Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Sistemas y Tecnologías Web

Proyecto: Iteración #3

Jacobo Labrador González

(alu0101119663@ull.edu.es)

Vlatko Jesús Marchán Sekulic

(alu0101321141@ull.edu.es)

Tanausú Falcón Casanova

(alu0101320878@ull.edu.es)

Índice

Introducción 2

Tareas finalizadas. 2

- **1. Cifrado de las contraseñas en la base de datos.** 2
- **2. Token JWT.** 2
- **3. Testing con Jest y React Testing Library.** 4
- **4. Pivotal Tracker** 7

Tareas a realizar en la iteración 4 8

Introducción

Tras la iteración 3, se han realizado las tareas planteadas en Pivotal Tracker:

- Cifrado de contraseñas
- Control inicio de sesión con JWT
- Testing de la aplicación con Jest y React Testing Library.

Tareas finalizadas.

1. Cifrado de las contraseñas en la base de datos.

Para la implementación del cifrado de las contraseñas en la base de datos se ha utilizado la librería 'bcryptjs'. Para ello, se procedió a instalar y modificar el backend en las funciones de '/signup' y '/login/'.

Para el cifrado de la contraseña se utilizó la función descrita en [1].

```
let passwordHash = await bcryptjs.hash(req.body.password, 10);
```

Figura 1: Función en '/signup'

Con esto se logra guardar un hash de la contraseña que la protege frente a filtraciones de la página web. Otra función utilizada a causa del cifrado de la contraseña es " compare" como se observa en [2]. Esta se utiliza cuando se lleva a cabo la comprobación de la contraseña en el inicio de sesión, dicha función permite comprobar si la contraseña digitada por el usuario (sin cifrar) coincide en hash guardado en la base de datos.

```
let compare = await bcryptjs.compare(passwd, account.password);
```

Figura 2: Función en '/login/'

2. Token JWT.

Con lo que respecta a la implementación del JWT para el control del inicio de la sesión, se han realizado una serie de modificaciones en el BackEnd.

Se creó el fichero ".env" que guardará las claves privadas del servidor para generar los tokens "ACCESS_TOKEN_SECRET" y "REFRESH_TOKEN_SECRET". Por otro lado, en la función "./login" [3], que se encarga de generar el token del usuario que se ha logueado para enviarselo al FrontEnd.

```
const accessToken = jwt.sign(user, process.env.ACCESS_TOKEN_SECRET);
res.send({
   "accessToken": accessToken
});
```

Figura 3: Función creación del token

Además, para ser capaz de comprobar en cada petición al servidor que el usuario se encuentra logueado mediante su sesión, se ha creado un **middleware** [4] capaz de comprobar el token del usuario.

```
/** Interface for a request */
export interface CustomRequest extends Request {
  token: string | JwtPayload;
}

/**
 * Function to verify JWT tokens
 */
const authenticateToken = async (req: Request, res: Response, next:
NextFunction) => {
  try {
    const token = req.header('authorization')?.replace('Bearer ',
'');

    if (!token) {
      throw new Error();
    }

    const decoded = verify(token, process.env.ACCESS_TOKEN_SECRET as
Secret);
    (req as CustomRequest).token = decoded;

    next();
  } catch (err) {
    res.status(401).send('Please authenticate');
  }
}
```

Figura 4: middleware

3. Testing con Jest y React Testing Library.

Las herramientas elegidas para realizar el testing de la aplicación web fueron **Jest** y **React Testing Library**. Para comenzar es necesario crear la carpeta '__test__' [5] lugar donde se almacenarán nuestros tests.

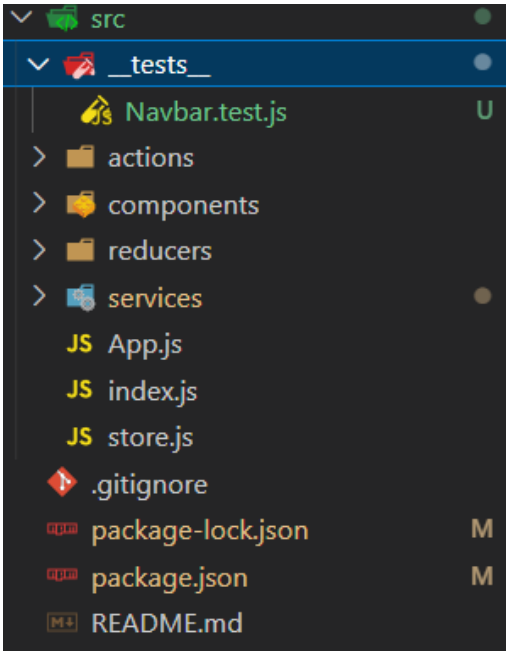

Figura 5: Estructura del directorio

Tras esto se generó un una prueba para comprobar el funcionamiento de la barra de navegación [6]. Prueba en la que se comprueba si se renderiza el título de la aplicación y el botón de la aplicación.

```
import React from "react";
import { render, screen } from "@testing-library/react";

import Navbar from '../components/Navbar';

describe("Navbar tests", () => {
  it("Must display the utopia name", () => {
    render(<Navbar />);

    const title = screen.getByText(/utopia/i);

    expect(title).toBeInTheDocument();
  });
});
```

