

# LOGIN CON ANGULAR SPRING BOOT Y MYSQL

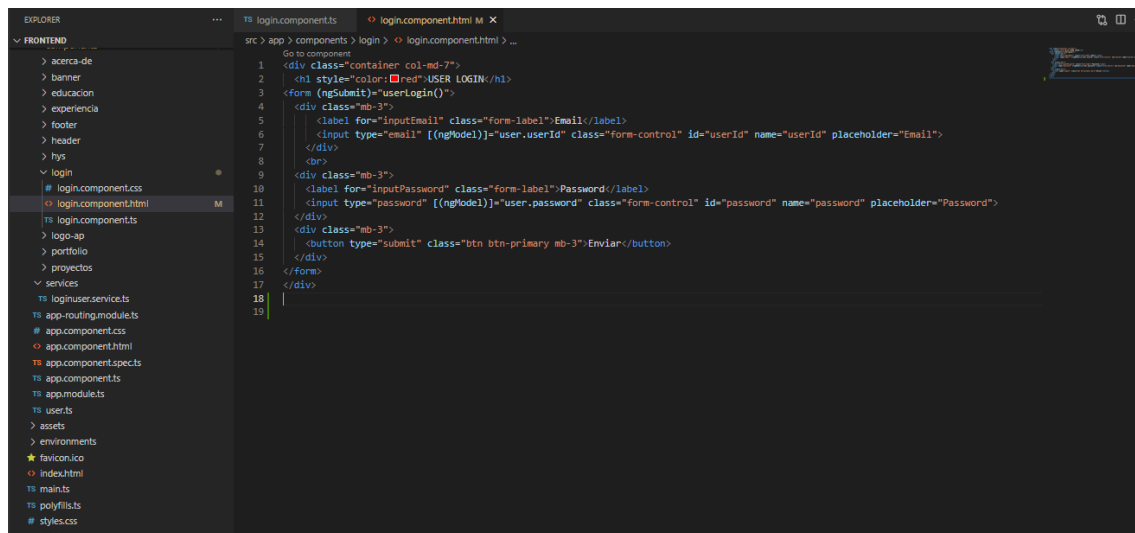
A continuación les voy a mostrar como hacer un logueo simple sin usar jwt. Usaremos todas las herramientas que nos brinda angular, spring boot y mysql. Vale aclarar que se da por sabido y entendido los módulos anteriores de angular básico y bases de datos. Si algún termino no se entiende les recomiendo volver un poco atrás a revisar los contenidos.

Ahora si arrancamos.

A esta altura del contenido asumimos que ya tenemos nuestro template del proyecto, casi terminado, con todas las secciones. Aparte deberemos contar con un botón de logueo en el head de nuestro proyecto.

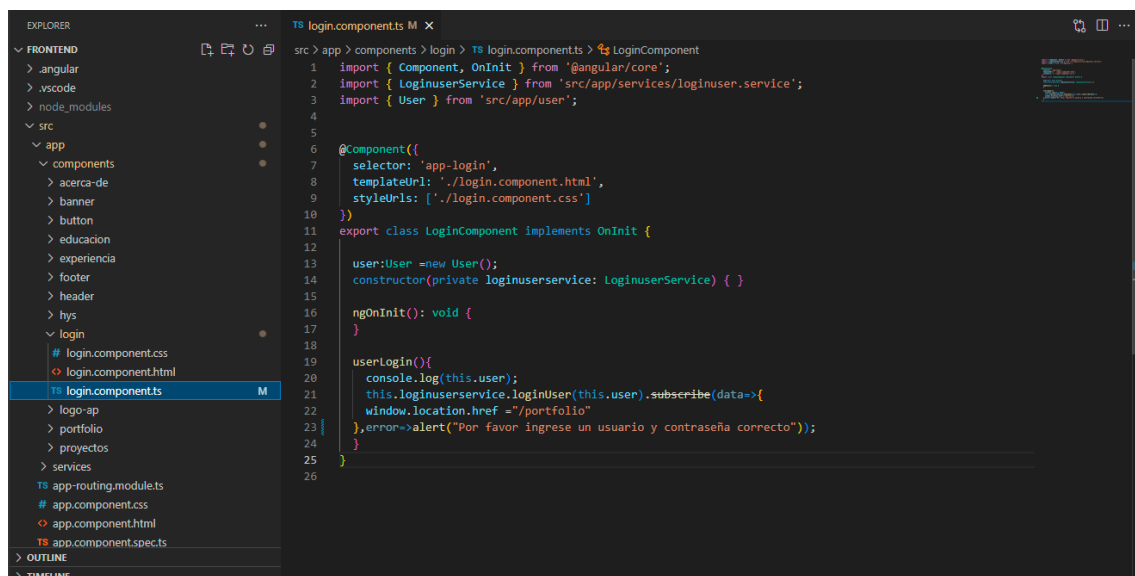
En primer lugar vamos a generar un componente para hacer un formulario simple de login

