

Fecha de entrega: 13 de abril de 2024

Práctica final: Implementación de un chat Multicast

Asignatura: Sistemas distribuidos

4ºA GITT

Prof. Luis Francisco Sánchez Merchante

Carlos Martín de Argila Lorente

Tabla de contenido

Información Importante.....3

 Como ejecutar:3

 Librerías Python necesarias:3

Introducción.....3

Desarrollo3

Uso de la aplicación5

 Registro o inicio de sesión5

 Crear o unirse a un chat6

 Crear un chat.....6

 Unir a un chat.....7

 Chat.....7

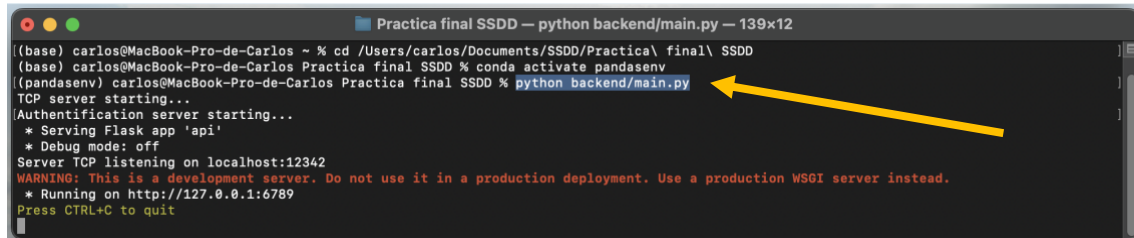
Información extra.....9

Conclusión.....9

Información Importante

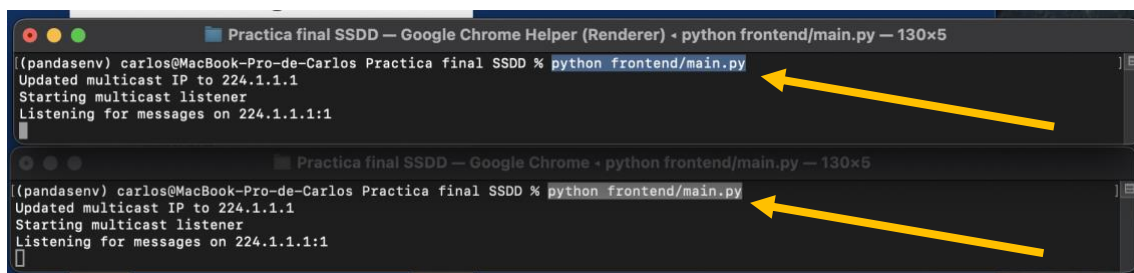
Como ejecutar:

Primero ejecutar el backend desde la carpeta Practica final SSDD:

A terminal window titled "Practica final SSDD — python backend/main.py — 139x12". The prompt is "(base) carlos@MacBook-Pro-de-Carlos ~". The user enters "cd /Users/carlos/Documents/SSDD/Practica\ final\ SSDD". The prompt changes to "(base) carlos@MacBook-Pro-de-Carlos Practica final SSDD". The user enters "conda activate pandasenv". The prompt changes to "(pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD". The user enters "python backend/main.py". The output shows "TCP server starting...", "Authentication server starting...", "* Serving Flask app 'api'", "* Debug mode: off", "Server TCP listening on localhost:12342", "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.", "* Running on http://127.0.0.1:6789", and "Press CTRL+C to quit". A yellow arrow points to the command "python backend/main.py".

```
((base) carlos@MacBook-Pro-de-Carlos ~ % cd /Users/carlos/Documents/SSDD/Practica\ final\ SSDD
(base) carlos@MacBook-Pro-de-Carlos Practica final SSDD % conda activate pandasenv
(pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD % python backend/main.py
TCP server starting...
Authentication server starting...
* Serving Flask app 'api'
* Debug mode: off
Server TCP listening on localhost:12342
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:6789
Press CTRL+C to quit
```

Y lo mismo con todas las instancias del frontend que se quieran:

Two terminal windows are shown. The top window is titled "Practica final SSDD — Google Chrome Helper (Renderer) - python frontend/main.py — 130x5". The prompt is "(pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD". The user enters "python frontend/main.py". The output shows "Updated multicast IP to 224.1.1.1", "Starting multicast listener", and "Listening for messages on 224.1.1.1:1". A yellow arrow points to the command "python frontend/main.py". The bottom window is titled "Practica final SSDD — Google Chrome - python frontend/main.py — 130x5". The prompt is "(pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD". The user enters "python frontend/main.py". The output shows "Updated multicast IP to 224.1.1.1", "Starting multicast listener", and "Listening for messages on 224.1.1.1:1". A yellow arrow points to the command "python frontend/main.py".

```
((pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD % python frontend/main.py
Updated multicast IP to 224.1.1.1
Starting multicast listener
Listening for messages on 224.1.1.1:1

((pandasenv) carlos@MacBook-Pro-de-Carlos Practica final SSDD % python frontend/main.py
Updated multicast IP to 224.1.1.1
Starting multicast listener
Listening for messages on 224.1.1.1:1
```

Librerías Python necesarias:

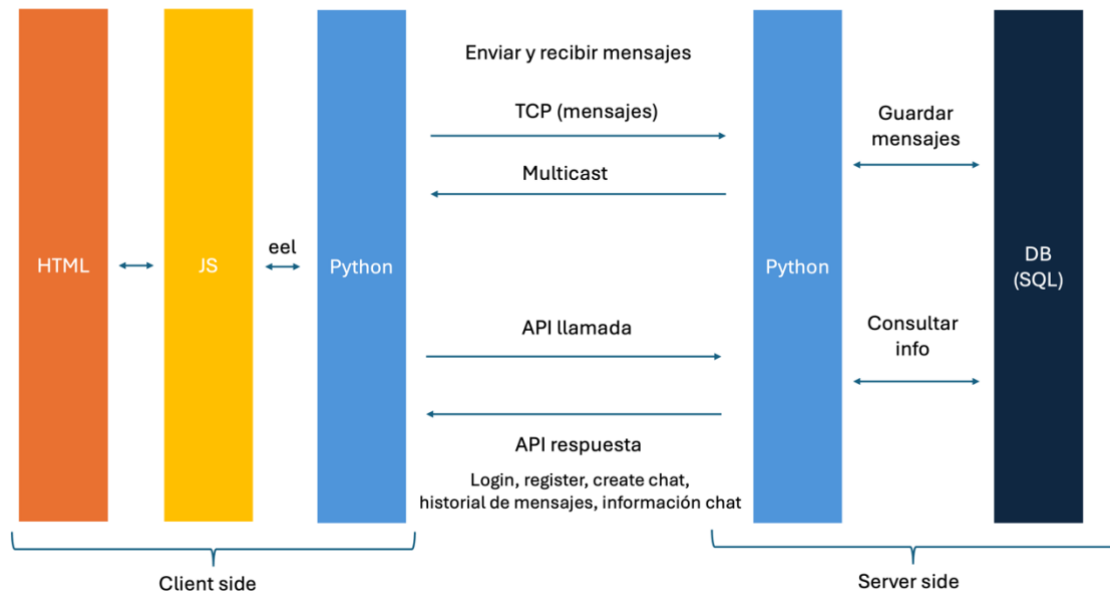
- Eel: pip install Eel
- Flask: pip install Flask
- Flask-Cors: pip install Flask-Cors
- Flask-HTTPAuth: pip install Flask-HTTPAuth

Introducción

En esta práctica vamos a desarrollar un chat multicast con una interfaz gráfica dividida en dos zonas esenciales: una para visualizar los mensajes de todos los participantes y otra para escribir y enviar mensajes. Este proyecto permitirá la comunicación en tiempo real entre varios usuarios. Hemos añadido además una función de login y de distintos chats para hacerla un poco más compleja.

Desarrollo

La arquitectura de la aplicación puede describirse con el siguiente esquema:



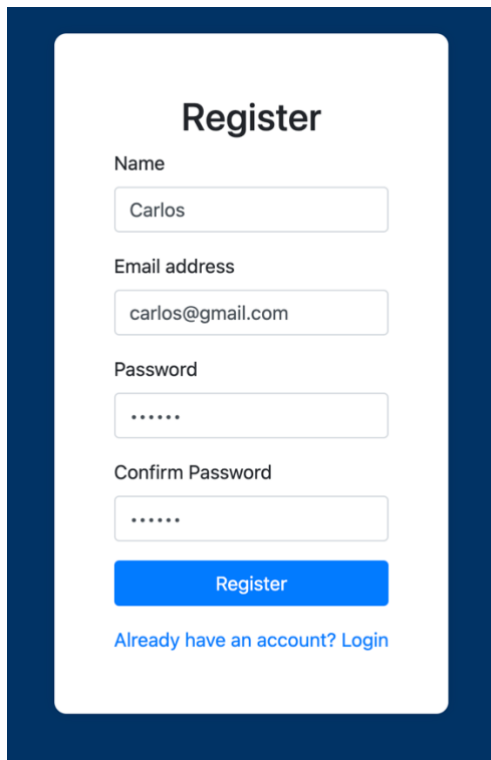
Cuando un cliente envía un mensaje, lo envía por TCP al servidor. Este lo que hace es guardar el mensaje y después con multicast se lo envía a todos los usuarios.

