Desarrollo de aplicaciones web

Silvia Perez Ruiz
E-mail: silvia.perezr@um.es, DNI: 49215974E
Raúl Hernández Martínez
E-mail: raul.hernandezm@um.es, DNI 24423648V:

Práctica Final

Grado en Ingeniería Informática
Desarrollo de aplicaciones web
Convocatoria de Junio 2022/2023
Profesor: Francisco Javier Bermudez Ruiz (fjavier@um.es)
Entrega: 12/05/2023

UNIVERSIDAD DE
MURCIA

Índice

1 Introducción.

Durante esta documentación vamos a explicar todas las tecnologías usadas para desarrollar este proyecto y diseños seguidos para realizar la implementación.

La alternativa que hemos usado para para el backend de la aplicación es la integración con la asignatura de Arquitectura del Software, aunque hemos creado una ventana de registro, sin funcionalidad, a modo de completar la lógica para entrar en una aplicación web.

2 Aspectos a tener en cuenta del proyecto de ArSo.

Para una de las ventanas se nos ocurrió mostrar un Top3 restaurantes de la aplicación, para ello tuvimos que implementar una nueva función en el paquete de `restaurantes` y su función REST correspondiente en el paquete `restaurantes-rest`. Lo primero que añadimos fue la función getTopTres() en la clase de `ServicioRestaurante.java`.

```
@Override
  public List<ResumenRestauranteTop3> getTopTres()  throws RepositorioException {

    LinkedList<ResumenRestauranteTop3> restaurantes = new LinkedList<>();

    for (String id : repositorio.getIds()) {
      try {
        Restaurante restaurante = getRestaurante(id);
        ResumenRestauranteTop3 resumen = new ResumenRestauranteTop3();
        resumen.setNombre(restaurante.getNombre());

        if(restaurante.getValoraciones() == null)
          continue;

        resumen.setValoraciones(restaurante.getValoraciones());
        resumen.setId(restaurante.getId());
        restaurantes.add(resumen);

      } catch (Exception e) {
        e.printStackTrace();
      }
    }

    Collections.sort(restaurantes, Comparator.comparingDouble(restaurante -> restaurante.
    getValoraciones().getCalificacionMedia()));

     List<ResumenRestauranteTop3> top3Restaurantes = restaurantes.subList(0, Math.min(3,
    restaurantes.size()));


    return top3Restaurantes;


  }
```

Listing 1: ServicioRestaurante.java

Lo que hacemos es recuperar la lista de restaurantes y ordenarlas por calificación media mediante un stream y de ahí cogemos una sublista de los 3 primeros.

La siguiente parte era añadir una función REST que llama a la implementada anteriormente.

```
1
2  @GET
3  @Path("/sitiosTuristicos/{x}/{y}")
4  @Produces({ MediaType.APPLICATION_JSON })
5  public Response getAllSitiosTuristicos(
6    @ApiParam(value = "x", required = true) @PathParam("x") double x,
7    @ApiParam(value = "y", required = true) @PathParam("y") double y) throws Exception {
8
9    List<SitioTuristico> resultado = servicio.getAllSitiosTuristicos(x,y);
10
11   return Response.ok(resultado).build();
12 }
```

Listing 2: RestauranteControladorRest.java

Para la ventana de los sitios turísticos, mostramos aquellos que ya están establecidos en el restaurante, además, mostramos aquellos sitios turísticos cercanos al punto de coordenadas del restaurante seleccionado para que el usuario pueda cambiarlos a su elección. Para llevar a cabo esta acción, hemos tenido que implementar en el proyecto de ARSO la forma de obtener esos sitios a través de una coordenada con el api de Geonames.

3 Manual de despliegue de la práctica.

Para el proyecto el proyecto de ArSo hay que ejecutar tres proyectos, pero antes de nada hay que hacer un `Maven install` de todos los proyectos menos el de `opiniones-rest` ya que no es un proyecto maven. A continuación ya si que podemos ejecutar los proyectos para poner en marcha la parte del backend.

1. Proyecto `restaurantes-rest`, hay que hacer `Maven Build` y poner el comando `jetty:run`.
2. Proyecto `pasarela`, hay que irse a la clase que se llama `PasarelaZuulApplication` y ejecutarla como una aplicación de Java.
3. Proyecto `opiniones-rest`, en la terminal de `Visual Studio Code` poner el comando `dotnet run`.

Para la de DAWeb primero hay que meterse desde la terminal a la primera carpeta de `express` y ejecutar `node app.js` y luego abrir otra terminal meterse en la carpeta de `react` y ejecutar `npm start`.

Con esto tendríamos ya desplegada todos los servicios para poder ejecutar la aplicación.

