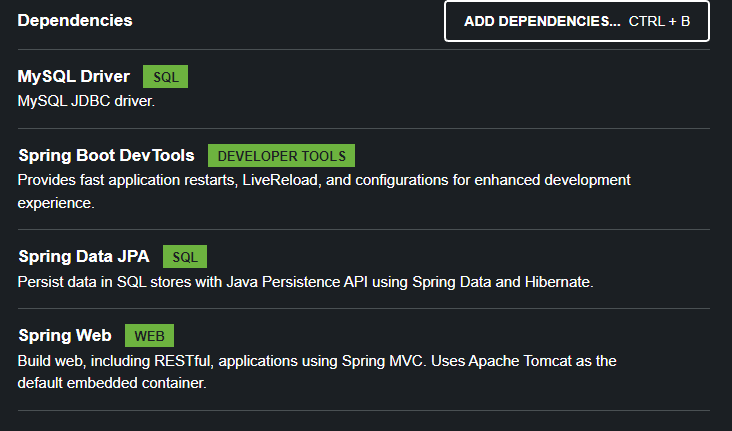


1 - Creamos el proyecto desde la web de Sprinboot y agregamos las dependencias que aparecen en -pantalla

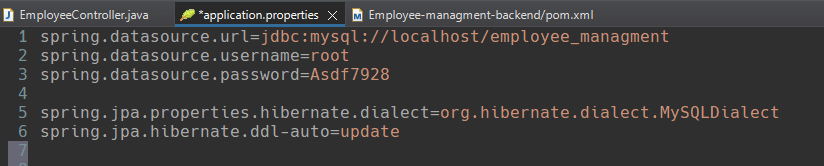
1. Desde MYSQL vamos a crear la base de datos y comenzar a utilizarla con los siguientes comandos

Create database “nombre de la db”

Use “nombre de la db”

(En este caso sería “create database employee managment”)

1. En Eclipse vamos a src/main/resources > application.properties y vamos a configurar la DB



\*1 – Es la dirección de url a la base de datos. En este caso es localhost y la siguiente sentencia es el nombre de la db

\*2 – Creería que es el nombre del user de MYSQL, lo estoy probando

\*3 – Mismo

\*5 – Hace referencia al dialecto utilizado  (Hay que revisar que sea compatible con la version deMySQL)

\*6 – Hace que cuando se abra la app lo primero que haga sea el update

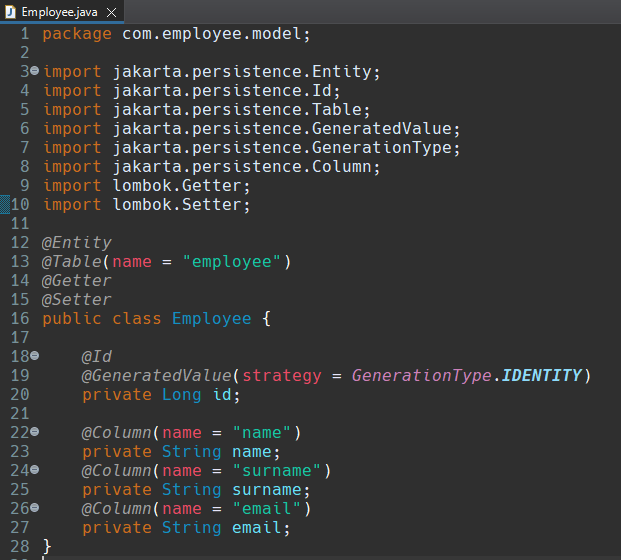
4 – En la carpeta src/main/java creamos el paquete MODEL para crear las clases a utilizar. En este caso Employee   
A la clase le agregamos las anotaciones @Entity y @Table (name = “employee”). Importamos ambas anotaciones (en name = employee es para marcar a que tabla de la DB va a hacer referencia)   
Le agregamos ya dentro de la clase:

@Id   
@GeneratedValue(stratefy = GenerationType.IDENTITY)

