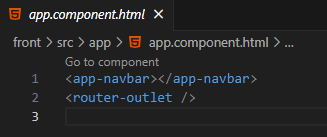
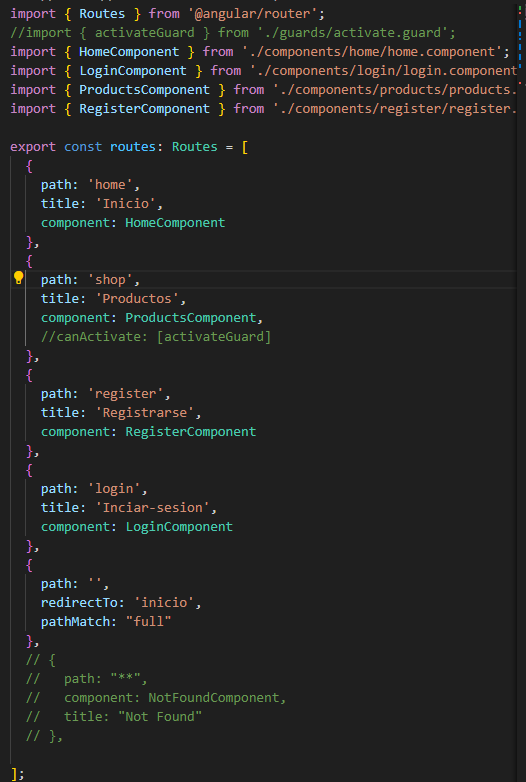
BACK Y FRONT UNION

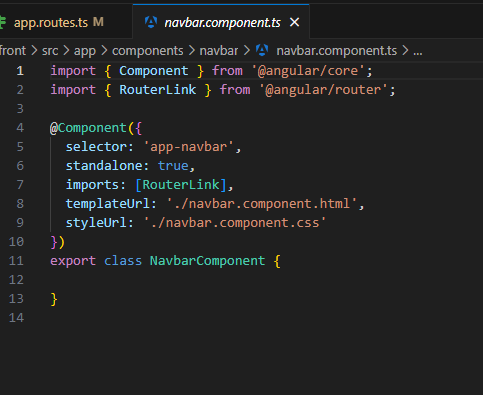
1. En mi app.component.html valido que este conectado el nav



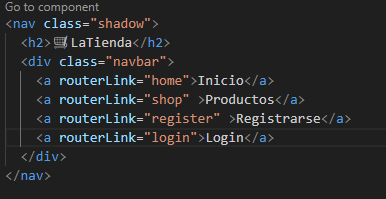
1. Validar que las rutas estén bien app.routes.ts



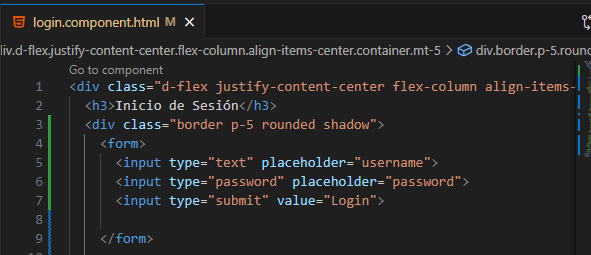
1. Habilitamos la opción router link en el navbar.component.ts



1. Verificar el cambio de href a router link en el navbar.component.html

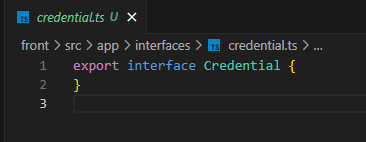


1. Luego seguimos con el login.component.html para iniciar el logueo del usuario y creamos un formulario de login

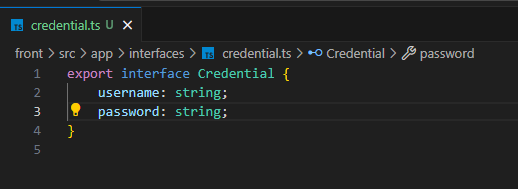


1. Luego voy a la lógica de mi componente en login.component.ts

* Para recibir los datos de usuario y la contraseña voy a crear una interface esto lo hago con el comando ng g interface interfaces/credential



