Práctica Final Programación Web II

Enunciado

Desarrollar, utilizando cualquiera de los Frameworks vistos en clase un sistema de autenticación mediante token JWT.

4 puntos-En un fichero solucion.pdf en la raíz del código, debe explicarse por qué se ha elegido el framework correspondiente, la estructura de la aplicación y un dibujo con el esquema tecnológico.

2 puntos-Formulario de registro. Validar si las contraseñas coinciden en el registro

2 puntos-Formulario de login

2 puntos-Finalizar sesión

Tecnologías usadas

Back-End

NodeJS

Node.js es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. La ejecución en tiempo real está pensada para usarse fuera del contexto de un explorador web (es decir, ejecutarse directamente en una computadora o sistema operativo de servidor). Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros.

En primer lugar, se ha escogido Node por la posibilidad de ejecutarse en varios servidores, yo trabajo en Windows, pero este sistema podrá ejecutarse en Linux. Otro de los aspectos que me ha llevado a trabajar con Node es su facilidad de uso y la experiencia que tengo con esta tecnología. Por último, reduce los tiempos de trabajo ya que permite escribir JavaScript tanto del lado del servidor como del cliente haciendo así que la transferencia de la información sea más rápida.

Express

Express es el framework web más popular de *Node*, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas). Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas. Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta. Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

Se ha escogido Express por los conocimientos previos que ya se tenían sobre esta tecnología y por lo tanto reducirían el tiempo de desarrollo. Además, es fácil de usar ya que es lenguaje JavaScript y se integra bien con Node JS.

MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

Se ha escogido MySQL por la familiaridad que ya se tenía con el sistema, así como ser una de las más extendidas que se incorpora en el XAMPP, por lo que reduce la posibilidad de tener que instalar software extra en el terminal.

Otras tecnologías necesarias

Bcryptjs

Bcrypt es una función de hashing de passwords. Lleva incorporado un valor llamado **salt**, que es un fragmento aleatorio que se usará para generar el hash asociado a la password, y se guardará junto con ella en la base de datos. Así se evita que dos passwords iguales generen el mismo hash y los problemas que ello conlleva, por ejemplo, ataque por fuerza bruta a todas las passwords del sistema a la vez.

Jsonwebtoken

JSON Web Token es un estándar abierto basado en JSON propuesto por IETF para la creación de tokens de acceso que permiten la propagación de identidad y privilegios

Front-End

React

Es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

¿Qué iba a usar, Angular? Se escogió React porque resultaba más fácil de usar e integrar en las tecnologías backend ya escogidas.

Axios

Axios es un Cliente HTTP basado en promesas para node.js y el navegador. Es isomorfico (= puede ejecutarse en el navegador y nodejs con el mismo código base). En el lado del servidor usa el modulo nativo http de node.js, mientras que en el lado del cliente (navegador) usa XMLHttpRequests.

Se escogió Axios para las solicitudes HTTP y almacenamiento local de información de usuario y JWT