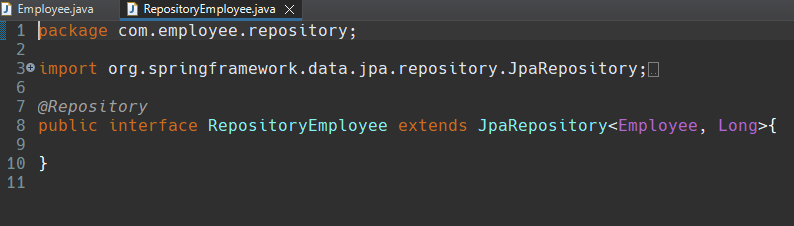
El ID es para marcar que va a ser la primary key y el generated... es para hacer que se auto incremente

Luego creamos los atributos privados e importamos la dependencia lombok para poder utilizar las anotaciones de getter y setter (los getters y setters tienen que estar por encima de la public class Employee(

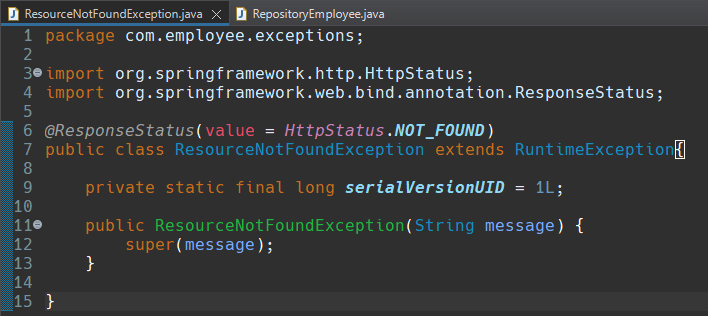
<**dependency**>   
 <**groupId**>org.projectlombok</**groupId**>   
 <**artifactId**>lombok</**artifactId**>    
 <**scope**>provided</**scope**>    
</**dependency**>



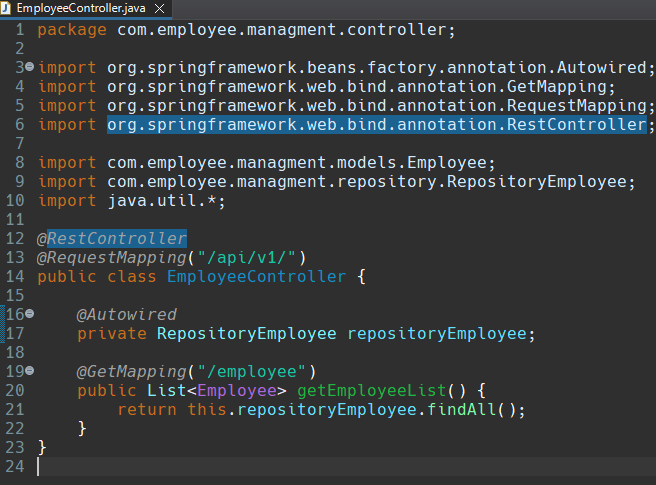
5 – En la carpeta src/main/java creamos el paquete com.employee.repository y creamos la interfaz RepositoryEmployee (esta se encarga de gestionar todas las operaciones de persistencia contra una tabla de la base de datos, es decir, es la que va a guardar y recuperar la información de la base de datos, por eso al extenderla de JpaRepository hay que pasar en los diamantes, la clase y el tipo de pk)   
A la interfaz, le agregamos la anotacion de @Repository   
Luego le agregamos “extends JpaRepository<Employee, Long>”   
Employee va a ser la clase que trabaja y Long es el ID o key   
Importamos la herencia de Jpa



6 – Ahora vamos a crear el paquete para la excepciones “com.employee.exceptions” (tambien  dentro de main/java)   
Creamos la clase ResourceNotFoundException   
Le agregamos la anotación @ResponseStatus(value HttpStatus.NOT\_FOUND) y le agregamos la extension de RuntimeException   
Creamos un atributo privado static final long “serialVersionUID = 1L;”   
Luego creamos el constructor que recibe un String “message” desde la clase super



