceso al chat:

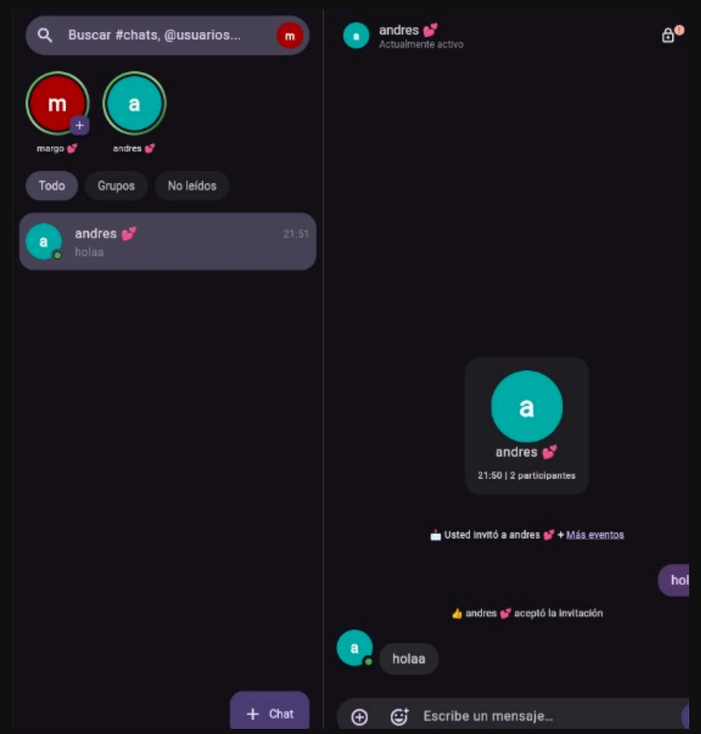


(**Los dos corazones los agrega solos...**)

Chat en Red

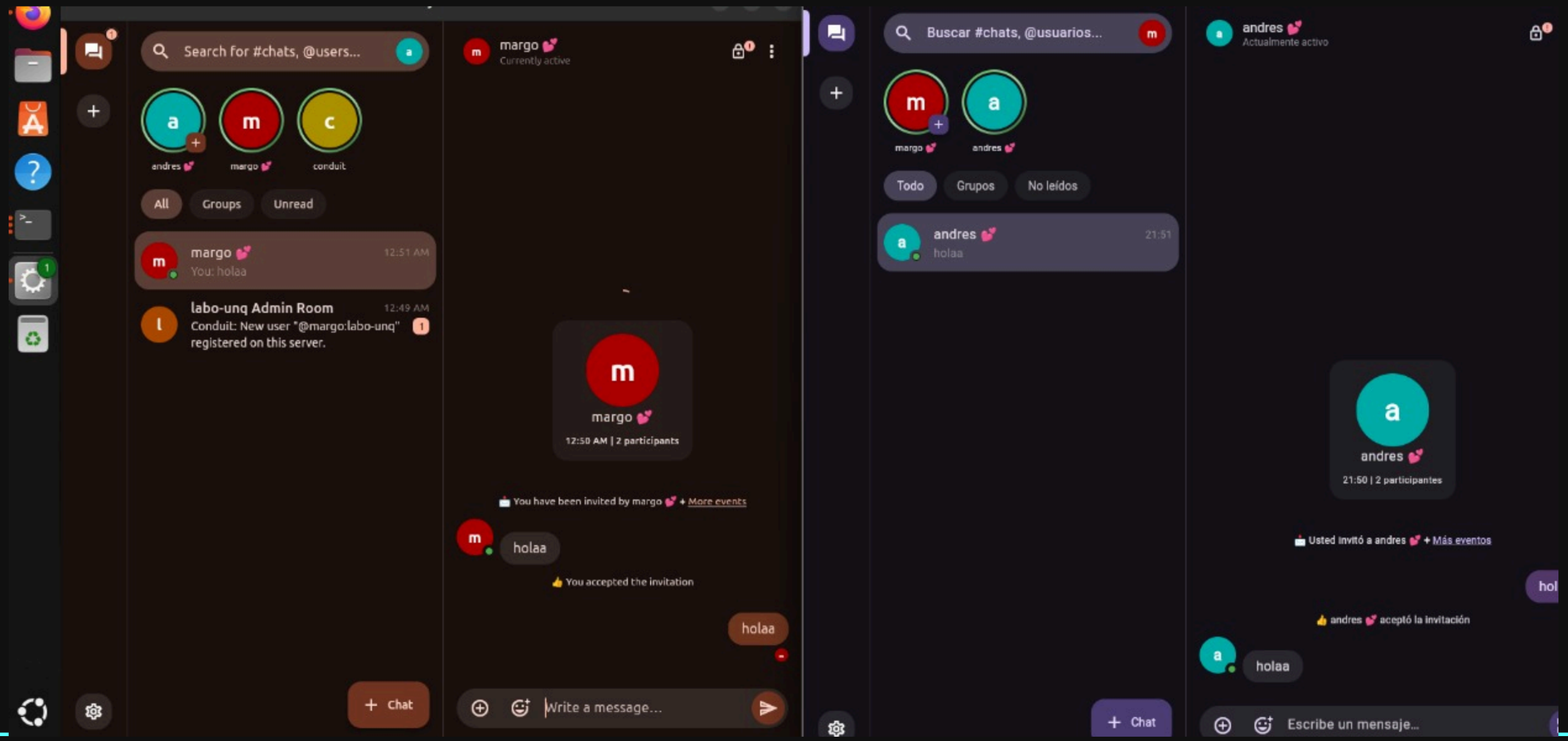
CHATEANDO EN RED

- Lo primero es conocer la ip de la máquina donde corra el servicio y que sea accesible desde la red. Si usamos una máquina virtual con VirtualBox debemos configurarla como “**Adaptador puente**” desde las opciones de red
- Abrimos **FluffyChat** e ingresamos al servidor. Si estamos usando la misma máquina donde está corriendo el servicio podemos ingresar directamente con http://localhost:8008. Sino debemos reemplazar “**localhost**” por la **ip** de la pc del **servidor**. Ingresamos nuestras credenciales.



Chat en Red

Ingresamos con otro usuario desde otra pc, buscamos el usuario con quien queremos chatear, iniciamos la conversación y listo! Ya podemos chatear en red!

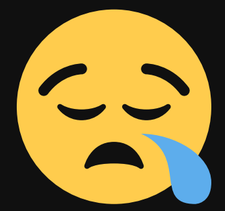


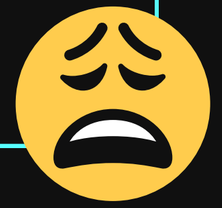
Problemáticas que tuvimos

- Incompatibilidad de dependencias y/o librerías deprecadas: tanto en el servidor como en el cliente nos encontramos con ciertos errores al intentar buildear la aplicación.

- Error para registrarnos en nuestro propio homeserver: nos faltaba una config allow_registration en true para permitirlo

- Restricciones de algunos clientes: fluffychat no permite registro de usuarios. Debimos hacerlo a través de la api y/o mediante otro cliente (Element)





CONCLUSIONES

1. Se logró instalar un servidor Matrix funcional.
2. Se compiló y configuró un cliente moderno.
3. Se registraron usuarios correctamente.
4. Se validó la comunicación en red local.

IMPLEMENTAR UN CHAT DESDE CERO DEMUESTRA CÓMO FUNCIONAN REALMENTE LOS SERVICIOS QUE USAMOS TODOS LOS DÍAS.

Fin.

GRACIAS!