Universidad Politécnica Salesiana

Carrera de Computación

Materia:

Sistemas distribuidos

Practica:

Virtualización de aplicaciones

Docente:

Ing. Cristian Timbi

Integrantes:

Juan Francisco Quizhpi Fajardo

Docker

Docker es una plataforma que se usa para el desarrollo y envió de aplicaciones de manera eficiente con el uso de los contenedores. Los contenedores son entornos ligeros y portables los cuales encapsulan a la a aplicación y sus dependencias para que puedan ser ejecutadas en todo momento en cualquier otro entorno ya sea en maquinas virtuales, maquinas con distintos sistemas operativos y hasta en la nube.

Contenedores de Docker

Un contenedor de Docker es instancia que tiene todo lo que requiere la aplicación para ejecutarse es decir puede tener código, bibliotecas y configuraciones.

Imágenes de Docker

La imagen de Docker es un paquete liviano que contiene todo lo necesario para ejecutar la aplicación. Las imágenes se utilizan para crear los contenedores además estas se pueden subir a Docker Hub para poder ser compartidas con otros.

Dockerfile

El Dockerfile es un archivo que tiene todas las instrucciones para construir la imagen de Docker. Este archivo nos permitirá definir el entorno de ejecución, incluir archivos y dependencias necesarias además de configuraciones necesarias para la imagen.

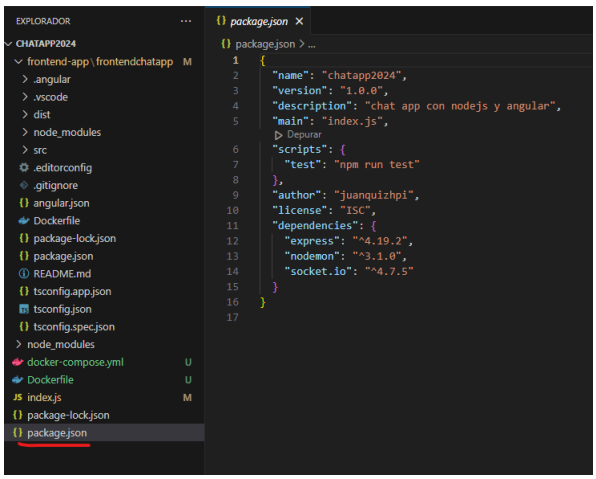
Docker Compose

Esta es una herramienta que ayuda a definir y gestionar aplicaciones multi-contenedor. Con ayuda de un archivo YAML se pude definir los servicios, redes y volúmenes que requerirá nuestra aplicación. Esto facilita el despliegue y gestión de aplicaciones complejas.

Desarrollo del proyecto

Backend

Se usará express y socket.io para la implementación del chat



En el archivo index.js se realiza la configuración del servidor

```
index.js M
index.js > ...
const {Server} = require('socket.io')

const server = new Server({cors: {origin:'http://localhost:4200'}});

server.on('connection',(socket) => {
    console.log('connected');
    socket.on('message',(data)=> {
        console.log(data);
        socket.broadcast.emit('received',{data: data, message:'Este es un test de mensaje del servidor'});
    })
})

server.listen(4000);
```

FrontEnd

Componente para el chat

En esta parte se realiza la creación del chat el cual recorrerá los mensajes de la lista de mensajes y los agrega al <li>. Además, se crea un pequeño formulario que contiene los botones para iniciar el usuario y enviar los mensajes al servidor.