7 – Creamos el paquete controller y adentro la clase EmployeeController.  (Controller es la clase que va a marcar las URL por las que el usuario puede interactuar con la app, va a llevar los metodos get, set, post, delete, etc)  
Le agregamos la anotación @RestController y @RequestMapping(“/api/v1/”)   
Creamos un atributo del tipo RepositoryEmployee y le agregamos la anotación @Autowired (inyección) para que sepa que esa información va a ser inyectada  
Luego creamos el metodo getEmployeeList() que retorna repositoryEmployee.findAll(); esto nos devolver la lista complete de los empleados. Se le agrega la anotación @GetMapping(“/employee”) para que en la url nos muestre el retorno (en este caso es la lista de empleados)



* + EN ESTA INSTANCIA YA PODEMOS PROBAR EL FUNCIONAMIENTO DE LA APP  
    VAMOS A MYSQL Y AGREGAMOS MANUALMENTE DOS EMPLEADOS Y DESDE LA URL <http://localhost:8080/api/v1/employee> VAMOS A VER LOS RESULTADOS

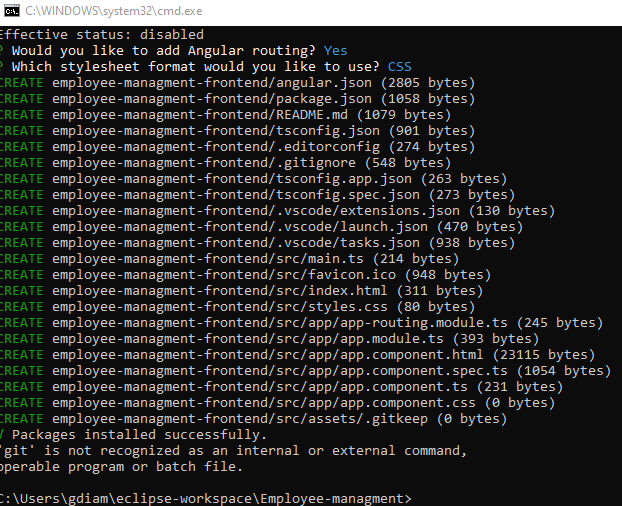
8 – Ahora vamos a crear la carpeta del FRONTEND

Primero debemos instalar angular, para hacerlo, abrimos el CMD (commandos) y parados sobre la ruta del entorno de trabajo (para esto escribimos “cd zºC:\Users\gdiam\eclipse-workspace\Employee-management”) y escribimos “npm install -g @angular/cli”

(También podremos actualizer la version de angular si es que lo deseamos)

Luego de hacer esto escribimos

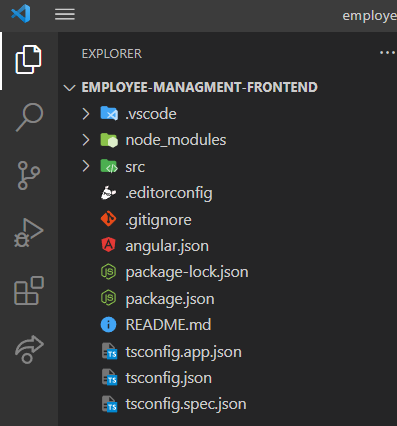
ng new employee-management-frontend y le damos enter, luego si pregunta algo de google le damos no, cuando pregunte si queremos agregar la ruta de angular ponemos que si y por ultimo al momento de elegir el stylesheet elegimos CSS