Esto nos permite tener las ventajas de TCP y usar Multicast para un chat de grupo. Y además al tener un servidor que hace de punto intermedio, podemos guardar todos los mensajes en una base de datos SQL que permite tener un historial.

Cuando un usuario requiere información, por ejemplo el nombre del chat, el historial del chat, que quiere crear un chat... Todo esto se hace a través de APIs que se han desarrollado con Flask.

Uso de la aplicación

Registro o inicio de sesión



Register

Name

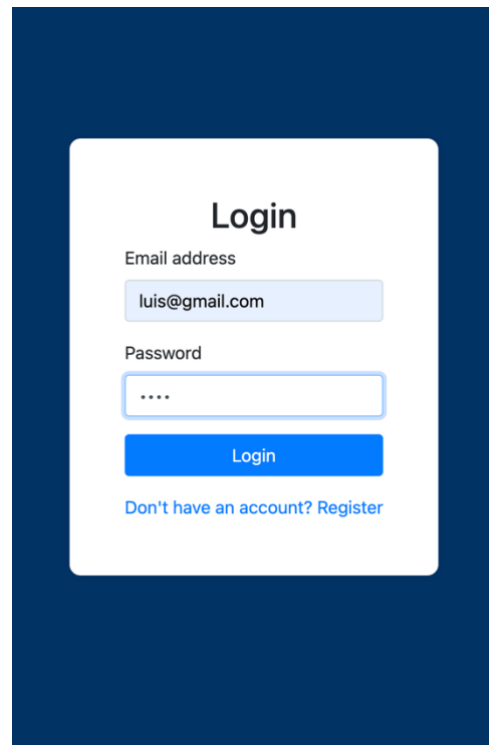
Email address

Password

Confirm Password

[Register](#)

[Already have an account? Login](#)



Login

Email address

Password

[Login](#)

[Don't have an account? Register](#)

Se pueden usar los siguientes usuarios para hacer pruebas o crear los que se quieran:

Usuario: [carlos@gmail.com](#)

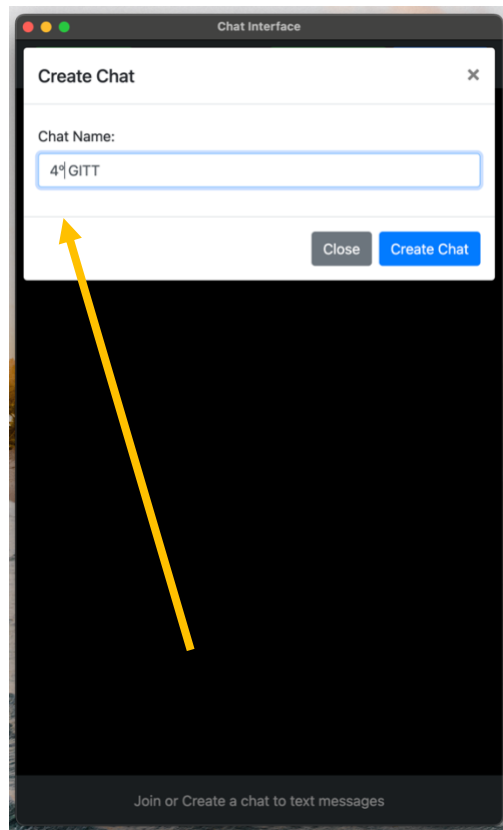
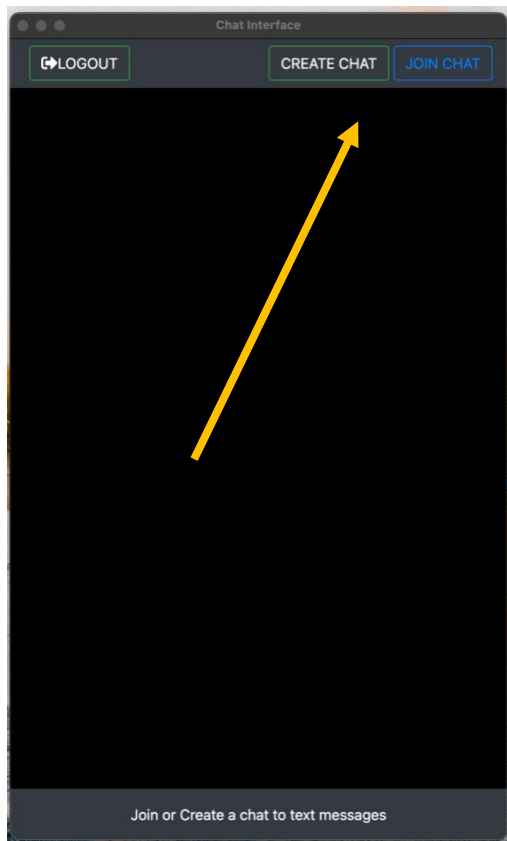
Contraseña: Carlos