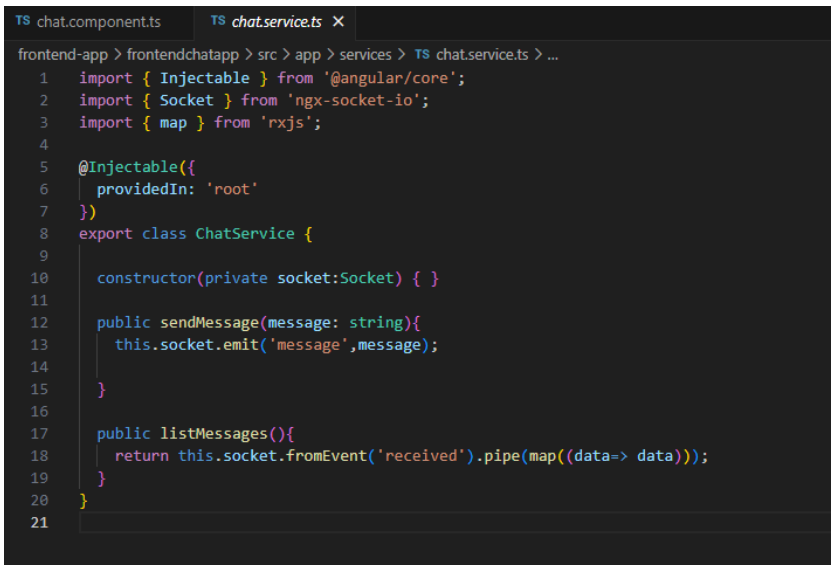
```
<div class="container mt-5">
    <h2>Conversación</h2>
    <div class="card">
      <div class="card-body">
        <ul class="list-group">
          <li class="list-group-item" *ngFor="let message of messages">{{message}}</li>
        </ul>
      </div>
    </div>

    <div class="row justify-content-center">
      <div class="col-md-6">
        <form>
          <div class="form-group">
            <input type="text" name="usuario" [(ngModel)]="usuario" [disabled]="inputBlocked" class="form-control">
            <button class="btn btn-primary" (click)="blockInput()">Iniciar Usuario</button>
          </div>
          <div class="form-group">
            <input type="text" name="message" [(ngModel)]="message" class="form-control">
            <button class="btn btn-primary" (click)="sendMessage()">Enviar</button>
          </div>
        </form>
      </div>
    </div>
  </div>
```

Ahora en se procede a usar los servicios del chat y se usa sus métodos como lo son enviar y listar los mensajes. Para cuando se pulsa el botón de iniciar usuario se bloque el input para que los mensajes tengan como usuario lo último que se escribió en el input de usuario. Esto se logra con el método de blockInput ().

```ts
import { Component, OnInit } from '@angular/core';
import { ChatService } from '../../services/chat.service';

@Component({
  selector: 'app-chat',
  templateUrl: './chat.component.html',
  styleUrl: './chat.component.css'
})
export class ChatComponent implements OnInit {

  //Creamos al usuario
  public usuario: string= '';
  //Para incio de sesión
  inputBlocked: boolean = false; // inicializa el estado de bloqueo del input
  //Para los mensajes
  public message: string = '';
  public messages:any = [];

  ngOnInit(): void {
    this.listMessages();
  }

  constructor(private chatService: ChatService){

  }

  public sendMessage(){
    //Pone en el otro usuario
    this.chatService.sendMessage(this.usuario+' : '+this.message);
    //pone en el que envia
    this.messages.push(this.usuario+' : '+this.message)
    this.message='';
  }

  public listMessages(){
    this.chatService.listMessages().subscribe((data : any)=>{
      console.log(data);
      this.messages.push(data.data);
    })
  }

  //Metodo para bloque input
  blockInput() {
    this.inputBlocked = !this.inputBlocked; // cambia el estado de bloqueo
  }

}
```