Una vez terminada la creación de la carpeta, vamos a vistual studio y hacemos

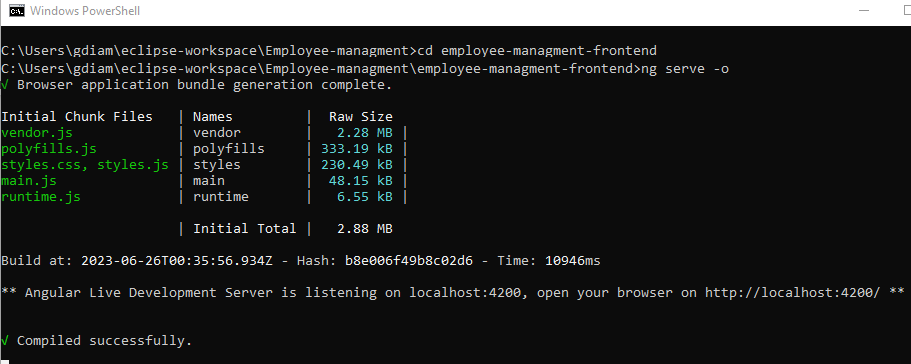
“file > open file > “ y buscamos la ruta donde se encuentra el Proyecto del frontend

C:\Users\gdiam\eclipse-workspace\Employee-managment\employee-managment-frontend



Ahora cambiamos la ruta del CMD y nos paramos dentro de la carpeta de frontend (cd employee-management-frontend) y escribimos “ng server -o” para ejecutar la aplicación (levanter el server)

(para acceder a la misma hay que ir a localhost:4200)



En VS vamos a source > app > app.component.html y borramos todo el codigo que allí se encuentra

Vamos a crear el front desde 0. Para empezar, vamos <https://getbootstrap.com/docs/4.0/migration/> luego en la barra de la izquierda seleccionamos “Getting Started” y copiamos el link de CSS

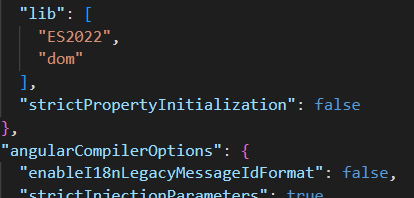
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

Este Código lo vamos a pegar en el body del index.html

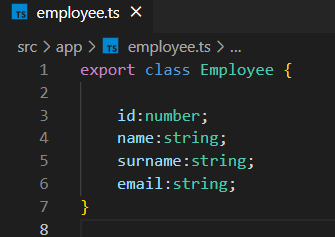
10 – Desde VS abrimos una terminal de powershell y tipeamos “ng g class employee” con esto Podemos crear la clase dentro de la app

(en caso de que haya un error para correr los scripts, tenemos que abrir como administrador desde windows el cmd de powershell y tipear lo siguiente “Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine”. Luego le damos Y y volvemos a correr el commando y le damos A y listo)

Ahora el archive tsconfig.json agregamos “strictPropertyInitialization” : false

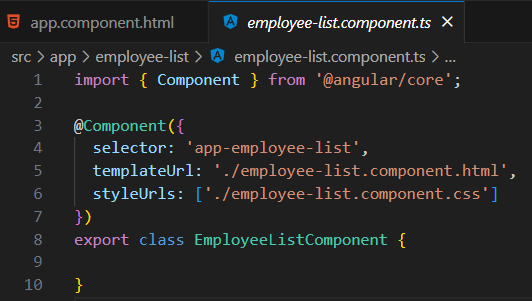


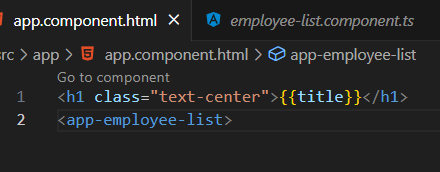
Y ahora en la clase Employee escribimos los atributos