4 Arquitectura tecnológica usada.

4.1 Express.

Express es un marco de desarrollo popular para Node.js que simplifica la creación de aplicaciones web y API. Su objetivo principal es agilizar la configuración de rutas, el manejo de solicitudes HTTP y la entrega de respuestas. Una característica clave de Express es su capacidad para definir fácilmente cómo debe responder la aplicación a solicitudes HTTP específicas, como POST, GET, entre otras. Esta funcionalidad permite a los desarrolladores crear servicios y API de manera eficiente.

Para la persistencia de datos, hemos optado por utilizar una base de datos MySQL. Express.js ofrece una integración sencilla con MySQL mediante módulos de terceros como mysql2. Esta integración facilita las operaciones de base de datos y garantiza un alto rendimiento en nuestras aplicaciones.

Al emplear MySQL con Express, podemos realizar consultas y manipulaciones de datos de manera eficiente. El módulo mysql2 nos brinda funciones y métodos que simplifican la interacción con la base de datos, como establecer conexiones, ejecutar consultas y recibir los resultados.

Además, la combinación de Express.js y MySQL nos permite aprovechar las ventajas de ambos. Express.js proporciona un marco de desarrollo rápido y flexible para la creación de aplicaciones web y API, mientras que MySQL ofrece una base de datos robusta y confiable para el almacenamiento de datos.

En concreto nos conectamos a una base de datos en localhost de MySQL. Nos permite crear de forma sencilla una estructura para incidencias y usuarios.

Nuestra aplicación comienza con el archivo /bin/www, que es un archivo ejecutable de Node.js. Este archivo desempeña un papel fundamental en el inicio de nuestra aplicación. En él, se llevan a cabo varias tareas importantes para poner en funcionamiento el servidor web.

En primer lugar, se crea una instancia del servidor HTTP, que nos permite manejar las solicitudes y respuestas HTTP. Esto se logra utilizando las capacidades integradas del módulo http de Node.js.

A continuación, se asigna un puerto en el cual el servidor escuchará las solicitudes entrantes. Esto se configura mediante la definición de un número de puerto específico, como el puerto 3000, aunque también podríamos utilizar cualquier otro puerto disponible. Esto permite que nuestra aplicación esté lista para recibir peticiones y responder a ellas de acuerdo con las rutas y lógica definidas en otros archivos de nuestra aplicación.

En app.js, definimos las rutas de nuestra aplicación utilizando los routers. Estos routers son módulos separados que se encargan de definir cómo responder a las solicitudes que llegan a esas rutas específicas.

El archivo javascripts/helper-database.js es un módulo dedicado a las interacciones con la base de datos. Su función principal es facilitar la comunicación con la base de datos MySQL permitiendo conectarse a la base de datos y ejecutar consultas SQL de forma senci-

lla. Dentro de él tenemos los métodos `registerUser()`, `getIncidencias()`, `deleteIncidenciasByPlato()` y `deleteIncidenciasByRestaurante()`.

4.2 React.

React, es una biblioteca de JavaScript que se utiliza para construir interfaces de usuario. Las aplicaciones se componen de componentes reutilizables que luego se usarán para crear las páginas de la aplicación. En nuestro caso tenemos creados dos componentes, uno para el header y otro para el footer.

Estos componentes luego los integrábamos en el fichero `App.js`. En este fichero llamamos a estos componentes simplemente con poner el nombre de cada uno de ellos y luego en la ejecución de esta página los mostraba.

Una ventaja que vemos sobre React es la limpieza y la claridad del código ya que, como bien hemos dicho antes, con poner una línea con el nombre del componente es suficiente para mostrarlo.

Aunque lo comentaremos más adelante, en el fichero `App.js` lo primero que hace nada más ejecutarse es llamar a la función `useEffect()` que esto lo aprovechamos para que haga una llamada al backend de express y cargue contenido de los restaurantes para mostrarlos en esa página.

4.3 HTML, CSS y Bootstrap.

HTML es un lenguaje marcado que se usa para la estructura y contenido de una página web y se compone de una serie de etiquetas con las cuales podemos definir diferentes elementos.

CSS es la parte que se encarga de darle una apariencia al código de HTML. En nuestro caso hemos optado por una gama de colores verdosos, desde colores más oscuros hasta algunos más claritos.

Junto con HTML hemos usado también Bootstrap que en su mayoría hemos usado componentes, como por ejemplo un carrusel para la página de inicio, cards y muchos modals entre otras cosas.

Juntando estos tres lenguajes hemos creado toda la parte de ficheros `.html` que hay en la carpeta de `express`.

4.4 JavaScript.

JavaScript es un lenguaje de programación ligero, basado en prototipos, multiparadigma y de un solo hilo. Es un lenguaje incluido en documentos HTML.

JavaScript lo usamos durante toda la práctica para poder darle un sentido e información a todo lo programado en los ficheros de `.html`.