* Dentro de la estructura de la interface voy a crear un objeto que se va a encargar de recibir el usuario y la contraseña de esta forma



1. Luego lo importo en mi login.component.ts esta interface

import { Credential } from '../../interfaces/credential';

* para usarla más adelante el siguiente paso es capturar los datos que se ingresan en el front para eso vamos a importar los formularios reactivos

import { ReactiveFormsModule, FormControl,FormGroup,Validators } from '@angular/forms';

* y los voy a activar en el import solo el ReactiveFormsModule



1. Ahora voy a dedicarme a capturar esos datos dentro de mi login.component.ts

Voy a crear un objeto llamado credentialsForm = que va a tener unas nuevas instancias del FormGroup es objeto se va a encargar de capturar la información que se ingresa en el formulario que esta en el login.componet.html

credentialsForm = new FormGroup({});

1. De aqui necesito un username que va a tener un nuevo formulario de control lo primero que va a traer es un string vacio en donde se van a enviar los datos que se digiten en el formulario y estos datos se les va a realizar una validación requerida para eso usamos Validators. Required

credentialsForm = new FormGroup({

    username: new FormControl('', Validators.required),

    password: new FormControl('', Validators.required),

  });

* Por ahora si probamos no está capturando ninguna información así que voy al login.component.html para ligar el login.component.ts con el login.component.html

1. En el html voy a utilizar una directiva para el formulario

<form [formGroup]="credentialsForm">

* En el input tengo que asociar el formcontrol con la propiedad del username con el atributo FormControlName realizo la asociación quedaría de esta forma

<form [formGroup]="credentialsForm">

      <input type="text" placeholder="username" formControlName="username">

    <input type="password" placeholder="password" formControlName="password">

      <input type="submit" value="Login">

    </form>

1. Ahora hace falta que el boton de submit almacene la información con un evento que debo poner en el formulario este evento es el (ngSubmit) dentro del cual voy a crear un método para manejar y enviar la informacion al cual voy a llamar handleSubmit() que significa manejarEnviar este me va a presentar error por que no existe en mi lógica o también llamado mi login.component.ts

<form [formGroup]="credentialsForm" (ngSubmit)="handleSubmit()">

      <input type="text" placeholder="username" formControlName="username">

      <input type="password" placeholder="password" formControlName="password">

      <input type="submit" value="Login">

    </form>

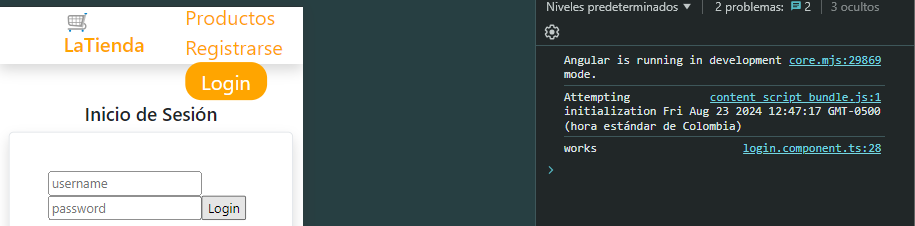
1. Voy a crear el método en mi login.component.ts con un console.log que diga Works para hacer una prueba

handleSubmit() {

    console.log("works");

};

1. Y doy clic en el botón para verificar que este escuchando ese clic por el momento reviso la consola de mi navegador



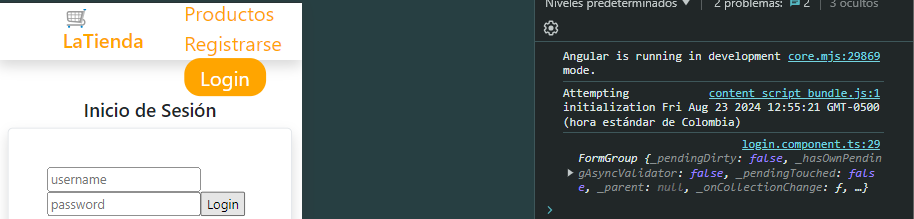
1. Ahora yo quiero saber si estas credenciales son validas para permitir el ingreso al usuario