Usuario: [luis@gmail.com](#)

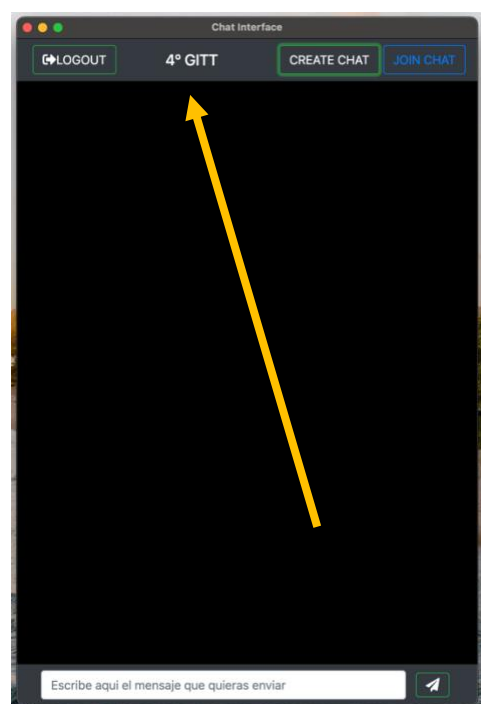
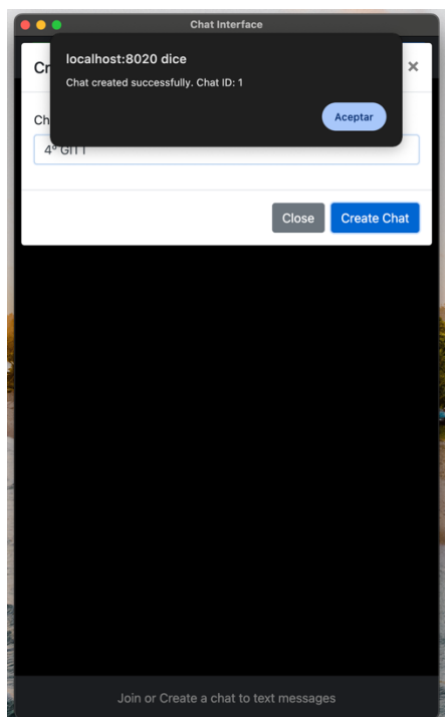
Contraseña: Luis

