En la carpeta del proyecto ejecutamos el comando de angular cli para crear el proyecto en este caso se llamra client, Le decimos que nos cree el archivo de rutas y que utilice css.

>>ng new client

Una vez creado el proyecto de angular vamos a crear los componentes que vamos a necesitar.

>>ng g c components/navigation

>>ng g c components/gameForm

>>ng g c components/gameList

Ahora vamos a crear los modelos esto lo hacemos manualmente, en la carpeta app creamos la carpeta models y en ella el archivo game.ts, por ahora este archivo estará vacío.

También debemos crear los servicios que son los que van a consumir recursos de nuestro backend.

>>ng g s services/games

Bien una vez ya tenemos los componentes que necesitamos vamos a iniciar el servidor y borramos el código de ejemplo del app.component.html, vamos a poner nuestro componente de navegación y el router outlet para ver en pantalla las rutas que se definan.

Ahora vamos a instalar Bootstrap para ello vamos al index.html y copiamos el cdn de la página de Bootstrap, ahora vamos a la documentación de Bootstrap y copiamos el navbar llamado nav.

Una vez tenemos lista la navegación vamos a configurar las rutas.

Después de tener las rutas vamos a iniciar a consumir los servicios del res api para ello vamos a los servicios, para ello lo primero es importar los servicios en el app.module.ts, y lo establecemos en el providers, también debemos importar el módulo de http client

import { GamesService } from './services/games.service';

providers: [

    GamesService

  ],

Ahora vamos al servicio e importamos el módulo de http para poder hacer las peticiones a nuestro servidor.

import { HttpClient } from '@angular/common/http';

y lo inyectamos en el constructor

constructor( private http: HttpClient ) {

  }

Ahora vamos a crear los métodos get, post, put y delete pero antes vamos a extraer la dirección en una propiedad para evitar escribirla varias veces.

API\_URI = 'http://localhost:3000/api';

Ahora vamos a crear los métodos y utilizamos esta dirección para ello vamos a cambiar las comillas simples por back tips.

getGames(){

    return this.http.get(`${this.API\_URI}/games`);

  }

Ahora vamos a crear los otros métodos.

El método de obtener un juego recibe como parámetro el id del juego y se hace mediante la petición games/id

getOneGame( id: string ){

    return this.http.get(`${this.API\_URI}/games/${id}`);

  }

El método de guardar un juego recibe como parámetro el objeto del juego este lo vamos a definir en el modelo game.ts que creamos.

export interface Game {

    id?: number;

    title?: string;

    description?: string;

    image?: string;

    created\_at?: Date;

    updated\_at?: Date;

}

Ahora en el servicio debemos importar esta interfaz para poder utilizar el tipo de dato.

import { Game } from '../models/game';

el método de guardar un nuevo juego se ejecuta por medio de la petición post esta siempre va a pedir el body en este caso es el game.

saveGame( game: Game ){

    return this.http.post(`${this.API\_URI}/games`, game );

  }

Vamos a crear el método de eliminar.

deletegame( id: string ){

    return this.http.delete(`${this.API\_URI}/games/${id}`);

  }

Y por último el método de actualizar, este recibe como parámetros el id y el juego con las propiedades que se van a actualizar.

updateGame( id: string, updateGames: Game ){

    return this.http.put(`${this.API\_URI}/games/${id}`, updateGames );

  }

**UTILIZAR EL SERVICIO GAMES**

Pata utilizar nuestro servicio debemos importarlo en los módulos que lo vamos a necesitar, para ello creamos una propiedad en el constructor de tipo GamesService y lo importamos desde Games service.

import { GamesService } from '../../services/games.service';

constructor( private gamesService: GamesService ) { }

ahora ya podemos hacer uso de el servicio para ello en el ngOnInit vamos a llamar al método del servicio getGames().

ngOnInit(): void {

    this.gamesService.getGames()

  }

Este método nos retorno un observable por lo que puede darnos la respuesta o un error, para obtener esta información debemos suscribirnos esto no puede dar la respuesta o un error vamos a imprimir ambos casos en consola.

ngOnInit(): void {

    this.gamesService.getGames().subscribe(

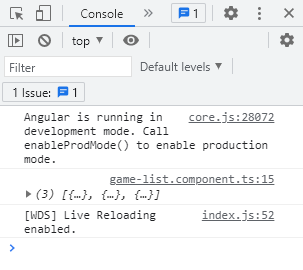
      resp => console.log(resp),

      err => console.error(err)

      );

  }

Podemos ver que en la consola ya tenemos el arreglo con los juegos.



Ya tenemos el arreglo en consola, pero necesitamos almacenarla para ello vamos a crear un arreglo e juegos.

ngOnInit(): void {

    this.gamesService.getGames().subscribe(

      //resp => console.log(resp),

      (res:any) => {

        this.games = res;

        console.log(this.games);

      },

      err => console.error(err)

      );

  }

Ya tenemos el arreglo almacenado en una propiedad ahora podemos utilizar el arreglo de juegos para usarla en el html.

3:6:0