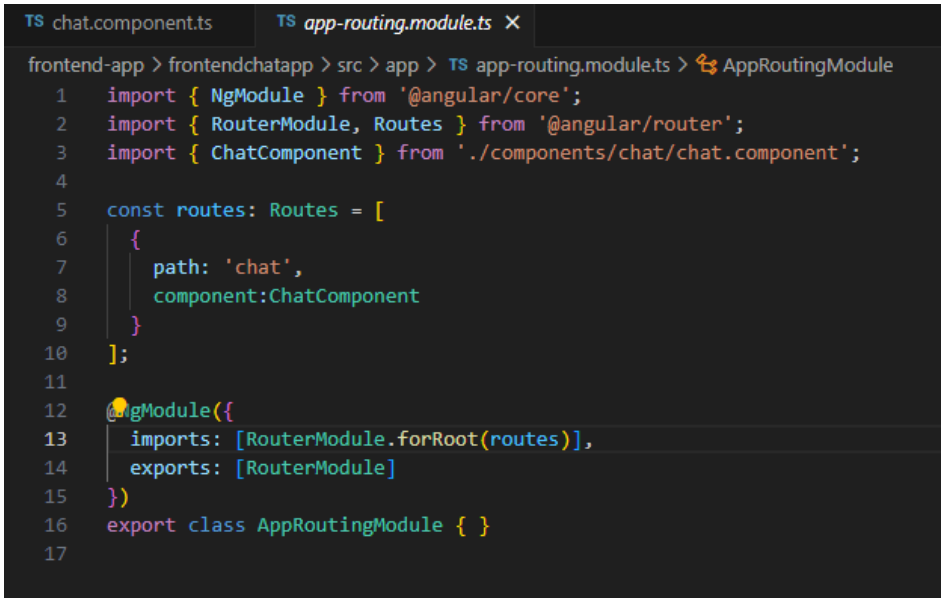
Servicio para el chat

En chat services se realiza el uso de sockets para el método de enviar mensajes y listar los mensajes.