Crear o unirse a un chat

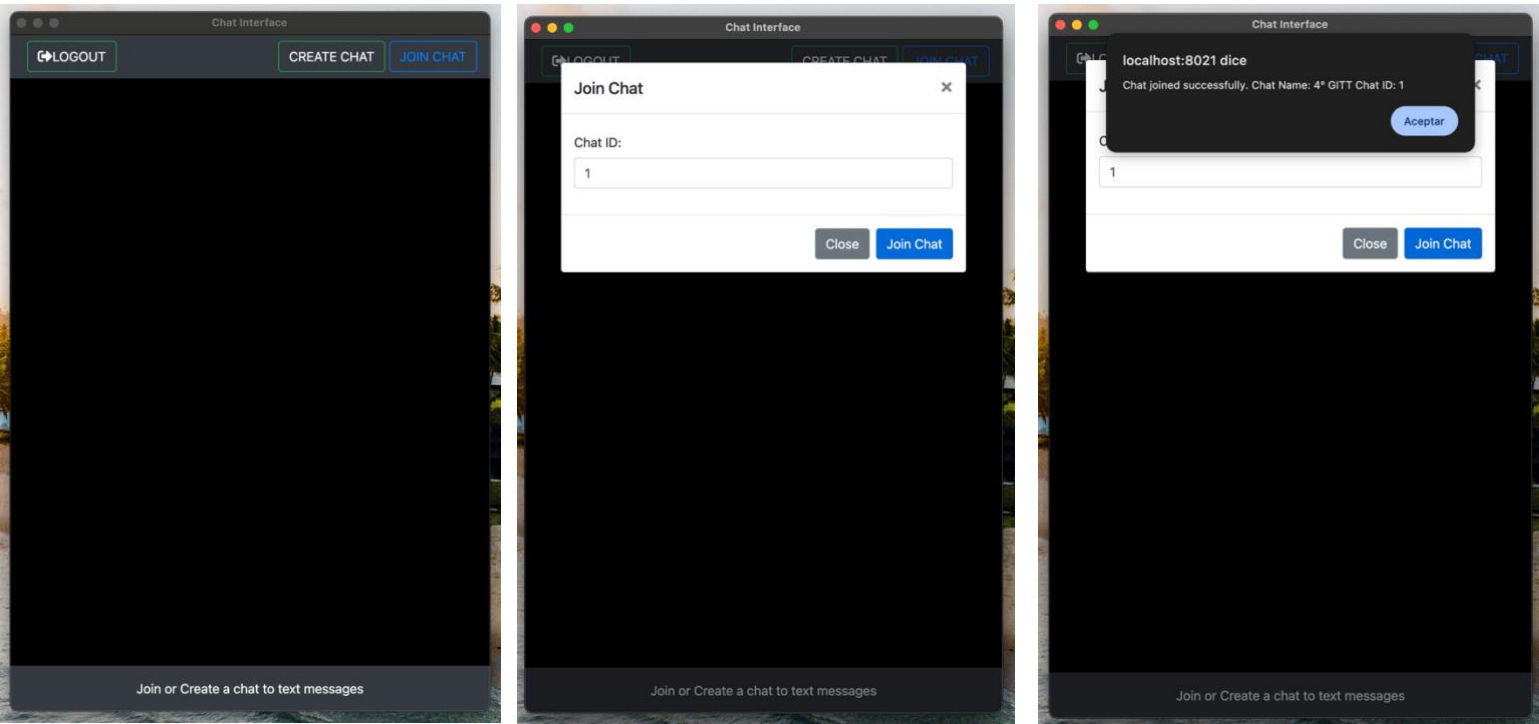
Crear un chat



El nombre es el que aparecerá después como título del chat. Una vez se crea un chat, la aplicación nos dará el ID del chat. Este es el ID que usarán los usuarios para unirse al chat.