* Primero muestro por consola que está en credentialForms

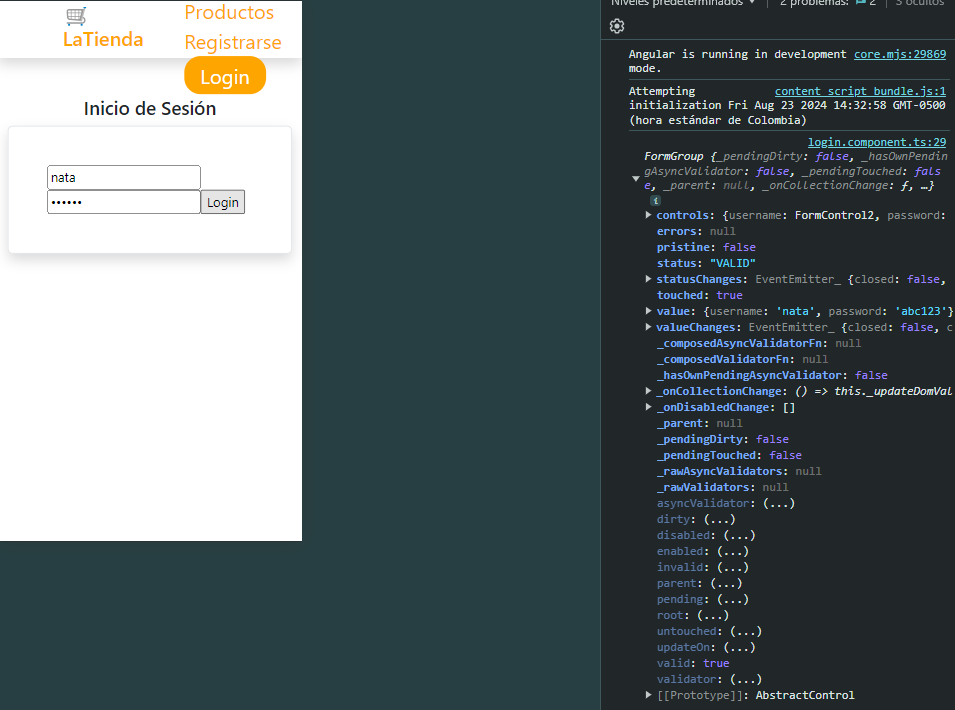
handleSubmit() {

console.log(this.credentialsForm);

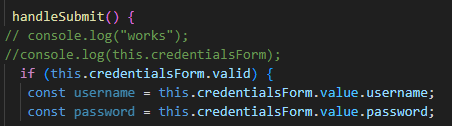
};



* Ahora se realiza la prueba con datos y ahora vemos que recibe los datos y el validator esta en true



1. Listo ya que sabemos que recibimos datos entonces vamos a armar un objeto para que cumpla con la estructura, asi que voy a crear una condicional que tenga dos const username donde almaceno lo que viene de this.credentialsForm y recibo el valor .value de username que es viene de la interface credentials y lo mismo con el password



* Como ya no me recibe nada por la condicional de los datos del credential voy a crear una constante que le voy a llamar credential y con los dos puntos le digo que tipo de dato es string,number, etc en este caso el dato va a ser de tipo Credential para que el sepa que su tipo va a ser de la interface en donde están almacenados los objetos que son mi user y mi password que ya están identificados previamente como tipo string, a esto le voy a asignar unos valores una propiedad que se llame username y password quedaría de la siguiente forma

handleSubmit() {

  // console.log("works");

  //console.log(this.credentialsForm);

    if (this.credentialsForm.valid) {

     const username = this.credentialsForm.value.username;

     const password = this.credentialsForm.value.password;

const credential: Credential = {

          username,

          password,

  };

} else {

  console.log('Error invalid form');

       }

   }

* Como esto me genera error que dice que el tipo string nulo o indefinido no es asignable a un tipo string entonces lo que pasa aquíi es que yo le estoy diciendo que este objeto debe tener una estructura si o si es un string el erro me dice que debo asegurarme que hay un string entonces de que manera me aseguro yo de eso
* Vamos a crear una condición para asegurar que el valor es de tipo string en donde diga si el tipo de dato que estoy usando es un string pero esto lo debo hacer tanto para username como para el password

if (typeof username === 'string' && typeof password === 'string') {

    const credential: Credential = {

          username,

          password,

  };

