Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Ingeniería en Ciencias y Sistemas

Practicas Iniciales

Sección C Grupo 3

Manual Técnico

Anyelo Gustavo Hernández Ayala 201807398

Victor Eduardo Franco Suret 202001767

Guatemala, 22 septiembre de 2022

Instalación de las dependencias:

Para instalar las dependencias para poder correr este proyecto se debe ingresar a la carpeta server desde una terminal y escribir el siguiente comando:

npm i

Luego podemos correr el servidor node.js con:

npm run dev

Una vez hecho esto, abrimos otra terminal para poder instalar las dependencias de Angular, para eso ingresamos a la carpeta Front desde una terminal y volvemos a escribir:

npm i

Y para abrir el servidor de Angular:

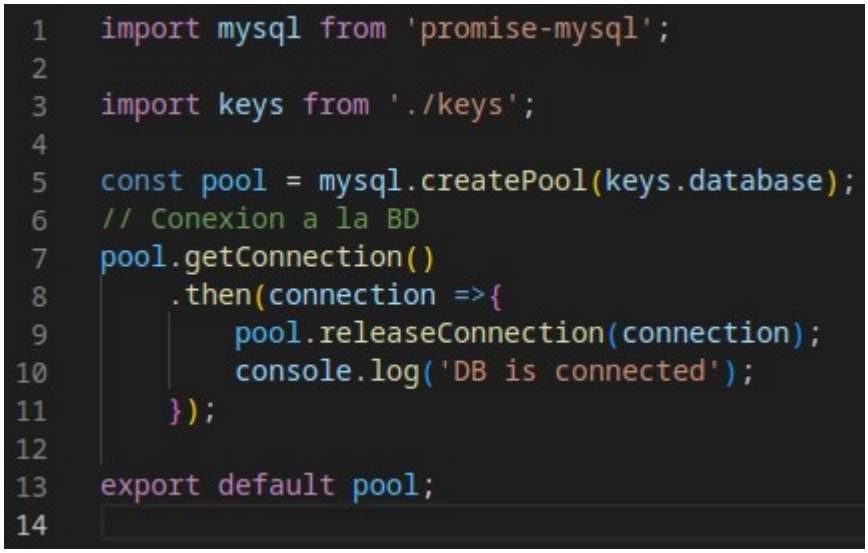
ng serve -o

Bases del proyecto:

El proyecto esta hecho en el framework de Angular, utilizando mysql, node.js, express, y nodemon. Angular nos permite desarrollar más eficazmente el frontend y conectar de una manera sencilla con el backend.

Funciones del backend:

En el backend se realizó la conexión a la base de datos mediante la librería de promise-mysql y se importo keys del archivo keys.ts en el cual están los datos de autenticación para usar el servidor mysql.



```
import mysql from 'promise-mysql';

import keys from './keys';

const pool = mysql.createPool(keys.database);
// Conexion a la BD
pool.getConnection()
    .then(connection =>{
        pool.releaseConnection(connection);
        console.log('DB is connected');
    });

export default pool;

```

También se realizaron las consultas a la base de datos, estas están ubicadas en la carpeta controllers. Como por ejemplo:

```
//  Obtener UNA publicacion
public async getPublicacion(req: Request, res: Response): Promise<any>{
    const { id } = req.params;
    const publicaciones = await pool.query('SELECT * FROM publicaciones WHERE id = ?', [id]);
    if (publicaciones.length > 0) {
        return res.json(publicaciones[0]);
    }
    res.status(404).json({text: "La publicacion no existe"});
}
```

También estan las consultas de DELETE, POST, PUT.

Routes:

Para consumir la API se crearon los routes en la carpeta routes, estos permiten consumir la API para poder accesar a la base datos.

```
import { Router } from 'express';
import { homeController } from '../controllers/homeController';
class HomeRoutes {

    public router: Router = Router();

    constructor() {
        this.config();
    }

    config(): void {
        this.router.get('/', homeController.list); // Obtiene una publicacion
        this.router.get('/:id', homeController.getPublicacion); // Obtiene UNA publicacion
        this.router.post('/', homeController.create); // Crea una nueva publicacion
        this.router.delete('/:id', homeController.delete); // Elimina una publicacion
        this.router.put('/:id', homeController.update); // Actualiza una publicacion
    }

}

const homeRoutes = new HomeRoutes();
export default homeRoutes.router;
```

Funciones del Frontend:

En la parte del frontend se utilizó el framework de Angular, en este framework se utilizan componentes, estos se crearon en la carpeta de app/components, los cuales se describen a ellos mismos.

En el archivo app-routing.module.ts se crearon los routes para poder utilizar los componentes.

Los cuales son las entradas que nos rediccionará el servidor, y también sirve para que carguen sus componentes solo en esas entradas.

```
const routes: Routes = [
  {
    path: '',
    redirectTo: '/login',
    pathMatch: 'full',
  },
  {
    path: 'login',
    component: LoginComponent
  },
  {
      path: 'home',
      component: PublicacionesComponent
  },
  {
    path: 'publicar',
    component: PublicarComponent
  },
  {
    path: 'home/editar/:id',
    component: PublicarComponent
  },
  {
    path: 'regis',
    component: RegisterComponent
  }
];
```

```
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    PublicacionesComponent,
    PublicarComponent,
    NavigationComponent,
    RegisterComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [
    LoginService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Otro archivo importante es app.module.ts el cuál hace declaración de los componentes utilizados y los paquetes importados.