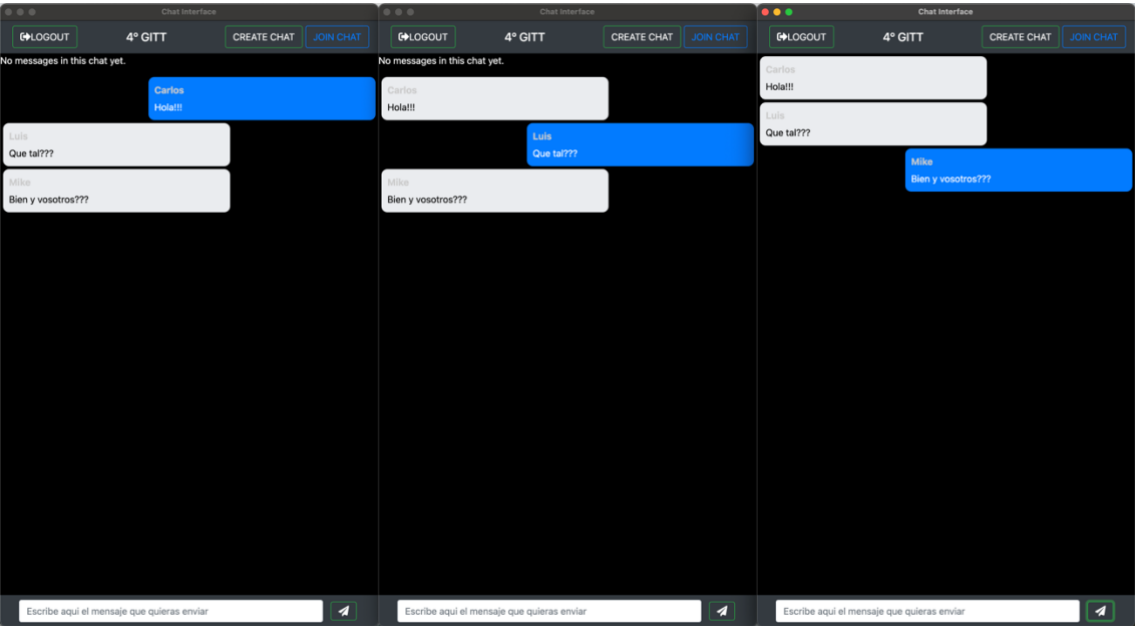


Unir a un chat



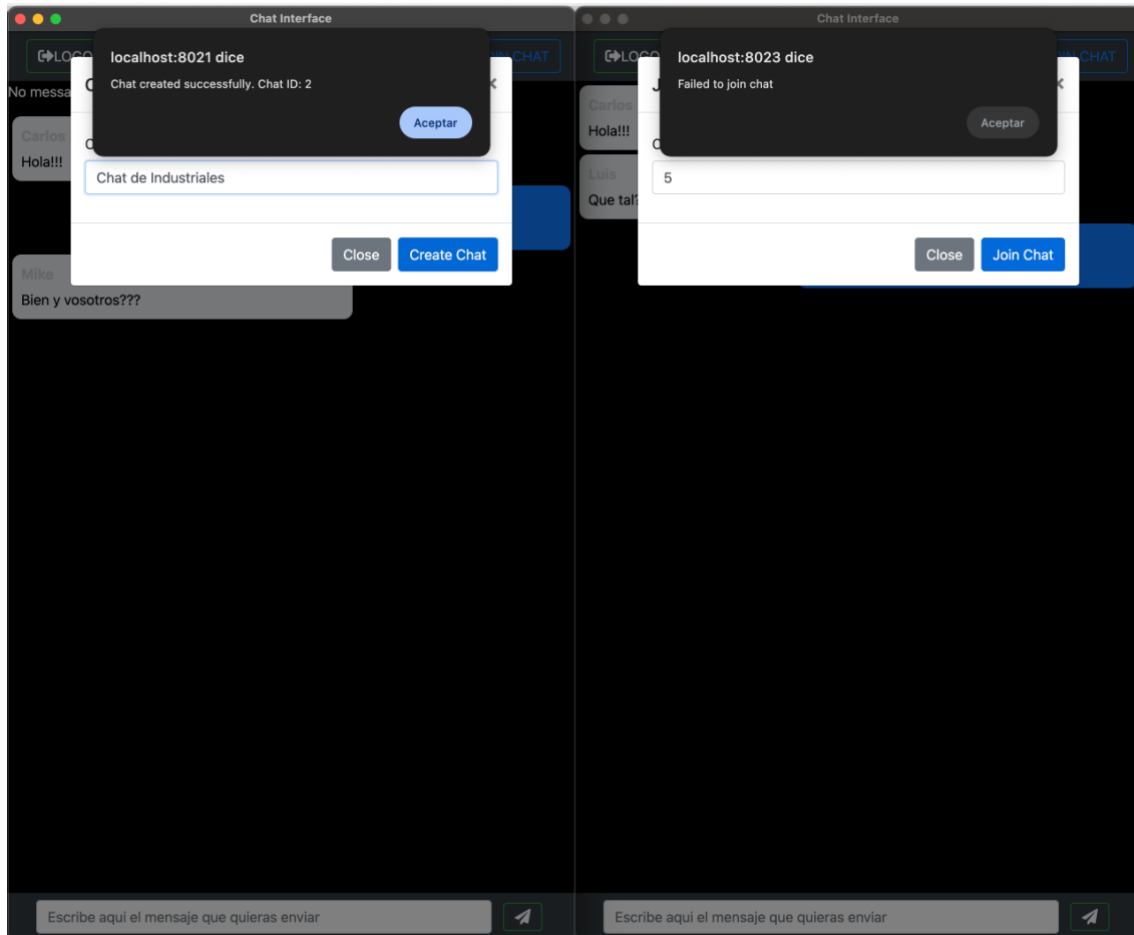
Chat

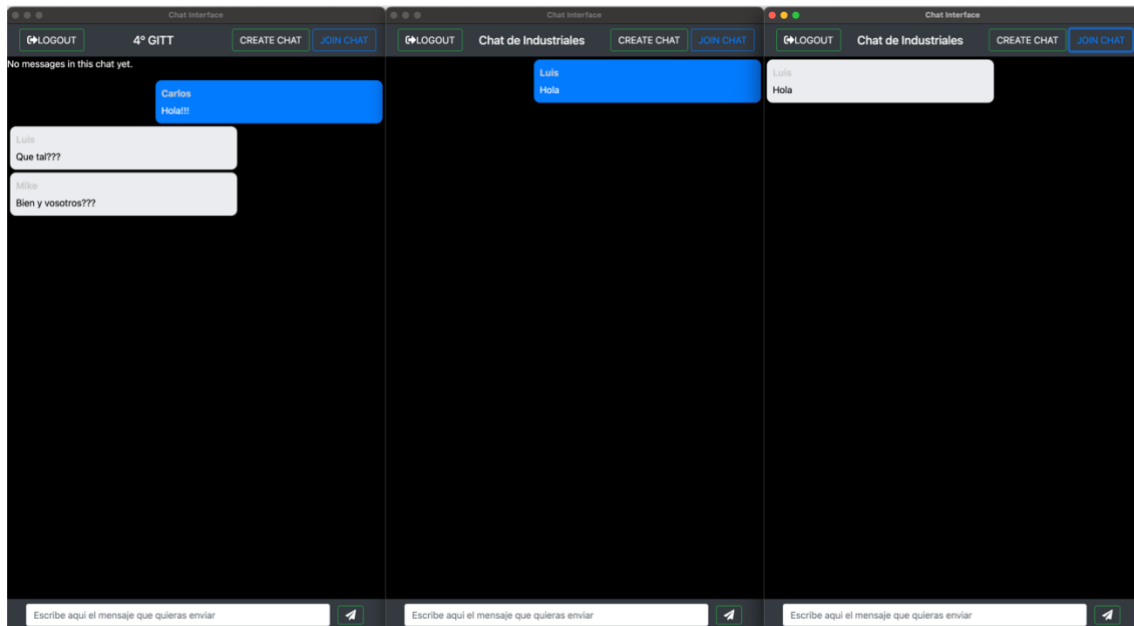
Una vez tenemos dos usuarios en un chat podemos empezar a escribir:



Aquí vemos 3 instancias y los 3 se pueden comunicar con todos. Sale el nombre de usuario en cada mensaje para identificarlo.

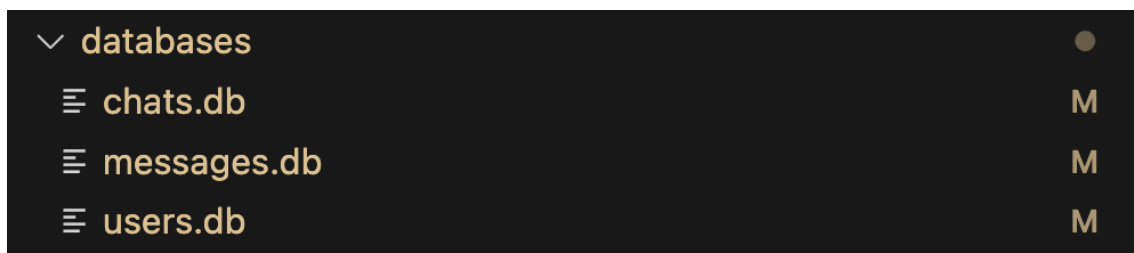
Y si ahora lo que hacemos es intentar unirnos a un chat que no existe efectivamente nos da un error y si creamos un chat nuevo (o nos unimos a un chat) se va a “borrar” de la pantalla del usuario la conversación del chat anterior y van a aparecer los mensajes de la nueva conversación.





Información extra

Los datos se guardan en estas 3 bases de datos:



Y cada chat multicast tiene un puerto diferente.

Conclusión

En esta práctica hemos desarrollado un sistema de chat multicast que facilita la comunicación en tiempo real entre múltiples usuarios. Hemos construido una interfaz gráfica intuitiva, dividida en una zona para visualizar mensajes y otra para escribirlos, logrando una interacción efectiva y simultánea entre los participantes.