El cual apuntara con `[[ngmodel]]` a un usuario y a una contraseña, la cual mas adelante se cargara por base datos.



```
1 <div class="container" col-md-7">
2   <h1 style="color:red">USER LOGIN</h1>
3   <form (ngSubmit)="userLogin()">
4     <div class="mb-3">
5       <label for="inputEmail" class="form-label">Email</label>
6       <input type="email" [(ngModel)]="user.userId" class="form-control" id="userId" name="userId" placeholder="Email">
7     </div>
8     <br>
9     <div class="mb-3">
10      <label for="inputPassword" class="form-label">Password</label>
11      <input type="password" [(ngModel)]="user.password" class="form-control" id="password" name="password" placeholder="Password">
12    </div>
13    <div class="mb-3">
14      <button type="submit" class="btn btn-primary mb-3">Enviar</button>
15    </div>
16  </form>
17 </div>
18
19
```

Hecho esto, nos quedaría modificar el archivo .ts del login. El mismo se va a comunicar con el servicio, que a su vez hará la conexión con la API, pero eso lo vamos a ver luego.

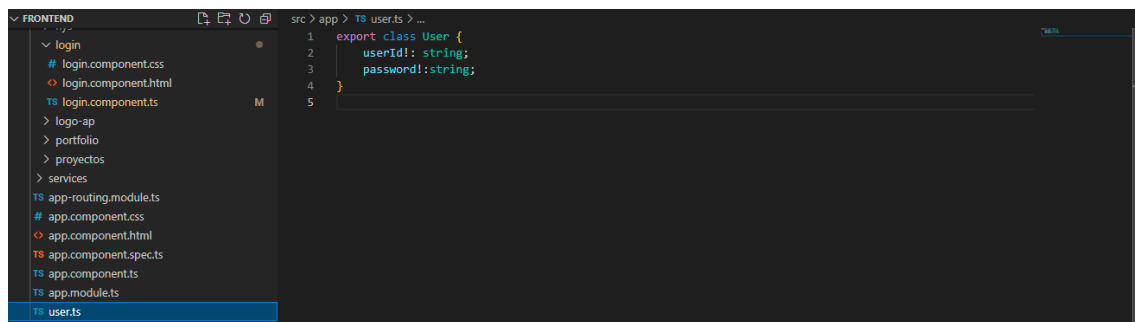


```
1 import { Component, OnInit } from '@angular/core';
2 import { LoginuserService } from 'src/app/services/loginuser.service';
3 import { User } from 'src/app/user';
4
5
6 @Component({
7   selector: 'app-login',
8   templateUrl: './login.component.html',
9   styleUrls: ['./login.component.css']
10 })
11 export class LoginComponent implements OnInit {
12
13   user: User = new User();
14   constructor(private loginuserService: LoginuserService) { }
15
16   ngOnInit(): void {
17   }
18
19   userLogin(){
20     console.log(this.user);
21     this.loginuserService.loginUser(this.user).subscribe(data=>{
22       window.location.href = "/portfolio"
23     },error=>alert("Por favor ingrese un usuario y contraseña correcto"));
24   }
25 }
26
```

Podemos ver varias cosas:

- Se inicializa un objeto user, el cual esta apuntando a una clase que vamos a crear después.
- En el constructor vemos a un servicio, se acuerdan que seria quien se va a comunicar con la API
- Tenemos una función userLogin(), que es la que habíamos llamado en el template del formulario de login. Esta función es la que va a tomar los datos que se le carguen en el template y los va a mandar al servicio para compararlos con los datos en la API. Si son correctos, nos redirige al template del portfolio, si son incorrectos nos muestra un mensaje de alerta que indica que se ingresen correctamente los datos.