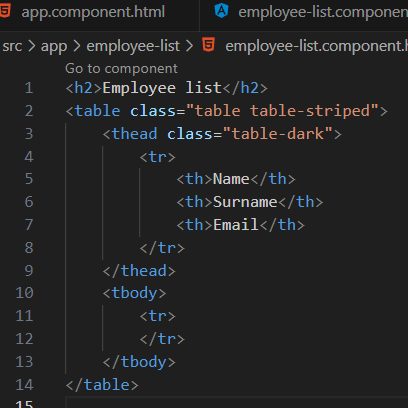
11 – Desde la terminal tipeamos “ng g c employee-list” para crear el componente del mismo nombre (la c es de component) que aparecera como una carpeta dentro de la carpeta “app”

Ahora, en el archive employee-list.component.ts copiamos el link que figura en selector y lo pegamos en el app.component.html





Vamos a Employee-list.component.html y copiamos la siguiente sintaxis



12 – Ahora vamos al archivos employee-list-component.ts y en la clase EmployeeListComponent tipeamos



Creamos una lista vacía y la llenamos con 2 empleados hardcodeados en el ngOnInit() que corre cuando se levanta la app

13 – Volvemos a employee-list.component.html y utilizamos el dialect de angular para llamar a la lista y que complete los campos con los datos de cada element de la misma

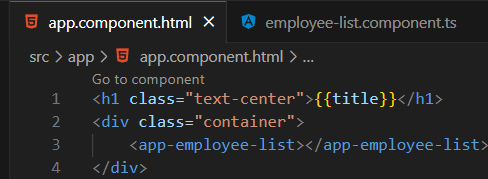


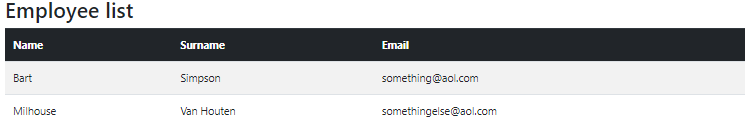
(notar las palabras clave “\*ngFor” que corre un foreach, “let” para crear la variable employee “of” employees)



Ahora ya Podemos ver la lista completada con los datos hardcodeados en la url http//localhost:4200

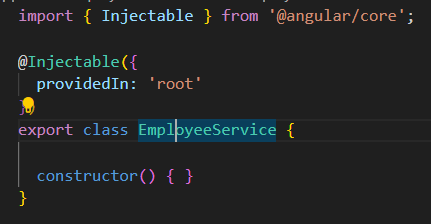
Aprovechamos la oportunidad y en app.component.html acomodamos el Código para que se vea major



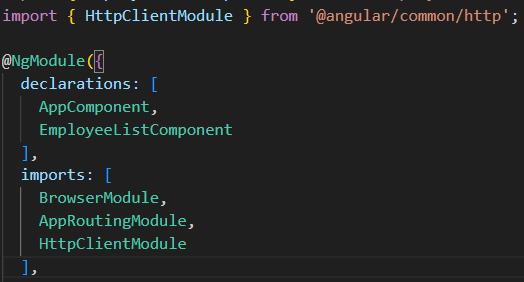


14 – Vamos a crear el servicio employee, para esto escribimos “ng g s employee” (s de service)

(hay que corroborar que en el archive employee.service.ts el servicio tenga la anotacion @injected para asegurar la inyeccion de información, y que el provideIn tenga la ‘root’ declarado



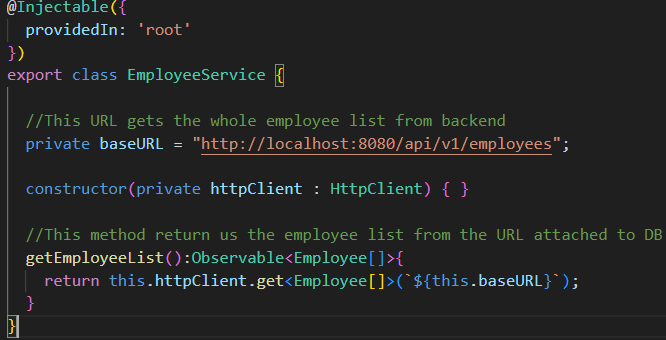
Ahora en el archive app.module.ts vamos a importar “HttpClientModule” y asegurarnos de escribir manualmente el import en la cabecera



15 – Vamos a employee.service.ts y declaramos dentro de la clase un atributo privado “baseURL” que contiene la direccion del getList del backend (CLAVE: el this.baseURL tiene que estar encerrado por ``, no pueden ser ‘’ ni ningún otro caracter…)

Luego en el constructor ponemos en los argumentos, “(private httpClient : HttpClient)” e importamos.