* Y creo un console log al objeto credential

if (typeof username === 'string' && typeof password === 'string') {

    const credential: Credential = {

          username,

          password,

  };

  console.log(credential);

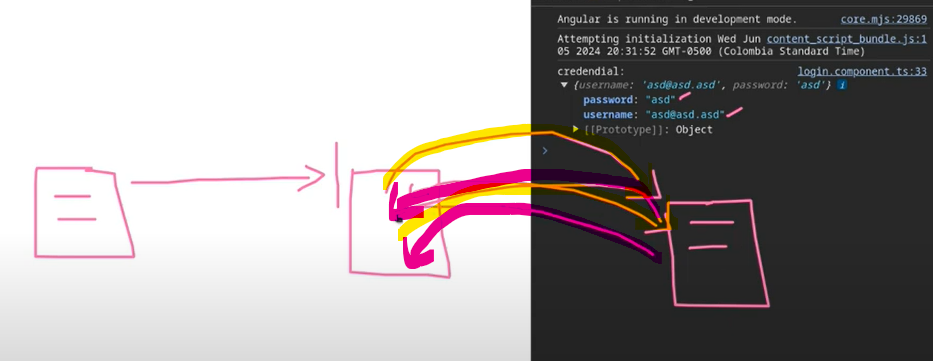
* para ver como se comporta si envio vacio me va a mostrar un error que es invalido mi formulario
* Si envio datos y los envio podre ver nuevamente los datos pero dentro de credential



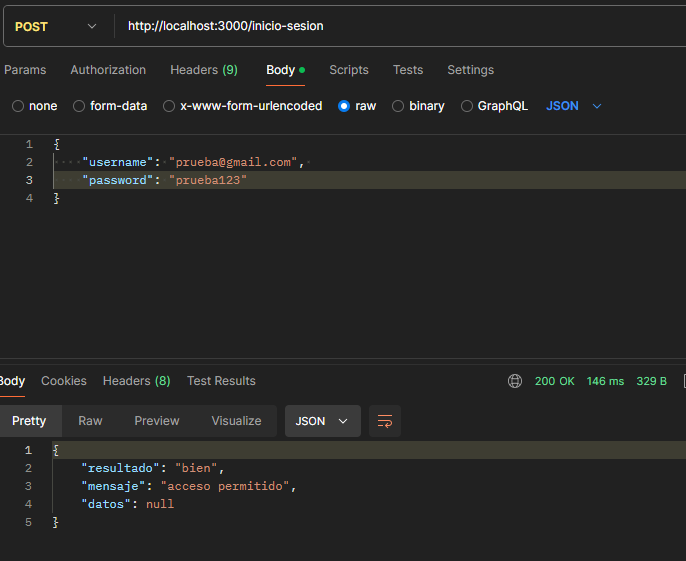
1. En este punto ya tengo los datos capturados pero que puedo hacer con estos datos

* Esperar respuestas
* Hacer la prueba donde ingreso datos por el postman y no se logue la persona asi los datos esten correctos por que aun faltan pasos para que se pueda validar esta información del back

1. Que es un API es como una puerta de ingreso para envia datos entonces en este caso debo crear un api que se va a encargar de loguear un usuario ya que los datos que tenemos los debemos usar para enviarlos a un backend entonces esto debe funcionar con un servicio que vaya y valide si los datos ingresados están en la base de datos de usuarios y luego responda si están bien los datos realice una accion que seria permitirle el ingreso

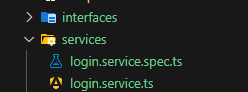


1. Vamos a revisar nuestra base de datos a la cual ya se le había creado una ruta para el incio de sesión
2. servidor.use('/inicio-sesion',enrutadorInicioSesion);
3. explico el controlador de inicio de sesión y muestro en potman su funcionamiento



1. luego de explicar lo del controlador de inicio de sesion y mostrar en postman que funciona vamos al front