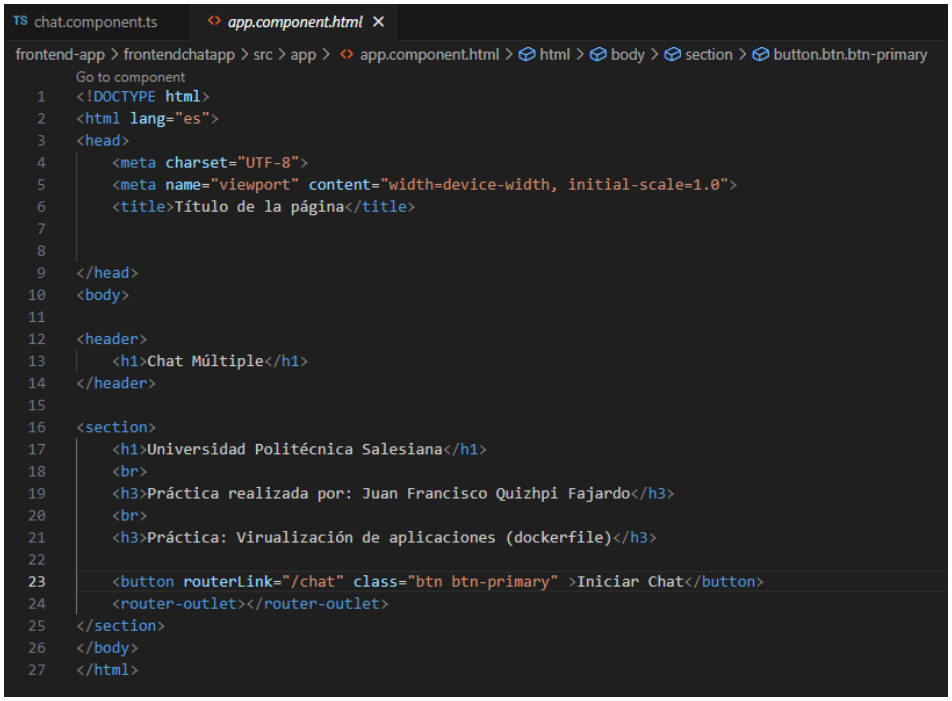
Rutas para el componente chat

De crea la ruta para el componente chat para agregarlo a nuestra página



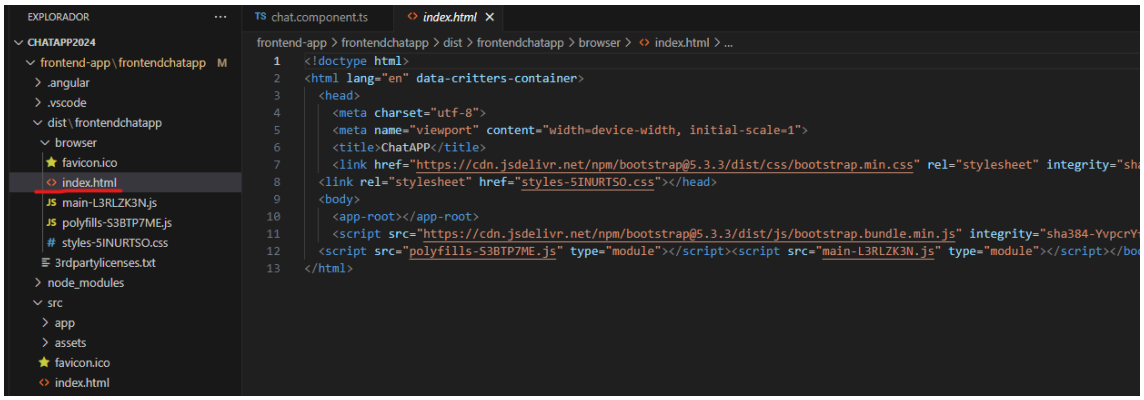
Página principal del chat

Se diseña la página y se agrega un botón para cargar el componente del chat.



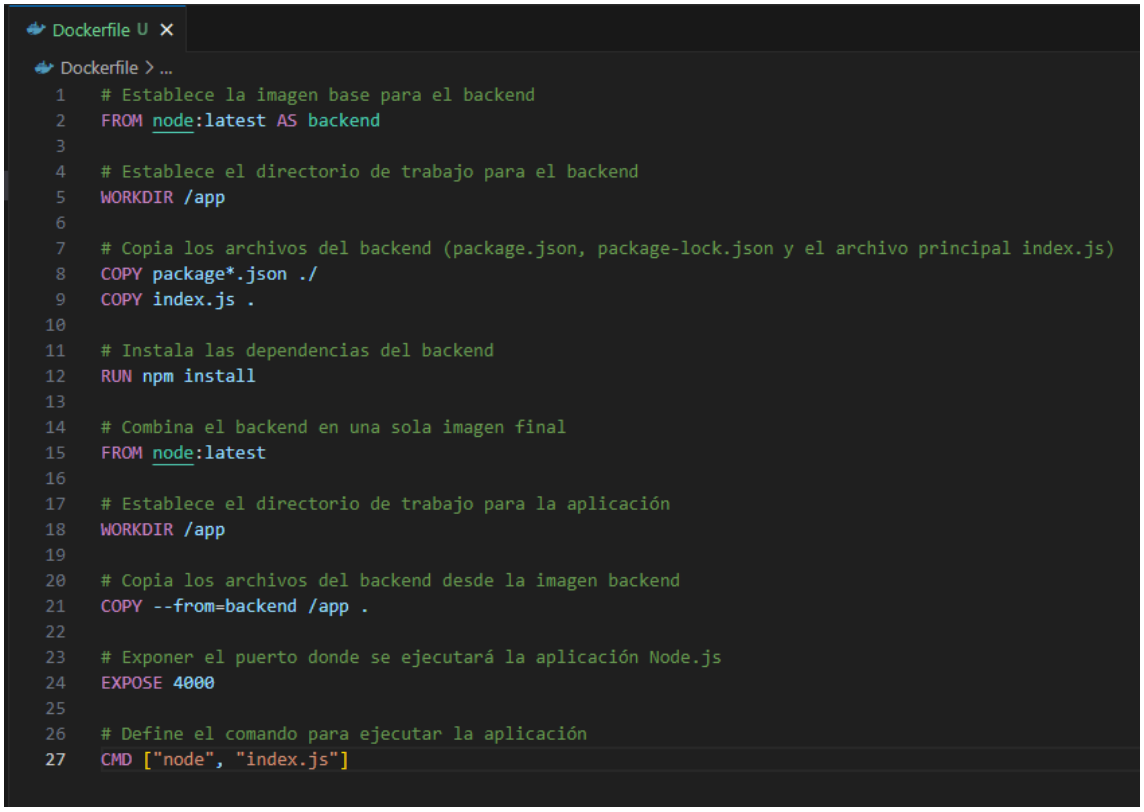
Producción de nuestra aplicación

Para poder crear el docker para el angular requerimos de ejecutare el comando ng build --prod con esto obtenemos la carpeta dist que contiene archivos que usaremos más adelante en la configuración del Dockerfile