Ahora si vamos a hacer la clase usuario (ng generate class user). Esta clase es la que nos va a indicar que tipo de dato requiere el user que habíamos usado anteriormente, que a su vez tiene que coincidir con la clase user de la API. El signo de admiración viene de la sintaxis propia de typescript que nos establece que los datos no sean nunca NULL

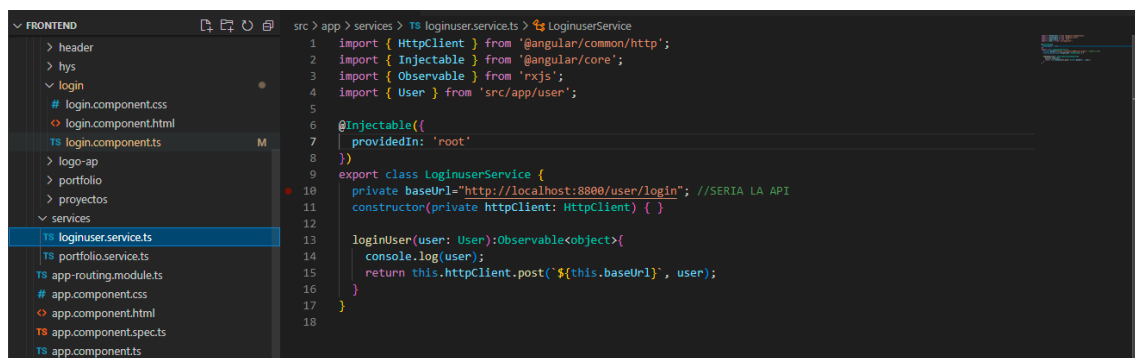


```
1 export class User {
2   userId: string;
3   password!: string;
4 }
5
```

No nos olvidemos de realizar las importaciones correspondientes, formmodule y el httpclientmodule.

Seguimos con el proceso...

Ahora vamos a crear el servicio, el cual se va a comunicar con la API (ng generate service loginuserservice)



```
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Observable } from 'rxjs';
4 import { User } from 'src/app/user';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class LoginUserService {
10   private baseUrl="http://localhost:8800/user/login"; //SERIA LA API
11   constructor(private httpClient: HttpClient) { }
12
13   loginUser(user: User):Observable<object>{
14     console.log(user);
15     return this.httpClient.post(`${this.baseUrl}`, user);
16   }
17 }
18
```

