Memoria de prácticas

Desarrollo de Aplicaciones Web

Juan Hernández Acosta - Miguel Sáez Martínez
juan.hernandeza@um.es - miguel.saezm@um.es

29/5/2024

Índice

1. Introducción

En esta práctica se afronta el desarrollo de una aplicación web completa para la gestión de alquileres de bicicletas. Esta memoria detalla cómo hemos realizado la implementación de la funcionalidad y vistas, así como los problemas encontrados durante la práctica de Desarrollo de Aplicaciones Web.

2. Creación de las vistas

Inicialmente, creamos gran parte de las vistas como archivos .html individuales. Esta decisión estaba apoyada en el hecho de que nos permitía practicar tanto su apariencia como la funcionalidad relacionada con tecnologías o elementos que debíamos usar como, por ejemplo, Bootstrap.

Una vez estuvimos conformes con la apariencia general y la funcionalidad de los formularios y la barra de navegación, siendo esta un componente de React, decidimos adaptar esos archivos .html e implementar el resto de funcionalidad necesaria para convertirlos en archivos .js que contenían las vistas finales que usamos.

3. Routers y Controllers

3.1 authRoutes y userRoutes

Estos enrutadores se encargan de conectar con los controladores “authController" y “userController”, respectivamente, que gestiona todo lo relacionado con los usuarios de nuestra base como, por ejemplo, registrarlos, recuperarlos, actualizarlos y las funciones de login y logout. Es durante el proceso de login que se crea el token jwt que se mantiene durante la sesión y se destruye una vez se realiza el logout

El modelado de usuarios “users” en nuestra base de datos es:

```
CREATE TABLE User (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(255) NOT NULL,
        apellidos VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL UNIQUE,
        password VARCHAR(255) NOT NULL,
        birthdate DATE NOT NULL,
        role ENUM('user', 'admin') NOT NULL DEFAULT 'user'
);
```

3.3 stationRoutes

Este enrutador se comunica con el controlador “stationController”. Este controlador maneja la funcionalidad de las estaciones entre la que se encuentra la creación, actualización y borrado de estaciones o la obtención de las bicicletas asignadas a una estación concreta.

El modelado de estaciones “stations” en nuestra base de datos es:

```
CREATE TABLE Station (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(255) NOT NULL,
        fechaAlta DATE NOT NULL DEFAULT CURRENT_DATE,
        codigoPostal VARCHAR(255) NOT NULL UNIQUE,
        capacidad INT NOT NULL
);
```

3.4 bicycleRoutes

Este enrutador conecta con el controlador “bicycleController” donde se hace la gestión de los aspectos relacionados con las bicicletas. Las bicicletas tienen una relación bidireccional implícita de muchos a uno con las estaciones, de forma que una bicicleta puede estar solamente en una estación y una estación puede tener varias bicicletas. La funcionalidad del controlador abarca desde añadir una bicicleta, actualizarla o borrarla hasta obtener todas las bicicletas de una estación o recuperar la estación en la que esta estacionada.

El modelado de bicicletas “bicycle” en nuestra base de datos es:

```
CREATE TABLE Bicycle (
        id INT AUTO_INCREMENT PRIMARY KEY,
        marca VARCHAR(255) NOT NULL,
        modelo VARCHAR(255) NOT NULL,
        estado VARCHAR(255) NOT NULL,
        stationId INT,
        FOREIGN KEY (stationId) REFERENCES Station(id)
);
```

3.5 rentRoutes

Este enrutador conecta con el controlador “rent Controller” que maneja las operaciones de alquiler y reserva de bicicletas. Estas incluyen funciones para reservar una bicicleta, confirmar una reserva, cancelar una reserva, obtener la reserva o alquiler activo de un usuario, obtener alquileres previos, alquilar una bicicleta directamente, y terminar un alquiler. Para esto hace uso de la lógica de reserva “book” que asocia el “userId” de un usuario con el “bicycleId” de una bicicleta. Una bicicleta es reservada por un usuario y una vez esta reserva es confirmada se crea un alquiler de la misma.

El modelado de alquiler “rent” en nuestra base de datos es:

```
CREATE TABLE Rent (
        id INT AUTO_INCREMENT PRIMARY KEY,
        userId INT NOT NULL,
        bicycleId INT NOT NULL,
        fecha DATE NOT NULL,
        horaInicio TIME NOT NULL,
        duracion INT DEFAULT 30,
        estado VARCHAR(255) DEFAULT 'activo',
        FOREIGN KEY (userId) REFERENCES User(id),
        FOREIGN KEY (bicycleId) REFERENCES Bicycle(id)
);


ALTER TABLE Rent ADD CONSTRAINT FK_User_Rent FOREIGN KEY
(userId) REFERENCES User(id);

ALTER TABLE Rent ADD CONSTRAINT FK_Bicycle_Rent FOREIGN KEY
(bicycleId) REFERENCES Bicycle(id);
```