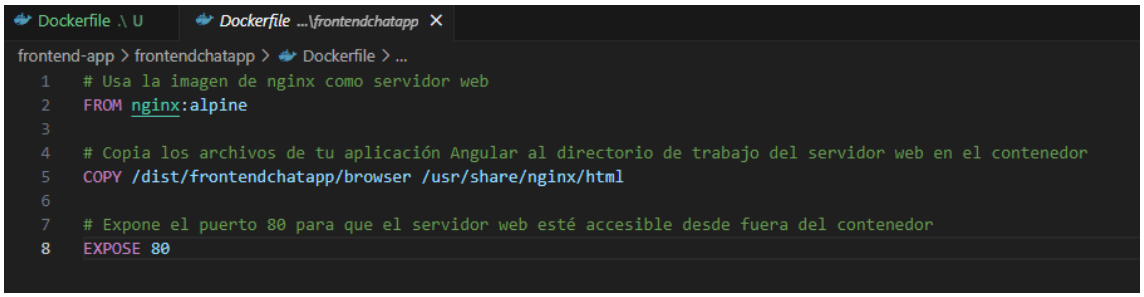
Docker File para el Backend

Para la creación del Backend se copian los archivos .json y index.js y luego se instalan dependencias necesarias. Y luego se expone en el puerto 4000 esto por la configuración de nuestro archivo index.js



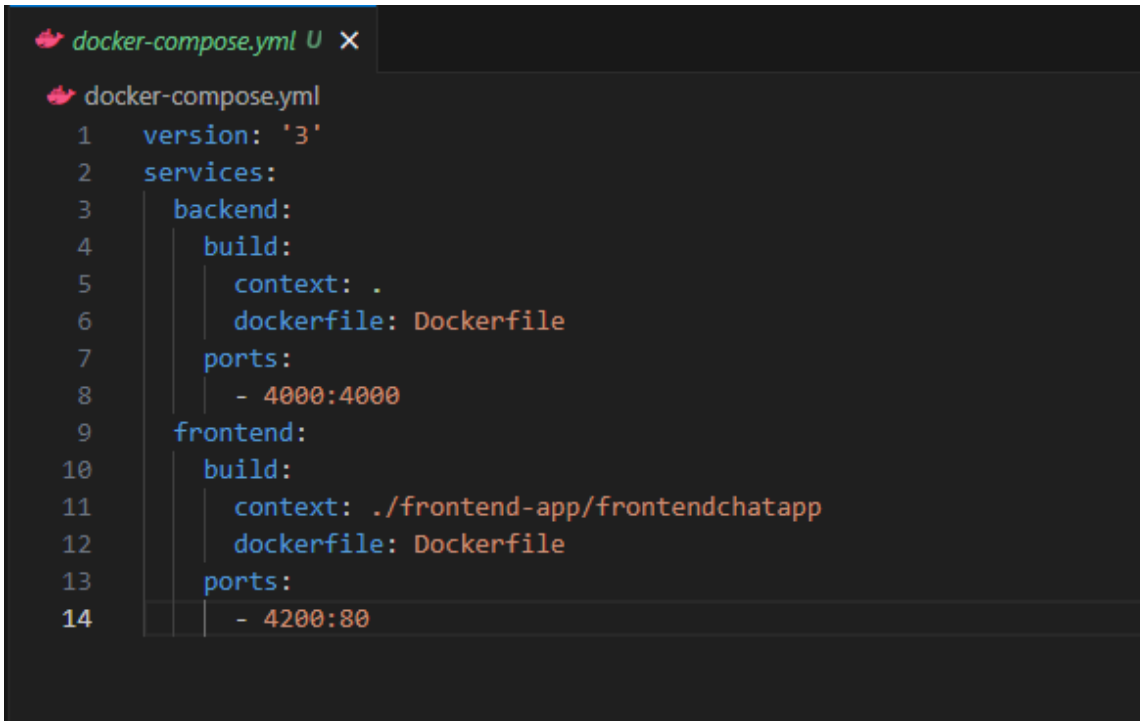
Dockerfile para Frontend

Se crea el Dockerfile para copiando los archivos de la carpeta dist en el servidor nginx y se expone en el puerto 80 para poder usarlo en nuestro navegador.



