FrontEnd (Angular) + BackEnd (Node+Express)

Este es un proyecto muy sencillo donde se ve un ejercicio completo Full Stack con Angular, Node.js y MySql

He intentado centrarme en lo fundamental, aquello sin lo cual el proyecto no funciona. He dejado de lado la utilización de librerían muy interesantes (nodemon, dotenv) pero que no son determinantes para entender el funcionamiento del Full Stack Proyect.

FrontEnd (Angular)

Hemos ido a lo mínimo imprescindible. Hemos creado un proyecto de Angular llamado front

Paso 1: Crear el Proyecto de Angular

Abre una terminal y ejecuta el siguiente comando para crear un nuevo proyecto Angular:

ng new back

Cuando se te pregunte si deseas incluir el routing, elige Yes y selecciona CSS o el preprocesador de estilos que prefieras.

Paso 2: Crear el Componente listadeautores como Standalone

Navega dentro del proyecto:

cd proyecto-autores

Crea el componente standalone listadeautores con el siguiente comando:

ng generate component listadeautores

Esto creará un componente independiente que no necesita ser parte de un módulo.

Paso 3: Implementar la Estructura del Componente

Abre el archivo listadeautores.component.ts y asegúrate de que tu componente se vea como sigue:

listadeautores.ts

```
import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http'; // Importar HttpClient para hacer la petición HTTP
import { CommonModule } from '@angular/common';

@Component({
  standalone: true,
  selector: 'app-listadeautores',
  templateUrl: './listadeautores.component.html',
  styleUrls: ['./listadeautores.component.css'],
  imports: [CommonModule]
})
export class ListadeautoresComponent {
  autores: any[] = [];

  constructor(private http: HttpClient) {}

  obtenerAutores() {
    const url = “http://localhost:3000/api/autores”; // Cambia esto por la URL de tu backend
    this.http.get<any[]>(url).subscribe({
      next: (data) => {
        this.autores = data; // Almacenar los datos recibidos
      },
      error: (error) => {
        console.error('Error al obtener los datos:', error);
```

```
    }
  });
 }
}
```

Paso 4: Crear el HTML del Componente

En el archivo listadeautores.component.html, crea el botón y la tabla para mostrar los datos:

listadeautores.html

```
<div class="container">
 <button (click)="obtenerAutores()" class="boton">Obtener Autores</button>

 <table *ngIf="autores.length > 0" class="tabla">
  <thead>
   <tr>
    <th>Nombre</th>
    <th>Email</th>
    <th>Imagen</th>
   </tr>
  </thead>
  <tbody>
   <tr *ngFor="let autor of autores">
    <td>{{ autor.nombre }}</td>
    <td>{{ autor.email }}</td>
    <td>{{ autor.imagen }}</td>
   </tr>
  </tbody>
```

```
  </table>
</div>
```

Paso 6: Configurar el Servicio HTTP

Asegúrate de haber importado el HttpClientModule para realizar peticiones HTTP. Esto se hace en el archivo main.ts ya que estás trabajando con componentes standalone.

Abre el archivo main.ts y asegúrate de que se importe el HttpClientModule:

Main.ts

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app/app.component';
import { provideHttpClient } from '@angular/common/http';

bootstrapApplication(AppComponent, {
  providers: [provideHttpClient()]
});
```

BackEnd (Node + Express)

Para crear el back primero creamos un proyecto de Node

Abre una terminal y ejecuta el siguiente comando para crear un nuevo proyecto Angular. Esto nos creará un archivo package.json que nos permitirá configurar las dependencias.

```
npm init -y
```

package.json

```
{
  "name": "actividad8definitiva",
  "version": "1.0.0",
```

```
 "main": "index.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [],
 "author": "",
 "license": "ISC",
 "description": "",
 "dependencies": {
  "cors": "^2.8.5",
  "express": "^4.21.0",
  "mysql2": "^3.11.3",
  "nodemon": "^3.1.7"
 }
}
```

Luego configuro el servidor en index.js

index.js

```
const express = require('express');
const app = express();
const port = 3000;
const userRoutes = require('./routes'); // Importar las rutas
const db = require('./db'); // Asegurar la conexión a la base de datos


//El error que estás viendo es causado por una política de seguridad de los navegadores conocida como CORS
//(Cross-Origin Resource Sharing). Esta política impide que una aplicación en un dominio (como
```

http://localhost:4200)

//acceda a recursos en otro dominio (como http://localhost:3000) sin la autorización adecuada.

```
const cors = require('cors');

// Middleware para procesar datos JSON

app.use(cors());

// Rutas base de la API

app.get('/', (req, res) => {

  res.send('API de Express con MySQL');

});

// Usar las rutas definidas

app.use('/api', userRoutes);

app.listen(port, () => {

  console.log(`Servidor corriendo en http://localhost:${port}`);

});
```

Luego configuramos la conexión en db.js

db.js

```
const mysql = require('mysql2');

// Crear la conexión a la base de datos

const connection = mysql.createConnection({

  host: 'localhost',

  user: 'root',      // Cambia al usuario de tu base de datos

  password: '1234', // Cambia a la contraseña de tu base de datos

  database: 'blog_unir'

});

// Conectar a la base de datos
```

```
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar con MySQL:', err);
  } else {
    console.log('Conectado a MySQL');
  }
});
```

```
module.exports = connection;
```
//Exporto la conexión para que la podamos reutilizar

Ya por último creamos la bbdd Blog_Unir en el servidor de MySQL

La base de datos está vacía, para que funcione con el GET es necesario que metas algún datos.

```
-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERR
OR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';



-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema blog_unir

-- -----------------------------------------------------



-- -----------------------------------------------------
```

```
-- Schema blog_unir

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `blog_unir` DEFAULT CHARACTER SET utf8mb3 ;

USE `blog_unir` ;


-- -----------------------------------------------------

-- Table `blog_unir`.`autores`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `blog_unir`.`autores` (

  `id` INT NOT NULL AUTO_INCREMENT,

  `nombre` VARCHAR(45) NOT NULL,

  `email` VARCHAR(45) NOT NULL,

  `imagen` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)

ENGINE = InnoDB

AUTO_INCREMENT = 22

DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------

-- Table `blog_unir`.`posts`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `blog_unir`.`posts` (

  `id` INT NOT NULL AUTO_INCREMENT,
```

```
  `titulo` VARCHAR(45) NOT NULL,
  `descripcion` VARCHAR(450) NOT NULL,
  `fecha` DATE NOT NULL,
  `categoria` ENUM('filosofia', 'arte', 'naturaleza') NOT NULL,
  `autores_id` INT NOT NULL,
  PRIMARY KEY (`id`, `autores_id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_posts_autores_idx` (`autores_id` ASC) VISIBLE,
  CONSTRAINT `fk_posts_autores`
    FOREIGN KEY (`autores_id`)
    REFERENCES `blog_unir`.`autores` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 87
DEFAULT CHARACTER SET = utf8mb3;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```