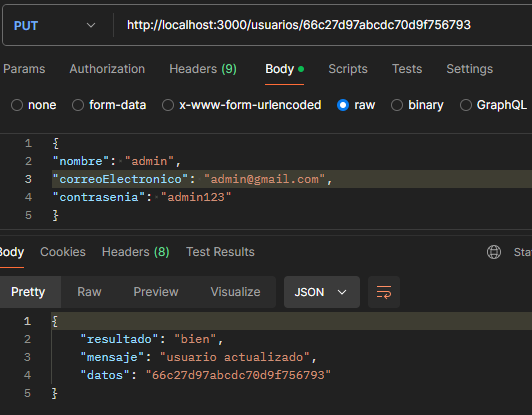
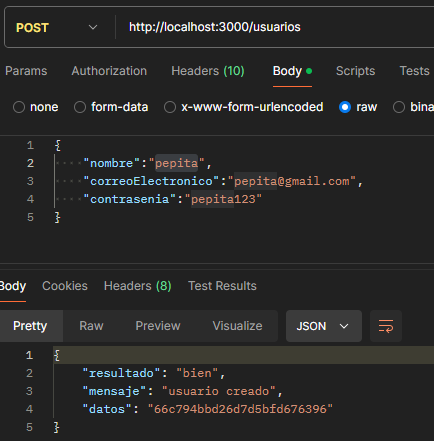
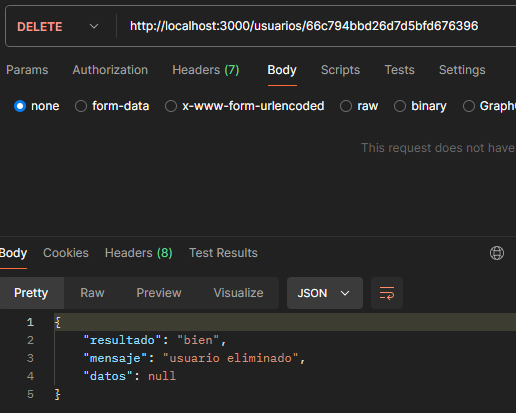
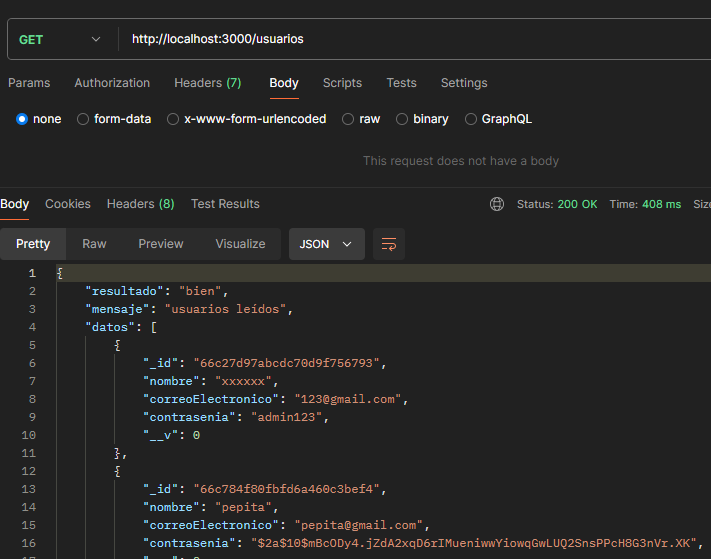


Loguearse y registrarse con Node Js, express, mongoDb, Angular

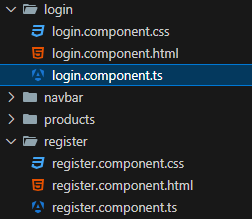
1. Realizar pruebas en postman al CRUD para verificar que si esta funcionando
2. Create c. Update



1. Read d.Delete

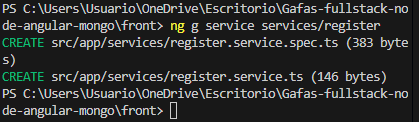


1. Cuando ya tenemos los componentes creados para el login y el register realizamos lo siguiente.

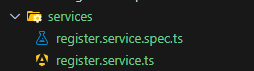


1. Creamos un servicio para el registro de usuarios

Con el comando ng g service services/register



Y creara una carpeta de la siguiente forma

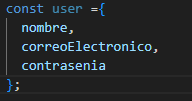


Vamos a nuestro archivo register.service.ts y realizamos la siguiente configuración

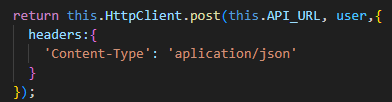
* Importamos inject de angular core // para injectar o agregar datos
* Importamos HttpClient de angular common http // Imagina que tu aplicación Angular es una casa y que necesita comunicarse con el mundo exterior para obtener información o realizar acciones. El HttpClient es como una ventana que permite a tu aplicación "asomar la cabeza" y hacer peticiones a un servidor remoto.
* creamos un Metodo llamado createUser(aquí dentro ingreso los datos del modelo de usuarios con su tipo)



* creamos una constante para los usuarios que trae los datos del schema



* Luego enviamos la solicitud al método POST del servidor para crear un nuevo usuario



Desglosando la función:

this.HttpClient.post:

Llama al método post del objeto HttpClient, que es el encargado de realizar peticiones HTTP de tipo POST.

this.API\_URL:

Especifica la URL del endpoint del servidor al que se realizará la petición. Este endpoint es donde se encuentra la lógica para crear nuevos usuarios.

user:

Representa los datos del nuevo usuario que se desea crear. Estos datos suelen ser un objeto JSON que contiene la información necesaria (nombre, email, contraseña, etc.).