* dentro del front en el login.component.ts realizamos lo siguiente
* como ya el método de hadleSubmit esta funcionando en parte por que ya recibe los datos pero aun no se envían al back aun no se pueden validar esos datos entonces vamos a crear un servicio con el siguiente comando ng g service services/login
* se crea este servicio debajo de la carpeta de interfaces



Ejecuto el comando para validar que mi consola del front corra sin errores con ng serve

1. dentro de login.service.ts primero voy a importar la dependencias que voy a necesitar

* inject que viene también de angular core
* un HttpClient que viene de angular common http pero para que este archivo no me genere errores debo ir al archivo de app.config.ts e importar algo que se conoce como el provideHttpClient que viene de angular common http y tambien lo debo agregar a mis providers el código n el app.config.ts quedaría de la siguiente forma

import { ApplicationConfig } from '@angular/core';

import { provideRouter } from '@angular/router';

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

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {

providers: [provideRouter(routes), provideHttpClient()]

};

* esto se hace para que al usar el HttpClient en mi archivo login.service.ts no se me generen errores entonces en mi archivo login.service.ts

import { Injectable, inject } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class LoginService {

  constructor() { }

  httpClient = inject(HttpClient);

}

* esta librería de http client lo que hace es enviar por debajo una petición es como si en postman yo al ingresar los datos hiciera clic en send el por debajo valida los datos de la solicitud y nos da una respuesta
* entonces ahora voy a crear un método que va a realizar una operación que funcione como clic en la función y lo que va a hacer es ir al método post y recibir la ruta que tenemos en postman del inicio de sesión y no va a devolver un payload el payload es el mismo objeto que tenemos en postman para el envio de datos a validar

login(){

    return this.httpClient.post('http://localhost:3000/inicio-sesion', payload );

  }

* para esto debo importar la interface que cree anes para las credenciales entonces la importo dentro de mi login.service.ts

import { Credential} from '../interfaces/credential';

* entonces como yo ya tengo los datos en mi front donde los recibo y también tengo mi parte del back donde mi controlador inicio de sesión ya esta validando los datos entonces las credenciales van a ser mi objeto dentro del método asi que dentro de mi método login voy a decirle que se llama credential y que es de tipo credential que en este caso valida el tipo de la interface el código en este punto quedaría asi

login(credential: Credential){

    return this.httpClient.post('http://localhost:3000/inicio-sesion', credential );

  }

* ya el método quedaría listo para poderlo utilizar, entonces quien va a llamar ese método en este caso quien lo va a usar es mi login.component.ts
* asi que lo importo

import { LoginService } from '../../services/login.service';

* y dentro de mi clase injecto la información del login service asi

export class LoginComponent {

loginService: LoginService = inject(LoginService);

* para poder usar este servicio que se creo dentro de mi componente de login dentro del método en donde recibíamos los datos por la consola asi que modificamos el método de la siguiente manera

this.loginService.login(credential).subscribe((response: any) => {

console.log(“response: ”, response);

* realizamos la prueba enviando los datos desde el front para visualizar la respuesta de la consola



1. Luego de que las credenciales esten bien voy importar el enrutador para enviarlo a otro lugar en este caso cuando se logue quiero que vaya al shop

import { Router } from '@angular/router';

* Dentro de la clase lo inyecto

export class LoginComponent {

  router = inject(Router);

* Entonces ya despues de importar ruotes le digo que en ese router navege hasta el shop

this.loginService.login(credential).subscribe((response: any) => {

          console.log("response:", response);

this.router.navigateByUrl('/shop');

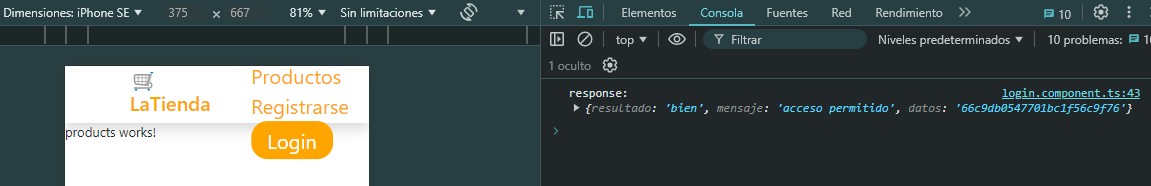
});