El modelado de reserva “book” en nuestra base de datos es:

```
CREATE TABLE Book (
        id INT AUTO_INCREMENT PRIMARY KEY,
        userId INT NOT NULL,
        bicycleId INT NOT NULL,
        fecha DATE NOT NULL,
        horaInicio TIME NOT NULL,
        duracion INT DEFAULT 30,
        estado VARCHAR(255) NOT NULL DEFAULT 'reservado',
        FOREIGN KEY (userId) REFERENCES User(id),
        FOREIGN KEY (bicycleId) REFERENCES Bicycle(id)
);

-- Relación de User con Book
ALTER TABLE Book ADD CONSTRAINT FK_User_Book FOREIGN KEY
(userId) REFERENCES User(id);

ALTER TABLE Book ADD CONSTRAINT FK_Bicycle_Book FOREIGN KEY
(bicycleId) REFERENCES Bicycle(id);
```

4. Roles

Nuestra aplicación ofrece dos vistas distintas en función del rol del usuario. El rol del usuario es un atributo que se elige durante el proceso de registro, siendo las opciones user o admin, y como se indica en el apartado 3.1 el token jwt creado durante el login del usuario contiene información sobre el id y el rol del mismo. Esta información se usa para determinar tanto las vistas como las funcionalidades a la que tiene acceso el usuario siendo estas diferentes entre los roles. Mientras dura la sesión este token se guarda en el almacenamiento del navegador y una vez que la sesión termina este token es borrado.

5. Tecnologías y puesta en marcha

Para el desarrollo del proyecto hemos usado diversas tecnologías como React, Express, CSS, AJAX, Bootstrap y HTML.

Uno de los componentes de React que hemos creado ha sido mapView, añadido a la vista home.js, que usa la API OpenStreetMap para mostrar un mapa. A través de ese mapa pretendíamos mostrar la ubicación de las estaciones pero debido a complicaciones en la implementación decidimos abandonar esta idea.

Para probar nuestro proyecto debemos seguir un par de pasos sencillos ejecutando comandos que pueden usarse desde un terminal de Visual Studio o el powershell de windows.

Previo

- Tener instalados los paquetes node.js, fnm (Fast Node Manager) y npm (Node package manager). Este último con los módulos react-scripts, react-router-dom, react-dom, bootstrap, web-vitals, jwt-decode, react-leaflet leaflet.

  Todos estos paquetes y módulos pueden ser instalados siguiendo estos pasos:

  1º Visita nodejs.org y descarga la versión LTS de Node.js. Node.js también incluye npm

  2º Instalar scoop (si no lo tienes ya) con los comandos:
  “Set-ExecutionPolicy RemoteSigned -scope CurrentUser”
  “iwr -useb get.scoop.sh | iex”

  3º Instalar fnm usando scoop con el siguiente comando:
  “scoop install fnm”

  4º instalar y usar node.js usando los comandos:
  “fnm install 20”
  “fnm use 20”

5º Instalar las dependencias necesarias con el comando:
“npm install react-scripts react-router-dom react-dom bootstrap web-vitals jwt-decode react-leaflet leaflet”

- Debemos asegurarnos de tener funcionando una base de datos MySQL en nuestro ordenador.

1º Abrimos una terminal y nos colocaremos en el directorio \citybike\backend y ejecutaremos el comando “node server.js”.

2º Desde otra terminal nos colocaremos en el directorio \citybike\frontend\citybike-app y ejecutaremos el comando “npm start”.

3º Si tras ejecutar el comando “npm start” no se abre nuestro navegador, podemos abrirlo manualmente e ingresar en la barra de búsqueda la dirección “localhost:3001”

4º Siguiendo todos los pasos anteriores ya podemos disfrutar de nuestro proyecto completamente desplegado y proceder a probarlo.

Adicionalmente, hemos dejado un fichero de datos “data.json” con varias bicicletas, estaciones y dos usuarios, uno de ellos con rol administrador y otro de ellos con rol usuario. Para hacer uso de estos datos en la vista “login” podemos pulsar el botón cargar Archivos para inicializar la base de datos con algunos ejemplos de prueba

Las credenciales de acceso a cada usuario son:

	Correo	Contraseña
Administrador	admin@prueba.com	admin
Usuario	usuario@prueba.com	usuario

6. Conclusión

Después de terminar este proyecto, hemos aprendido muchísimo sobre cómo funcionan las interfaces web y cómo crearlas a nuestro gusto con las funcionalidades adecuadas. Aunque nos ha llevado más tiempo del que pensábamos, debido a la implementación del backend, y algunas partes fueron más complicadas de lo esperado, en general, creemos que esta práctica ha sido muy adecuada y enriquecedora.

A pesar de los retos y los retrasos en algunas tareas, hemos visto el valor de este ejercicio práctico. Nos ha dado una visión de lo que implica el desarrollo full stack real y nos da experiencia para futuros proyectos. Ha sido una gran experiencia que sin duda nos ayudará en nuestras carreras como desarrolladores de software.