Glosario parte del Frontend

1. **implements**: Palabra clave en TypeScript utilizada para declarar que una clase implementa una interfaz específica.
2. **OnInit**: Interfaz de Angular que define un método **ngOnInit()**, el cual se llama después de que Angular haya inicializado todas las propiedades del componente.
3. **selecto**: Posiblemente se refiera a **selector**, que es una propiedad de los componentes en Angular que define el nombre de la etiqueta HTML utilizada para instanciar el componente en la plantilla.
4. **templateUrl**: Propiedad de los componentes en Angular que especifica la ruta del archivo HTML asociado al componente.
5. **styleUrls**: Propiedad de los componentes en Angular que especifica la ruta o rutas de los archivos CSS asociados al componente.
6. **export**: Palabra clave en TypeScript utilizada para hacer que clases, funciones, variables o módulos estén disponibles para su uso fuera del archivo actual.
7. **class**: Palabra clave en TypeScript utilizada para definir una clase.
8. **ngOnInit**: Método en Angular que se llama automáticamente después de que Angular ha inicializado todas las propiedades del componente.
9. **subscribe**: Método utilizado para suscribirse a un flujo de datos en Angular, generalmente a un observable.
10. **ngFor**: Directiva estructural en Angular utilizada para iterar sobre una lista y renderizar un elemento para cada elemento de la lista.
11. **routerLink**: Directiva en Angular utilizada para establecer la navegación entre vistas en una aplicación de una sola página.
12. **ngIf**: Directiva estructural en Angular utilizada para mostrar o ocultar elementos del DOM basados en una expresión booleana.
13. **length**: Propiedad de JavaScript que devuelve la longitud de una cadena o matriz.
14. **get**: Método en TypeScript utilizado para obtener el valor de una propiedad de un objeto.
15. **put**: Método HTTP utilizado para actualizar recursos en un servidor.
16. **delete**: Método HTTP utilizado para eliminar recursos en un servidor.
17. **post**: Método HTTP utilizado para enviar datos al servidor para crear un nuevo recurso.
18. **FormBuilder**: Clase en Angular utilizada para construir instancias de FormGroup.
19. **FormGroup**: Clase en Angular que representa un grupo de controles en un formulario.
20. **Validators**: Clase en Angular que proporciona funciones de validación para ser utilizadas en formularios.
21. **ActivateRouter**: Error tipográfico. Posiblemente quisiste decir **ActivatedRoute**, que es un servicio en Angular que proporciona información sobre la ruta activa.
22. **Router**: Servicio en Angular utilizado para la navegación entre las vistas de una aplicación de una sola página.
23. **component**: Palabra clave en Angular utilizada para definir un componente.

Glosario del Servidor

1. **await**: Palabra clave en JavaScript utilizada para esperar a que una promesa se resuelva antes de continuar con la ejecución del código.
2. **find**: Método en Mongoose (y en muchas bases de datos) utilizado para buscar documentos que coincidan con ciertos criterios en una colección.
3. **json**: Método en Express utilizado para enviar una respuesta en formato JSON.
4. **req**: Objeto en Express que representa la solicitud HTTP entrante.
5. **res**: Objeto en Express que representa la respuesta HTTP que se enviará al cliente.
6. **body**: Propiedad de la solicitud en Express que contiene los datos enviados por el cliente, generalmente utilizada en las solicitudes POST.
7. **save**: Método en Mongoose utilizado para guardar un documento en la base de datos.
8. **catch**: Palabra clave en JavaScript utilizada para manejar errores en promesas.
9. **status**: Método en Express utilizado para establecer el código de estado HTTP de una respuesta.
10. **send**: Método en Express utilizado para enviar una respuesta al cliente.
11. **async**: Palabra clave en JavaScript utilizada para definir funciones asíncronas.
12. **mongoose**: Biblioteca de modelado de datos para MongoDB en Node.js.
13. **require**: Palabra clave en Node.js utilizada para importar módulos.
14. **module**: Objeto en Node.js que representa el módulo actual.
15. **controller**: Componente en el patrón de diseño MVC (Modelo-Vista-Controlador) que se encarga de manejar las solicitudes del cliente y de interactuar con el modelo y la vista.
16. **routers**: Componente que define las rutas de la aplicación en Express, que agrupa controladores y define cómo se manejan las solicitudes en cada ruta.
17. **api**: Interfaz de programación de aplicaciones, generalmente utilizada para definir cómo se accede y se manipula la información en un servidor.
18. **listen**: Método en Express utilizado para iniciar un servidor y hacer que escuche en un puerto específico.
19. **cors**: Módulo de Node.js utilizado para habilitar el intercambio de recursos entre diferentes dominios en Express.
20. **express**: Framework de Node.js utilizado para crear aplicaciones web y APIs de manera más sencilla y organizada.
21. **schema**: Estructura de datos en Mongoose que define la forma de los documentos en una colección de MongoDB.

# Bibliografía

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

https://stackoverflow.com/

https://angular.io/docs

https://nodejs.org/api/

https://www.mongodb.com/docs/