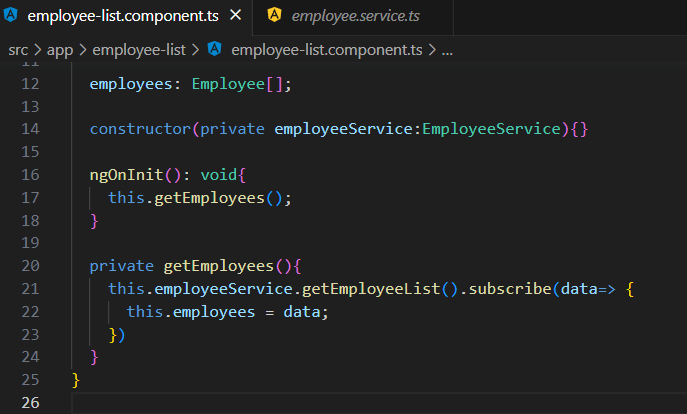
Luego creamos el metodo un metodo para retornar el arreglo de empleados



16 – Volvemos a employee.list.components.ts y al constructor de la clase le enviamos como argumentos “private employeeService:EmployeeService”

Luego creamos un metodo llamado private getEmployees() y hacemos que subscriba al obersvable de employeeService

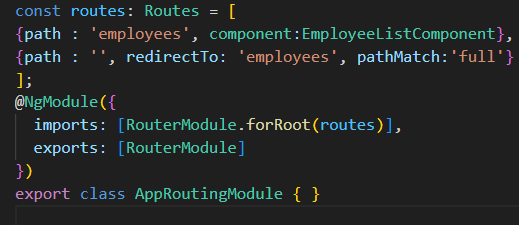
Modificamos el ngOnInit() borrando la info hardcodeada y en su lugar llamado al getEmployees()



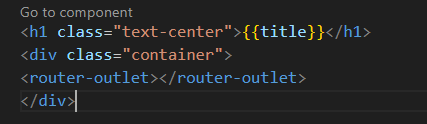
En Eclipse, al archive EmployeeController le agregamos la anotación @CrossOrigin(origins=””) e importamos.  
Esto nos va a permitir cruzar la información entre el backend y el frontend

17 – En app-routing.module.ts y vamos a cofigurar las rutas que vamos a utilizar cuando entramos con navegador

En las rutas agregamos las sigueintes lineas para redireccionar los links a la lista de empleados

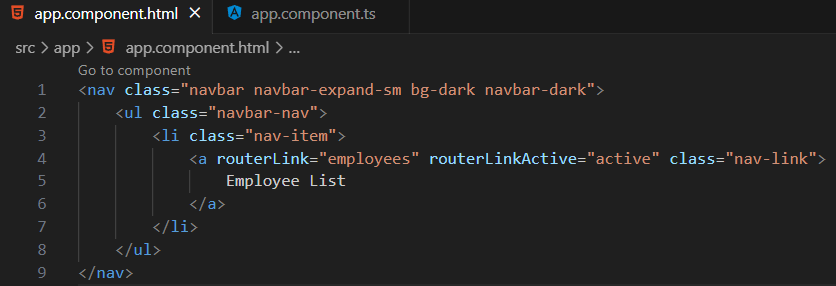


18 – en app.component.html cambiamos la línea de app-component-list por router-outlet



En app.component.ts cambiamos el title por el que querramos. Hacemos lo mismos en index.html

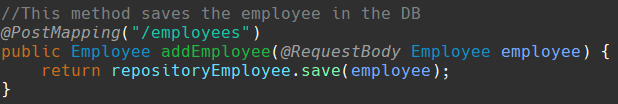
Vamos a app.component.html y previo al h1 codeamos lo siguiente



Con este Código vamos a crear una barra de navegación al comienzo de la pantalla con un tono obscuro

19 – Volvemos al backend y en el controller creamos el metodo addEmployee. Agregamos la anotación de @PostMapping y en los args le agregamos la anotación de @BodyRequest Employee employee para que sepa que por parametros debe recibir al empleados que va a grabar.