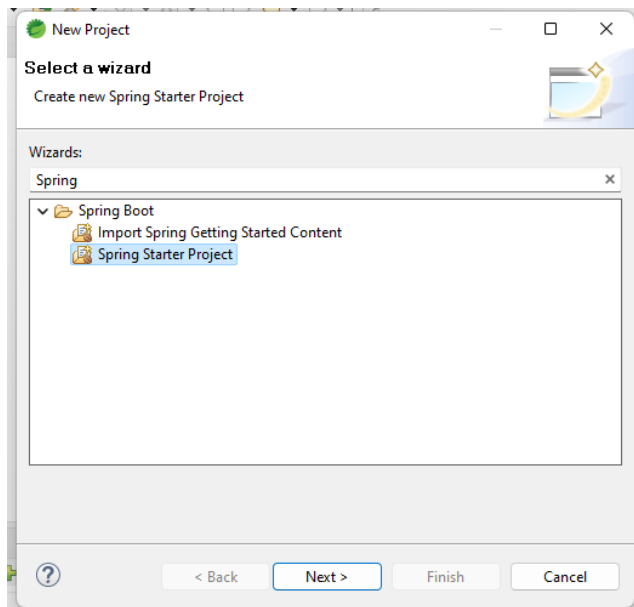
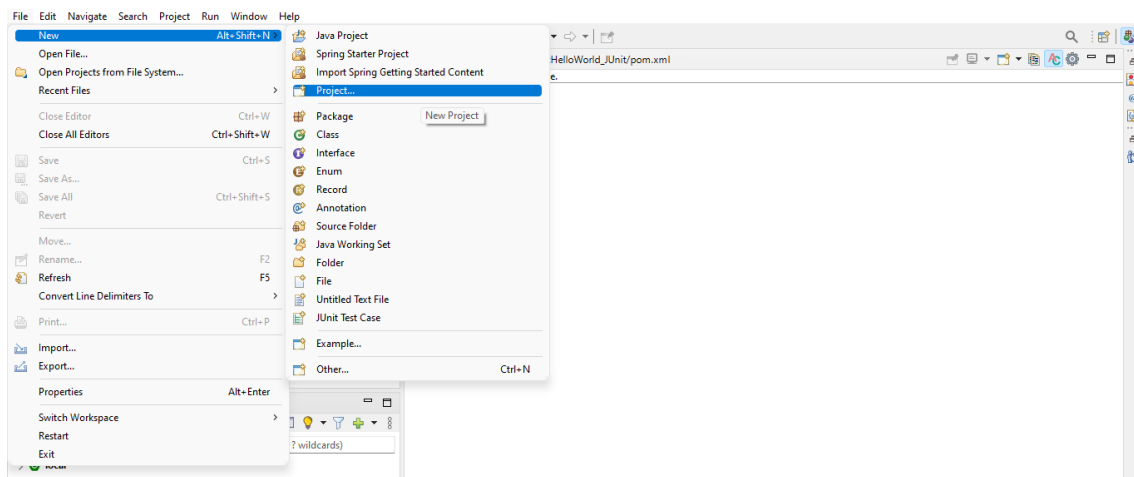
En el servicio vemos la conexión con nuestra API y un método llamado loginuser el cual capta los datos que se ingresaron en el formulario del template.

También se ve que importamos la clase user.

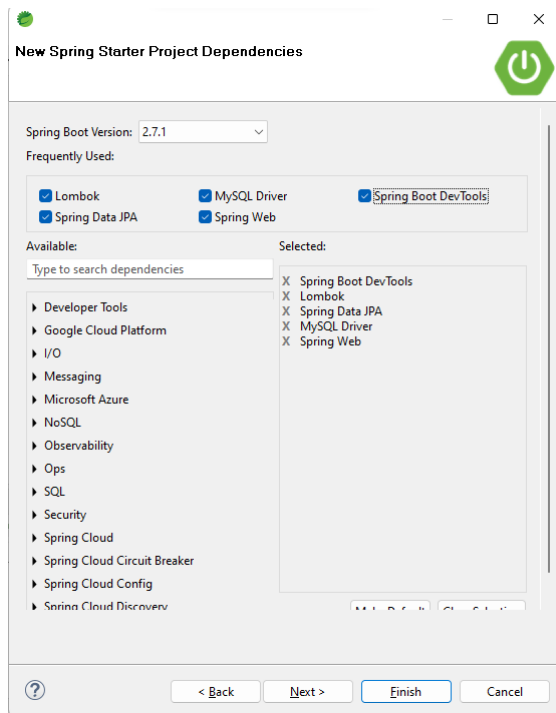
Del lado del frontend, eso seria todo. Obviamente al template de formulario lo pueden modificar y darle estilo a su gusto. También se le puede agregar las validaciones que vimos en el material de lectura.

Pasaremos a spring boot, en mi caso uso la aplicación spring tolos suite 4, ya que me resulto mas comoda, pero pueden utilizar la que guste (netbeans si es caso).

Creamos el proyecto.