Lo más interesante que consideramos de esta tecnología es que podemos hacer llamas asíncronas al backend de JavaScript para que este realice las funciones y así pueda mostrar el contenido que le ha devuelto el backend en el frontend. Por así decirlo es como el mediador del backend y frontend.

5 Funcionalidad del proyecto.

5.1 Restaurante.

La funcionalidad principal de los restaurantes se encuentran en los archivos `allRestaurantes.js`, `altaRestaurantes.js` y `restaurante.js`.

5.1.1 Visualizar restaurantes.

Para llevar a cabo esta funcionalidad, cargamos la página junto a la obtención de todos los restaurantes que tenemos guardados en nuestra base de datos de MongoDB mediante JavaScript y llamada a express. Al obtenerlos, mostramos en forma de lista unos cards con el nombre del restaurante y sus coordenadas correspondientes. Además, podemos filtrar mediante búsqueda parcial, rango de valoraciones, por ciudad y por distancia de coordenadas. Al pinchar en el card de un restaurante en concreto nos redirigirá a la página de toda la información del restaurante.

5.1.2 Crear restaurante.

La acción de crear un restaurante nuevo lo tenemos en la parte del header de la página, ya que nos redirigirá a la página para poder crear un restaurante. En esta página tendremos que introducir el nombre y las coordenadas X e Y deseadas del nuevo restaurante que queremos crear. Al aceptar se hará uso de un llamamiento a métodos de JavaScript y express para llevar a cabo el registro del nuevo restaurante.

5.1.3 Información del restaurante.

En esta página nos mostrara los detalles del restaurante seleccionado, cargando todos los datos dependiendo de él mediante llamadas JavaScipt y express. Mostramos el nombre del restaurante, sus coordenadas, el número de platos, el número de valoraciones, calificación media de las valoraciones, las valoraciones añadidas por los usuarios y los platos que contiene el restaurante.

Aquí hablaremos sobre las valoraciones. Las valoraciones se muestran en una lista de cards donde aparece el nombre del usuario que ha proporcionado su valoración, el comentario de la valoración y la calificación que da al restaurante en forma de estrellas rellenadas mediante JavaScript.

5.1.4 Eliminar restaurante.

En la misma página de la información del restaurante tenemos la opción de poder eliminar un restaurante. Cuando optamos por esta acción, nos saldrá un modal para confirmar nuestra acción. Si aceptamos, se procederá a eliminarse el restaurante haciendo uso de llamadas JavaScript y express.

5.1.5 Editar restaurante.

En la misma página de la información del restaurante tenemos la opción de poder editar un restaurante. Nos saldrá un modal para introducir el nuevo nombre y nuevas coordenadas para el restaurante. Se actualizará el restaurante haciendo uso de llamadas JavaScript y express y se actualizara la página.

5.1.6 Añadir opinión.

En la misma página de la información del restaurante tenemos la opción de añadir una opinión al restaurante. Nos saldara un modal donde pondremos nuestro correo de usuario, el número de la calificación y el comentario de la valoración. Al aceptar, se subirá nuestra valoración al restaurante que hayamos seleccionado y se actualizara la página.

5.2 Plato.

Donde podemos ver todos los platos que tiene un restaurante, se hace en la página de un restaurante en específico. Se muestra una lista con todos los platos con su nombre, su descripción y su precio. Esto se hace que cuando se carga la página del restaurante, en el fichero de `allRestaurantes.js` se hace una comprobación se si está en la página de `resturante.html` si es así, que en este caso lo será, entonces hace varias llamadas a las funciones y entre ellas está la llamada a express para que consiga todos los platos del restaurante, una vez que se lo ha devuelto se muestran mediante una lista de cards.

Otra funcionalidad que tienen los platos es que puedes crearlos. En la misma página del restaurante aparece un botón en el cual si le das aparecerá un modal donde podrás añadir la información relevante para crear un plato. Si le damos al botón de cancelar, el modal se esconde y si le damos a crear entonces se llama a una función definida en `allRestaurante.js` que como hemos dicho antes, hará una llamada asíncrona a express y en este caso crea un plato. Al crear el plato se puede ver en la lista comentada anteriormente.

Para poder usar las siguientes funcionalidades del plato, hay que meterse dentro de su página, para ello bastará con pinchar sobre el plato que queremos ver.

En esta página podremos ver la misma información que aparecía en la lista anterior, pero más grande y justo debajo de esta información están las incidencias asociadas a este plato y la posibilidad de crear una incidencia desde ahi.

Para crear una incidencia tendremos que describir lo que queremos poner en la incidencia y darle al botón de crear incidencia. Esto lo que hace es que mediante una función en `allRestaurante.js` hará una llamada a express y este en vez de hacer una llamada a nuestro backend de ArSo, llama a nuestro `helper-dataBase` que meterá la incidencia en `MySQL`. Una vez registrada la incidencia, se recarga la página y mostrará la incidencia. Esto se hace también con una función que llama a express y recoge las incidencias de ese plato y las muestra en una lista de cards.