Configuración de Docker Compose

Se configura el archivo docker-compose este tiene en especial los puertos los cuales irán dado de la mano de la configuración realizada desde un principio. Además, para el segundo Dockerfile lo deberemos buscar esto se realiza en el context.



Creación de imágenes de Docker para Backend y Frontend

Se usa los comandos run y se verifica la creación además de usar los comandos push para subir estas imágenes a Docker Hub.

```
PS C:\Users\juanf\OneDrive\Escritorio\chatapp2024> docker images
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
juanqqq/backchatfinal       latest    faa64da1931d   5 hours ago     1.12GB
juanqqq/frontchatfinal      latest    edd58c9da9f8   7 hours ago     48.6MB
juanqqq/miwebjq             latest    5256b05b4d1c   5 days ago      201MB
henrytacuri/miwebht         latest    3ce498ba9bec   5 days ago      188MB
nginx                       latest    1d668e06f1e5   10 days ago     188MB
ubuntu                      latest    bf3dc08bfed0   2 weeks ago     76.2MB
ubuntu                      22.04     52882761a72a   2 weeks ago     77.9MB
juanqqq/dockerback          latest    fe74a0a1f06c   3 months ago    828MB
juanqqq/proyectodesp        latest    8087a2431f8f   4 months ago    4.92GB
jorgesayago/serpdocker      latest    eff3fd96d9d3   4 months ago    42.6MB
juanqqq/jqcalculadoraimc    latest    562a7f6fc101   4 months ago    916MB
juanqqq/calcimc             latest    53a58e53c9e6   4 months ago    916MB
kaarjoseph/unsa             latest    3fff5620f2e1   5 months ago    198MB
juanqqq/tareaclase          latest    747255d8eea1   5 months ago    176MB
juanqqq/prueba              latest    af79ca8e93f4   5 months ago    1.02GB
httpd                       latest    a6ca7b52a415   5 months ago    168MB
hello-world                 latest    9c7a54a9a43c   12 months ago   13.3kB
PS C:\Users\juanf\OneDrive\Escritorio\chatapp2024>
```