Luego, en el return lo grabamos mediant el metodo save



20 – En VS creamos una nueva terminal y vamos a crear un nuevo componente llamado add-employee

Tipeamos “ng g c add-employee” G de generate y c de component

En app-routing.models.ts agregamos el path 

Vamos a app.component.html y agregamos al nav el link para ir a add employee

 <li class="nav-item">

        <a routerLink="add-employee" routerLinkActive="active" class="nav-link">

                Add employee</a></li>

21 – En app.module.ts en la parte de las importaciones, agregamos FormModule

imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule,

    FormsModule

  ]

En add-employee.components.ts dentro de la clase creamos un nuevo empleado y un metodo llamado onSubmit() donde mostraremos por console al empleado

export class AddEmployeeComponent {

  employee: Employee = new Employee();

  constructor() { };

  ngOnInit(): void {

}

onSubmit(){

    console.log(this.employee);

  }

Ahora, en add-employee.component.html vamos a crear el form para la carga de los datos

Utilizando el ngModel en el input, vamos a cruzar los datos que se reciben desde el onSubmit() y crear el nuevo empleado para agregar de la clasee AddEmployeeComponent



Hacemos 3 divs, uno por cada atributo (name, surname, email; todo en minuscula)

(tener en cuenta que es desde el div interno y que en el div del Email el type es “email”

Por ultimo, agregamos un botón “add” para que se envién los datos

<button class="btn btn-succes"> Add </button>

En employee.service.ts voy a crear el metodo addEmployee que recibe un employee y retorna un oberservable formado por la conjunction de la url y el empleado

 addEmployee(employee:Employee):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}`, Employee);

  }

En add-employee.component.ts vamos a modificar el constructor y vamos a enviarle mediante los argumentos un EmployeeService y un Router

constructor(private employeeService: EmployeeService, private router: Router) { };

Luego creamos el metodo addEmployee que no recibe nada y lo que hace es subscribirse al obersvable que retorna el metodo anterior, luego tiene una function arrow por la que envía la info recibida (el empleado)

Por ultimo agregamos un error y un console log que muestra el error

 addEmployee() {

    this.employeeService.addEmployee(this.employee).subscribe(data => {

      console.log(data);

this.backToList();

    }, error => console.log(error));

  }

Creamos un método para que nos re-direccione a la list de empleados post add

backToList(){

    this.router.navigate(['/employees']);

  }

Agregamos el metodo al addEmployee

Ahora modificamos el onSubmit (lo que hará la app una vez que apretes el boton) para que llame al metodo add

 onSubmit() {

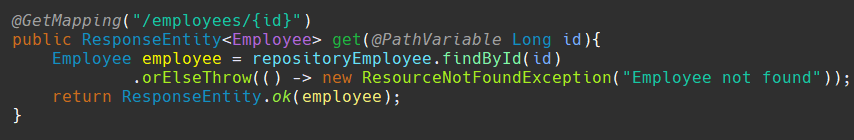
    this.addEmployee();

  }

Con esto, el método add ya debería agregar el employee nuevo a la DB

22- Ahora vamos a crear el método para buscar un empleado

En Eclipse en la carpeta de Controller creamos el siguiente metodo

Retorna una entidad que va a ser empleado, recibe por parametros el ID a buscar

Dentro del metodo utiliza el findby Id y en caso de que no lo recupere tira el error que habiamos personalizado