Las dos últimas funcionalidades del plato es que se puede borrar y que se puede editar. Esto se puede hacer con los botones que hay abajo en la página. Si se pincha en el de borrar aparece un modal que te dice que si lo quieres borrar si le das a si te lo borra. El otro te aparece otro modal en el cual puedes poner los datos que quieras cambiar y si le das a actualizar te recarga la página y te lo actualiza.

Con todo esto ya tendríamos toda la funcionalidad de plato explicada.

5.3 Usuario.

En esta parte lo que queremos resaltar es el uso de `jwt` para poder limitar al usuario a las páginas que entra.

Cuando un usuario inicia sesión al darle al botón de `GitHub` aparte de redirigirte a la pasarela de Git también guarda el usuario en `MySQL` esto al principio lo hicimos para poder guardar el usuario y poder recogerlo para poner que usuario hacia la incidencia, pero luego nos dimos cuenta que como esa información del usuario está en la cookie no hacía falta guardarlo pero ya que lo teníamos hecho lo dejamos.

En el navbar de cada página, hay un logo de un usuario que si le pinchas te despliega una lista con dos botones. El segundo es el de logout que cuando el usuario le pincha lo que hace es que borra las cookies. Esto es importante ya que tenemos, en los JavaScripts que nos interesa que si no está la cookie del token jwt entonces redirige al usuario a la página de inicio de sesión.

Para concluir con la parte del usuario, destacar que el primer botón que despliega el icono te lleva a una página que simularía con el perfil del usuario. Es una página en la que solo te muestra el nombre del usuario que esta loggueado.

5.4 Sitios turísticos.

Para la parte de los sitios turísticos, en la página donde se visualiza la información del restaurante, podemos elegir la acción de modificar los sitios turísticos del restaurante, que nos redirigirá a la página correspondiente.

Nos dará dos listas, una de sitios turísticos disponibles y otra de sitios turísticos establecidos actualmente en el restaurante.

En la de sitios disponibles, nos saldrán en forma de checkBox todos los sitios turísticos cerca del restaurante dependiendo de las coordenadas de dicho restaurante. La forma de obtener estos sitios turísticos se hace llamando a un método de JavaScript y express con las coordenadas del restaurante para utilizar el api de Geonames en el servidor. El usuario podrá seleccionar los sitios turísticos que quiera establecer para su restaurante. Una vez seleccionados se actualizará la página.

En la lista de sitios actuales, mostrara los sitios turísticos ya establecidos en el restaurante, de esta forma el usuario podrá visualizar cuales están ya.

5.5 Componente NavBar en React.

Este componente carga un NavBar, con sus redirecciones a crear un restaurante y a ver todos los restaurantes que hay. También se crea el icono del usuario donde puede meterse al perfil y cerrar sesión.

Hacemos uso de componentes ya creados en React como `AppBar`, `ToolBar`, `IconButton`, `Menu` y `MenuItem`.

También creamos como acciones y una de ellas es `handleClose()` que es que cierre el menu desplegable y la otra es `handleClick()` que es que se se abre el menu cuando se pinche en el menu.

5.6 Componente Footer en React.

Este componente es muy simple, solo muestra un texto.

6 Conclusión.

En conclusión, al trabajar con HTML, CSS, JavaScript, React y Node, hemos adquirido un amplio conocimiento y experiencia en tecnologías clave utilizadas en el desarrollo de software en el mercado laboral. Estas tecnologías nos han brindado un conjunto de herramientas versátiles y potentes para crear aplicaciones web modernas y robustas.

Durante este proyecto, hemos explorado diferentes tecnologías y frameworks, lo que nos ha permitido tener una perspectiva más amplia y tomar decisiones informadas sobre cuáles utilizar en proyectos futuros. En particular, hemos encontrado gran utilidad en el uso de componentes en React, lo cual nos ha permitido desarrollar el Front End de manera más eficiente y modular con pequeñas cosas.

Además, la integración con los microservicios de ARSO presentó dificultades significativas y nos demandó una parte considerable de tiempo y esfuerzo. Consideramos que este desafío fue uno de los aspectos más exigentes de la práctica, pero nos permitió enfrentar situaciones realistas que podríamos encontrar en proyectos a futuro.

En resumen, trabajar con HTML, CSS, JavaScript, React y Node nos ha brindado una valiosa experiencia y nos ha preparado para enfrentar los desafíos del desarrollo web. Hemos aprendido a utilizar herramientas modernas y poderosas, a resolver problemas técnicos y a tomar decisiones fundamentadas en la selección de tecnologías