Ejecución de Docker-Compose

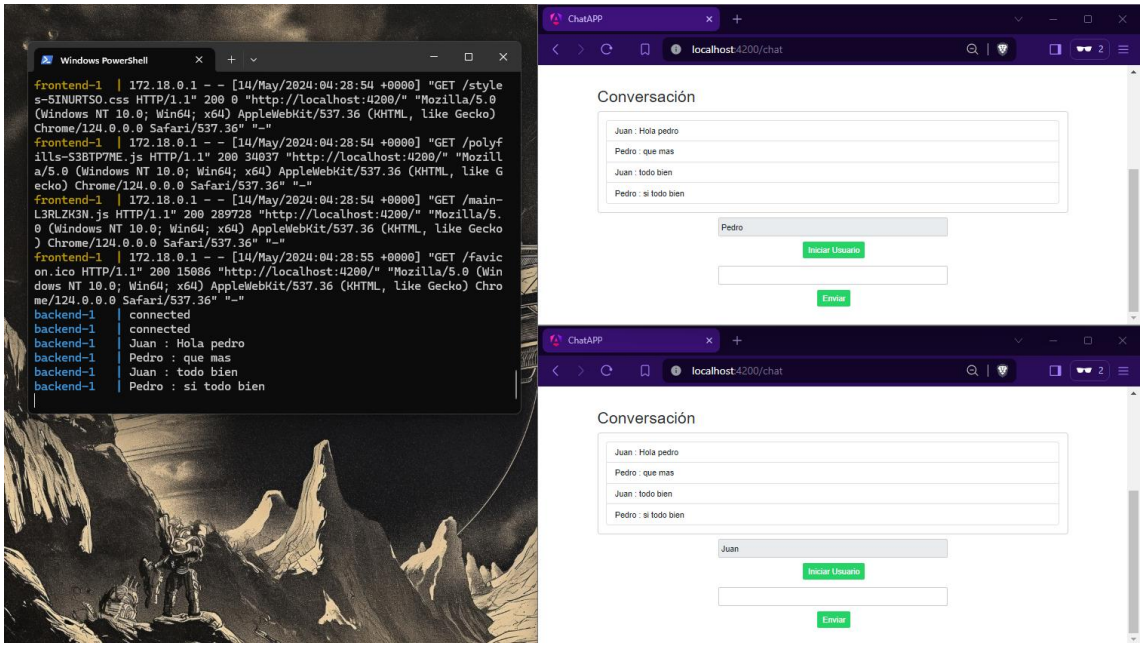
Nos ubicamos en la carpeta donde se encuentra el archivo docker compose y ejecutamos el comando docker-compose up y nos quedara de la siguiente manera.

```
PS C:\Users\juanf\OneDrive\Escritorio\chatapp2024> docker-compose up
2024/05/13 23:27:01 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 11.9s (20/20) FINISHED                                                                          docker:default
```

```
[+] Running 2/2
 ✓Container chatapp2024-frontend-1  Created                                                                          0.0s
 ✓Container chatapp2024-backend-1   Recreated                                                                        0.2s
Attaching to backend-1, frontend-1
frontend-1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
frontend-1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
frontend-1  | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
frontend-1  | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
frontend-1  | /docker-entrypoint.sh: Configuration complete; ready for start up
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: using the "epoll" event method
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: nginx/1.25.5
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: built by gcc 13.2.1 20231014 (Alpine 13.2.1_git20231014)
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker processes
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 22
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 23
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 24
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 25
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 26
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 27
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 28
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 29
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 30
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 31
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 32
frontend-1  | 2024/05/14 04:27:15 [notice] 1#1: start worker process 33
frontend-1  | 172.18.0.1 - - [14/May/2024:04:27:26 +0000] "GET / HTTP/1.1" 200 858 "-" "Mozilla/5.0 (Windows NT 10.0; Wi
e/124.0.0.0 Safari/537.36 Edg/124.0.0.0" "-"
frontend-1  | 172.18.0.1 - - [14/May/2024:04:27:26 +0000] "GET /styles-5INURTSO.css HTTP/1.1" 200 0 "http://localhost:42
ebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36 Edg/124.0.0.0" "-"
frontend-1  | 172.18.0.1 - - [14/May/2024:04:27:26 +0000] "GET /polyfills-S3BTP7ME.js HTTP/1.1" 200 34037 "http://localh
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36 Edg/124.0.0.0" "-"
frontend-1  | 172.18.0.1 - - [14/May/2024:04:27:26 +0000] "GET /main-L3RLZK3N.js HTTP/1.1" 200 289728 "http://localhost:4
eWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36 Edg/124.0.0.0" "-"
frontend-1  | 172.18.0.1 - - [14/May/2024:04:27:26 +0000] "GET /favicon.ico HTTP/1.1" 200 15086 "http://localhost:4200/"
t/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36 Edg/124.0.0.0" "-"
backend-1   | connected
```

Prueba de Chat

Si todo sale correcto cuando ejecutemos el comando podremos usar el chat cuando se inicie el angular podremos ver como se conectan al servidor además que los mensajes que se envían los clientes.



Repositorios de código

Backend

https://github.com/JuanQuizhpi/chatapp2024.git

Frontend

https://github.com/JuanQuizhpi/frontendchatapp.git

Repositorios de Dockerfile

Backend

docker pull juanqqq/backchatfinal

Frontend

docker pull juanqqq/frontchatfinal