{ headers: { 'Content-Type': 'application/json' } }

Define los encabezados de la petición HTTP:

Content-Type: application/json: Indica al servidor que los datos que se están enviando en el cuerpo de la petición están en formato JSON. Esto es fundamental para que el servidor pueda interpretar correctamente los datos del nuevo usuario.

AQUÍ SE FINALIZA LO QUE SE HACE CON EL SERVICIO DE REGISTER

1. Vamos a la terminal e instalamos lo siguiente   
   npm install cors express jsonwebtoken ngx-localstorage ngx-toastr rxjs tslib zone.js

 **cors:** Permite habilitar el acceso a recursos de otros orígenes en tu servidor Node.js, lo cual es esencial para las aplicaciones Angular que se comunican con un backend diferente.

 **express:** Un framework minimalista y flexible para Node.js que se utiliza comúnmente para construir API backend.

 **jsonwebtoken:** Una biblioteca para trabajar con tokens JSON Web Tokens, que se utilizan comúnmente para autenticación y autorización.

 **ngx-localstorage:** Un módulo Angular para trabajar con el almacenamiento local del navegador.

 **ngx-toastr:** Un módulo Angular para mostrar notificaciones al usuario.

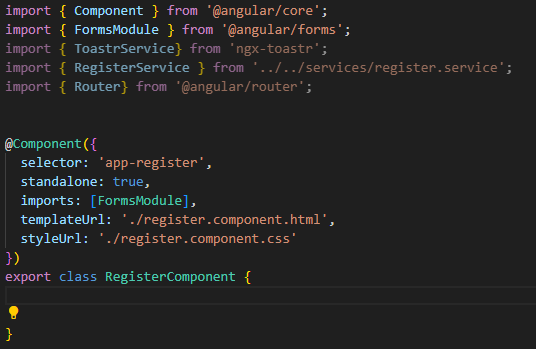
 **rxjs:** Una biblioteca para la programación reactiva en JavaScript, utilizada ampliamente en Angular.

 **tslib:** Un helper para TypeScript que proporciona utilidades para la transpilación.

 **zone.js:** Un polyfill para la zona de ejecución de Angular, esencial para el cambio de detección.

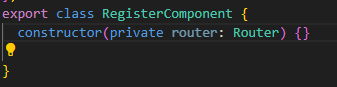
1. Vamos a nuestro componente register.component.ts

* Importamos



Luego en lo que vamos a exportar

* Creo un constructor(private router: Router){}



**constructor:** Es un método especial en TypeScript que se ejecuta cuando se crea una instancia de una clase (en este caso, tu componente).

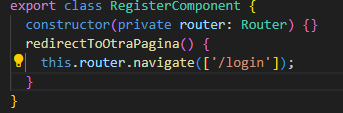
private: Indica que la propiedad router solo será accesible dentro de la clase del componente.

router: Es una instancia del servicio Router de Angular, que es responsable de gestionar la navegación entre diferentes vistas de tu aplicación.

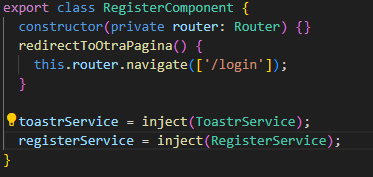
**¿Por qué es útil?**

**Redireccionamientos condicionales:** Puedes redirigir al usuario a una página de inicio de sesión, una página de confirmación o una página de error en función del resultado del proceso de registro.

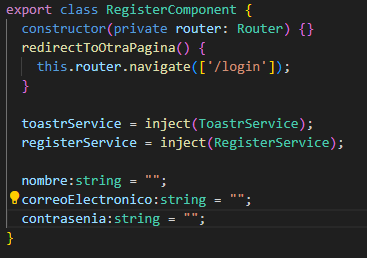
* Creo un método para redirigir a otra pagina después del registro en este caso vamos a dirigir al usuario al login para que se loguee después de registrarse



* Luego inyecto el servicio de toastr y el de register



* Luego monto los atributos del formulario que se van a enlazar para tener acceso a las propiedades



* Luego creo un método que va a manejar y enviar los formularios
* **¿Qué hace este método?**
* Este método, comúnmente llamado handleSubmit en componentes de Angular que manejan formularios, se encarga de procesar los datos ingresados por el usuario en un formulario y enviarlos a un servicio para crear un nuevo usuario.

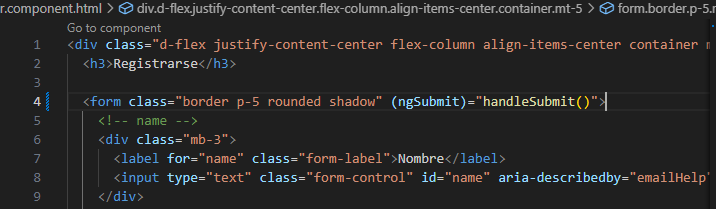
En resumen, este método handleSubmit realiza las siguientes acciones:

* **Valida** que el usuario haya ingresado un nombre.
* **Invoca** un servicio para crear un nuevo usuario con los datos del formulario.
* **Maneja** la respuesta del servidor.
* **Redirige** al usuario a otra página, independientemente del resultado de la creación del usuario.

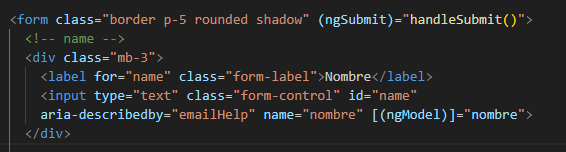


1. Vamos a nuestro componente register.component.html

Aquí como ya tenemos una estructura básica de un formulario para recibir los datos para el registro entonces vamos a realizar unas modificaciones



* En el input se agregará el nombre como esta en el shema y el ngModel llamando a la propiedad del shema



* Quedaría de esta forma