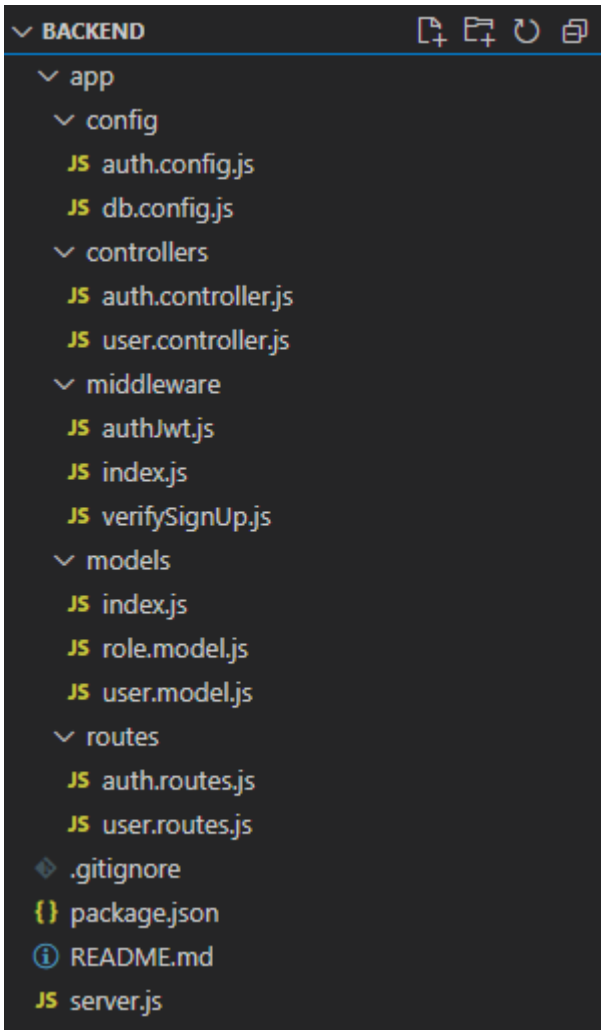
Bootstrap

Es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.

Se escogió esta librería por la familiaridad que se tenía ella.

Estructura

Back-End



El back se ha desarrollado con Node y Express.

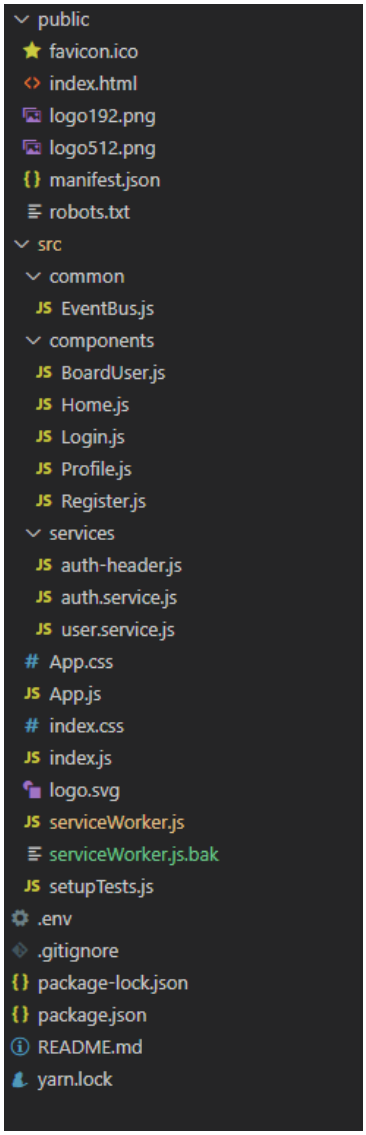
- En el directorio config se encuentran los archivos de la base de datos, Sequelize y Auth Key. Estos archivos se deben configurar antes de iniciar la aplicación.
- En el directorio controllers se encuentran dos archivos:
    - uth.controller.js: maneja las acciones de registro e inicio de sesión
    - user.controller.js: devuelve el contenido público y protegido

- En el directorio middlewares se encuentran dos archivos:
    - verifySignUp.js: verifica si el nombre de usuario o el correo electrónico están ya en uso
    - authJwt.js: Verifica el TOKEN y al usuario

- En el directorio models se encuentra los archivos Sequelize que en un principio iban a servir para dar diferentes roles al usuario.
- En el directorio routes están los archivos:
    - auth.routes.js: POST registro e inicio de sesión
    - user.routes.js: GET recursos públicos y protegidos

Por ultimo en el directorio raíz está server.js que se encarga de iniciar el servicio

Front-End



La estructura del front contiene muchos archivos de estilos e imágenes propias de la parte que se encarga del aspecto visual, además de los archivos del serviceworker. El directorio importante es:

Services:

Auth.service.js importa Axios para solicitudes HTTP y almacenamiento local para información de usuario y JWT mediante los métodos:

Login(): POST (usuario, contraseña) y guarda el JWT en Local Storage

Logout(): elimina el JWT del Local Storage

Registrer(): POST (usuario, email, contraseña)

GetCurrentUser(): obtiene la información guardada del usuario

En user.service.js tenemos los métodos para recuperar datos del servidor. En el caso de que accedamos a recursos protegidos, la solicitud HTTP necesita un encabezado de autorización.