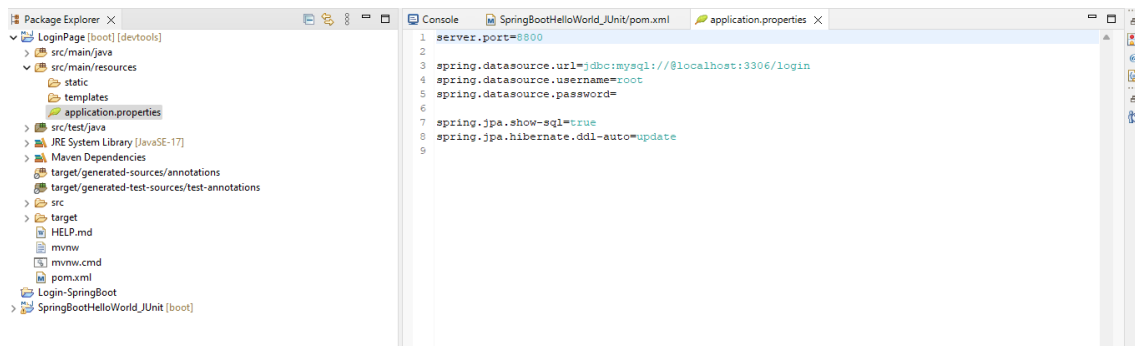
En wizard buscamos spring, y seleccionamos, Spring Starter Project



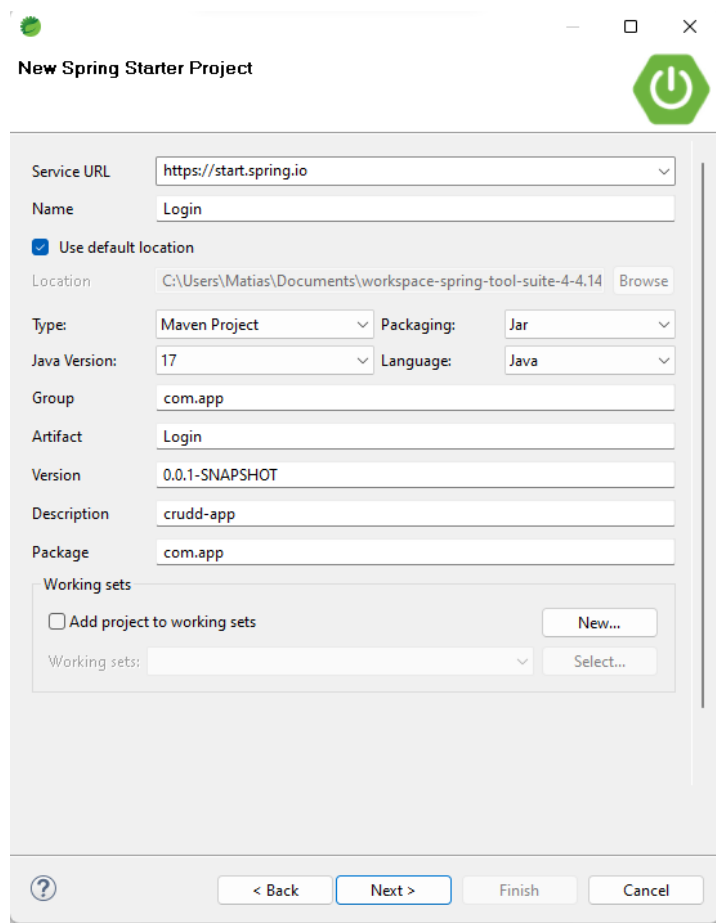
Luego de estos pasos, NEXT Y FINISH. Tomara un momento en la creación del proyecto.