}

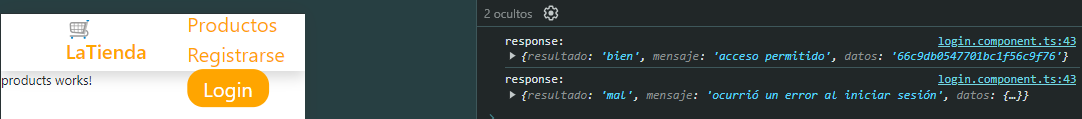
} else {

console.log('Error invalid form');

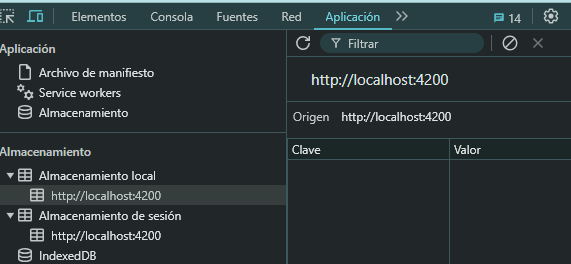
* Si realizamos la prueba ingresando los datos me va a dirigir a el shop



* Pero si probamos con datos incorrectos prácticamente nos va a permitir ingresar por que la condición es que hallan datos dicho esto vamos a realizar otras cosas mas



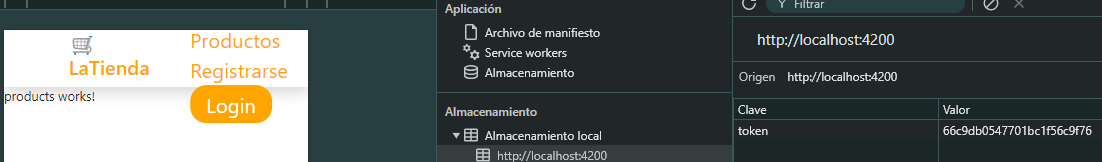
* Nosotros tenemos una zona de almacenamiento en nuestro navegador entonces vamos a aplication



* En este caso vamos a usar nuestro local storage para guardar datos en este lugar nosotros debemos guardar los datos como llave y valor en formato json entonces antes de redireccionar al usuario vamos a realizar lo siguiente dentro de mi login.component.ts

localStorage.setItem('token', response.datos);

* en este caso puedo poner el nombre que yo quiera le voy a poner token a mi clave y le voy a indicar que token es igual a response.datos realizamos la prueba nuevamente y vemos como se almacena la información



* entonces antes del localstorage vamos a montar una condición para que valide el resultado y dependiendo de la respuesta le permita ingresar o no al shop

if (response.resultado === 'bien') {

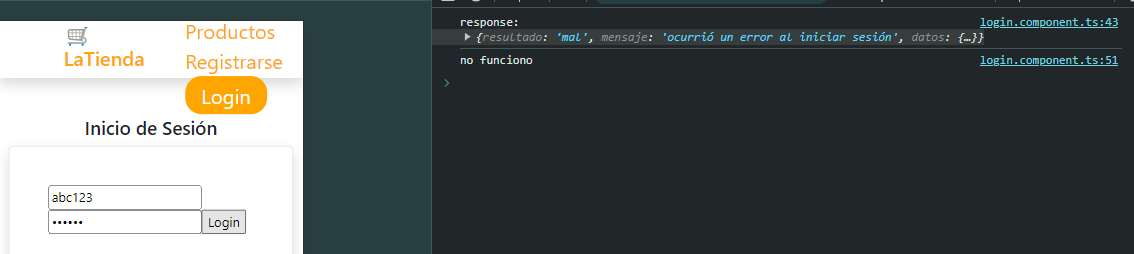
             localStorage.setItem('token', response.datos);

          this.router.navigateByUrl('/shop');

          } else {

            console.log('no funciono');

* y probamos nuevamente



* Ahora con los datos reales

Y vemos como nos redirige al shop