Diagrama de secuencia de registro y login

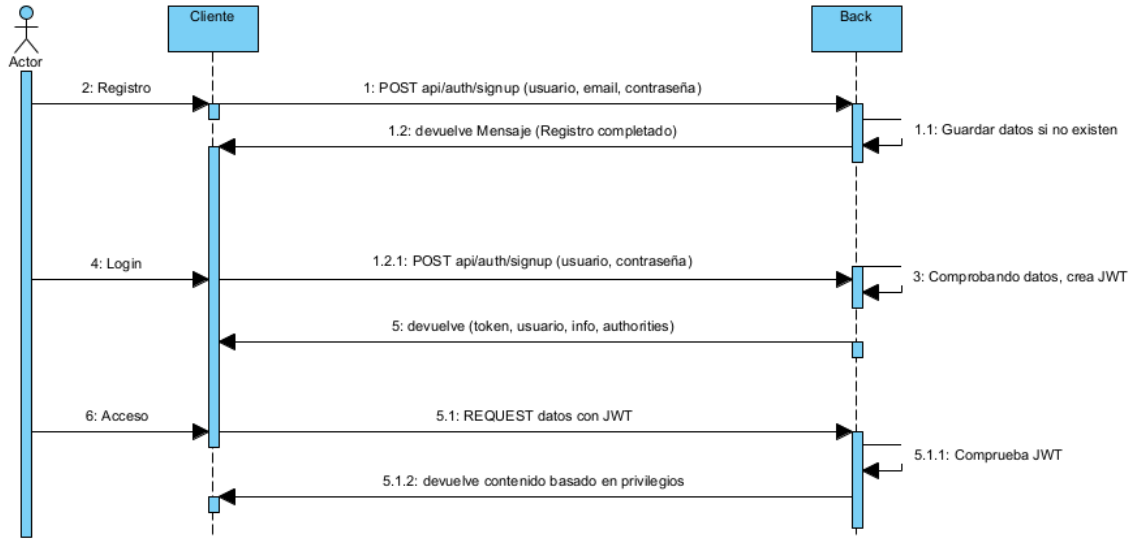


Diagrama de secuencia para Token expirado

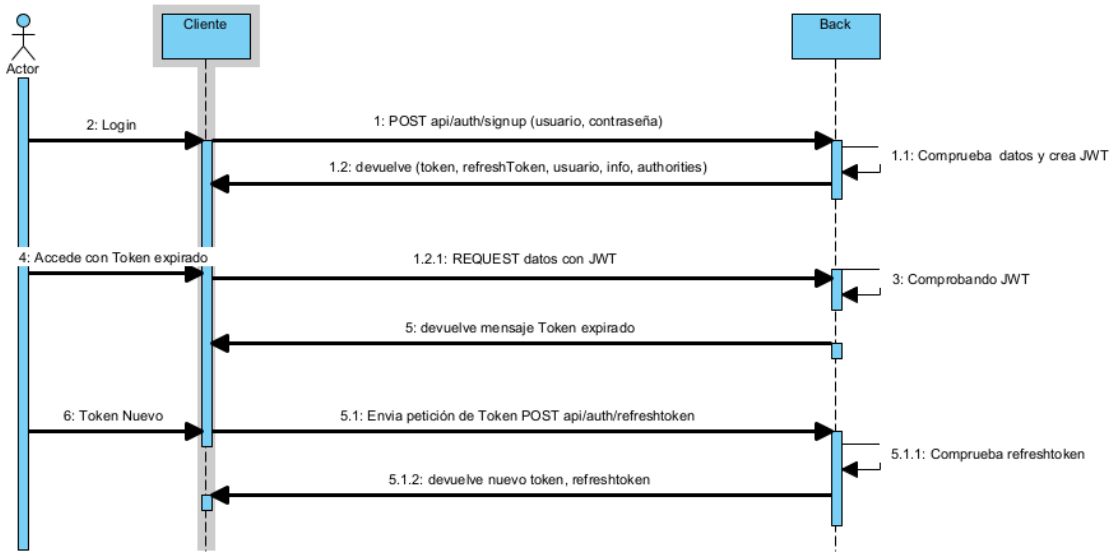


Diagrama Tecnológico