Ahora vamos a poner mano al código de nuestro proyecto spring

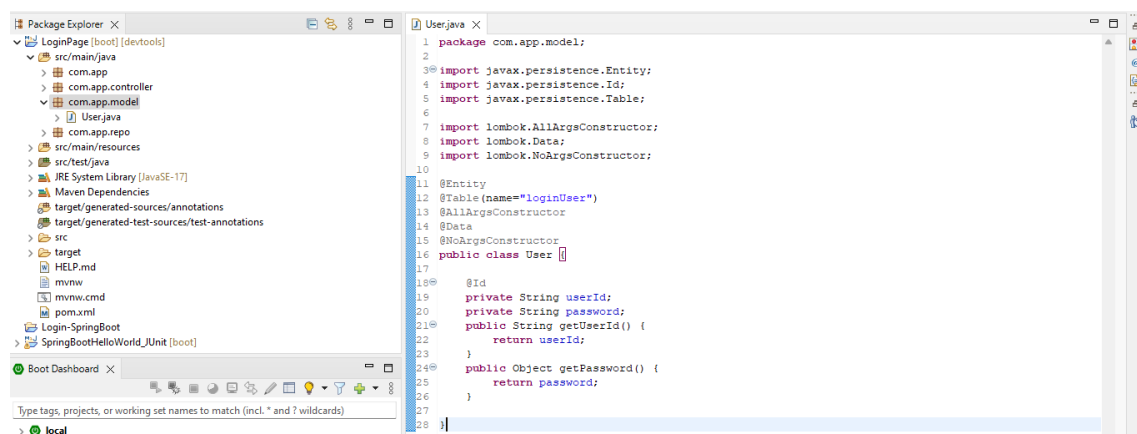
Lo primero que tenemos que hacer es configurar la conexión con la base de datos. En nuestro caso se llamara “login”



Lo siguiente que vamos a hacer es crear la clase user

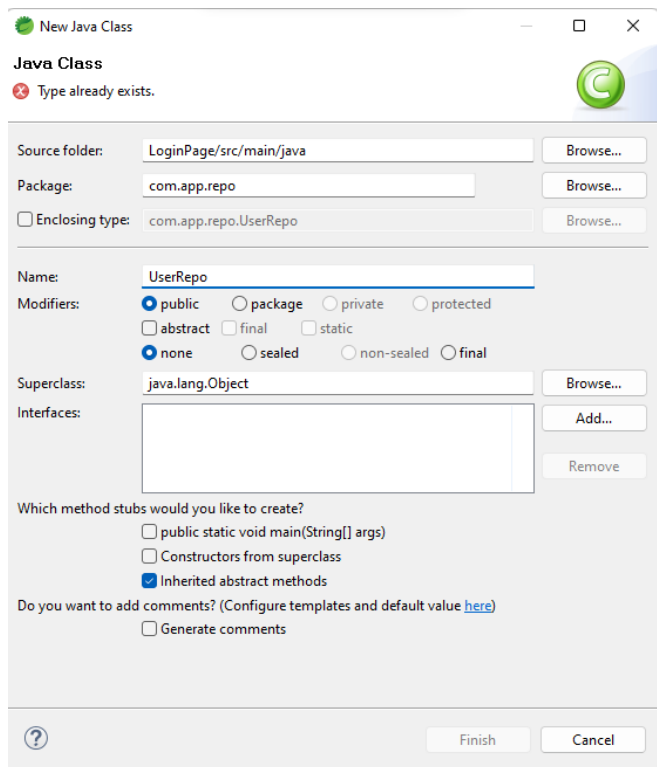


En esta clase vamos a volcar todo el contenido de la clase user que traíamos de nuestro front

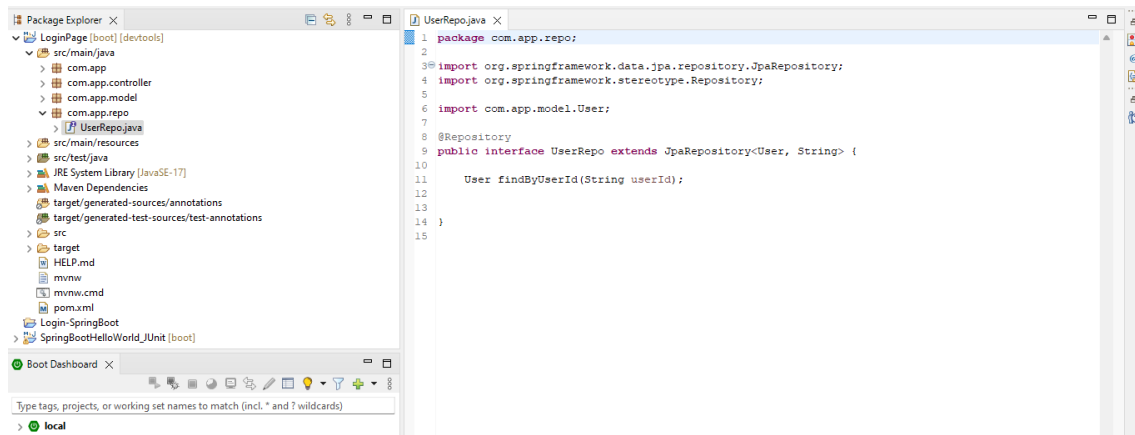


No hay mucho que explicar... Recuerden de hacer las importaciones para que no les de error el código.

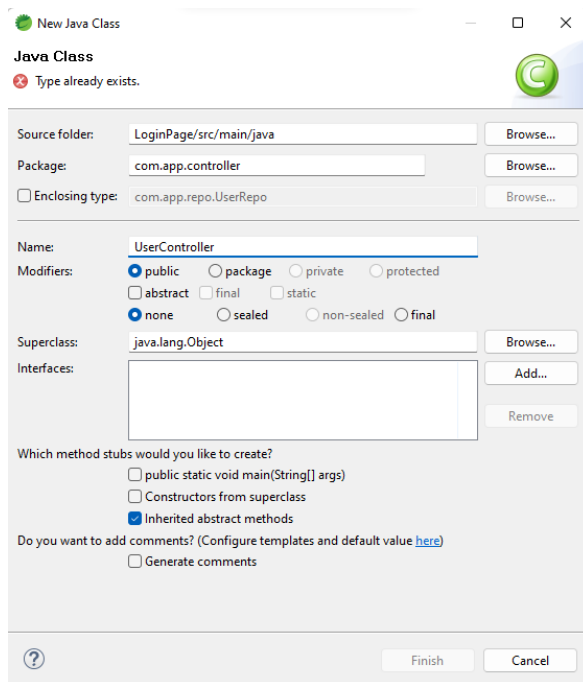
Seguidamente vamos a crear la interface del usuario



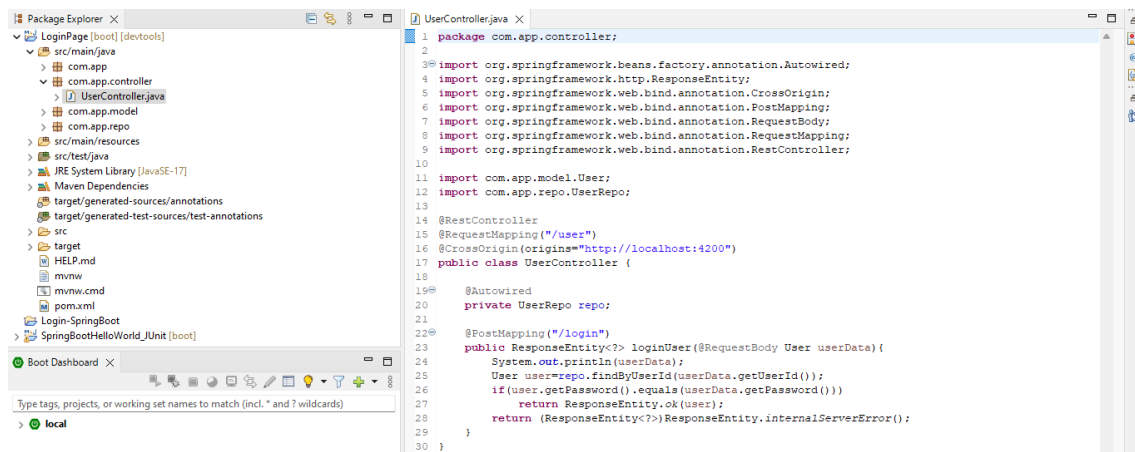
Y este seria el código, (tengan en cuenta que para que esta clase funcione como interface le tenemos que agregar luego de “Public” el termino interface)



Creamos el controlador que es el que se va a comunicar con el template.











Si se fijan ahí encontramos de nuevo el puerto 4200, que coincide con nuestro proyecto angular levantado (ng serve). No se olviden las importaciones...



The screenshot shows the SQL Server Enterprise Manager interface. The 'Navigator' pane on the left displays the 'login' database expanded, showing its 'Tables' (login\_user), 'Columns' (user\_id, password), 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', and 'Functions'. The 'user\_id' column is selected and highlighted in blue. The 'password' column is also visible. Other databases like 'portfolio', 'posts', 'spring\_jpa', and 'sys' are listed below. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The bottom bar shows 'Administration' and 'Schemas' tabs.

login\_user

Result Grid  Filter Rows:  | Edit:    | Export/Import:   | Wrap Cell Content: 

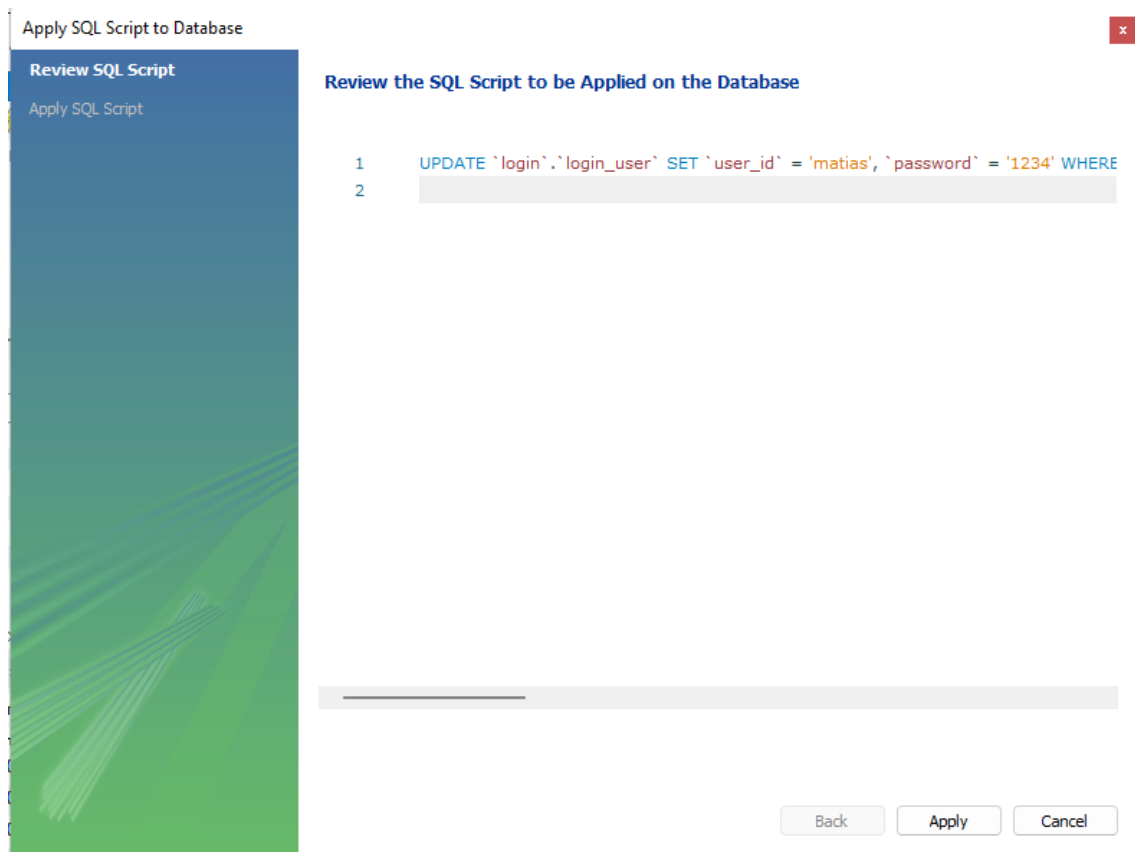
	user_id	password
	matas	1234
	NULL	NULL
	NULL	NULL
	NULL	NULL
	NULL	NULL
	NULL	NULL
	NULL	NULL
	NULL	NULL

login\_user 1 x

Apply Revert

Result Grid  
Form Editor





Y de nuevo apply. El usuario ya fue creado con su respectiva clase.

Espero no haberme olvidado de nada, recuerden que esto no es obligatorio, es una ayuda para que puedan acercarse mas al final del proyecto integrador. Si encuentran algún error, o hay algún problema con algo que haya explicado me lo hacen saber a través del foro o bien por mensaje privado.

Ojala les sirva.

Saludos!!