Figura 6: Test barra de navegación

A continuación, se procede a ejecutar el test y produce un error [7]

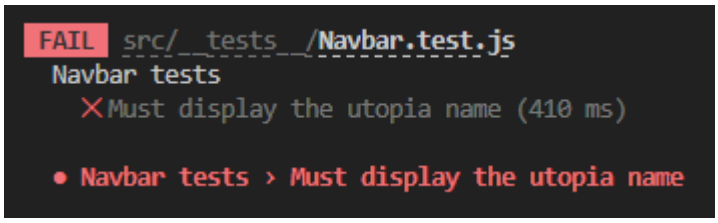

Figura 7: Resultado del test

Para que el test se ejecute exitosamente se añadió la renderización del componente [8].

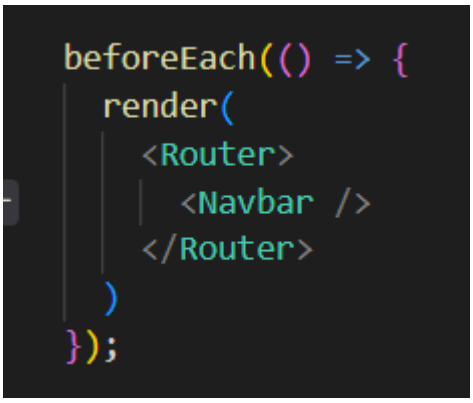

Figura 8: Renderizado del componente.

Una vez hecho esto, se volvieron a ejecutar los test [9] dando un resultado exitoso.

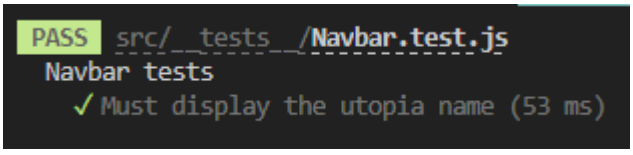

Figura 9: Resultado del test

Para el desarrollo de los siguientes test ha sido necesario crear una función render propia [10], para evitar problemas con los módulos de Axios.

```
import React from 'react';
import { render as rtlRender } from '@testing-library/react';
import { Provider } from 'react-redux';
import store from '../store';


const render = (children) => rtlRender(
  <Provider store={store}>
    {children}
  </Provider>
)

export { render };
export * from '@testing-library/react';
```

Figura 10: función render propia.

Se realizaron pruebas para cada uno de los componentes. Un ejemplo de los test realizados puede ser el hecho en la página **register** que podemos dividirlos en:

- Comprobación del display de la página comprobando que los elementos que la forma se encuentran

```
describe("SignUp component tests", () => {
  it("Must have the text Sign up", () => {
    const signupText = screen.getByText(/sign up/i);
    expect(signupText).toBeInTheDocument();
  });
  it("Must have an input for the username", () => {
    const userNameField = screen.getByLabelText(/username/i);
    expect(userNameField).toBeEnabled();
  });
  it("Must have an input for the account name", () => {
    const accountNameField = screen.getByLabelText(/account name/i);
    expect(accountNameField).toBeEnabled();
  });

  it("Must have an input for the email", () => {
    const emailField = screen.getByLabelText(/email/i);
    expect(emailField).toBeEnabled();
  });

  it("Must have an input for the password", () => {
    const passwordField = screen.getByLabelText(/password/i);
    expect(passwordField).toBeEnabled();
  });

  it("Must have a button with the text sign me up", () => {
    const buttonsign = screen.getByRole('button', {name: /sign me up/i});
    expect(buttonsign).toBeInTheDocument();
  });
});
```

Figura 11: Test realizados

```
PASS src/__tests__/Navbar.test.js
PASS src/__tests__/Register.test.js
PASS src/__tests__/Login.test.js

Test Suites: 3 passed, 3 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        2.479 s, estimated 6 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Figura 12: Resultado test.

- Correcto funcionamiento del formulario de registro.

```
describe("User Sign Up test", () => {
  it("The user login correctly", async () => {
    const userNameField = screen.getByLabelText(/username/i);
    const accountNameField = screen.getByLabelText(/account name/i);
    const emailField = screen.getByLabelText(/email/i);
    const passwordField = screen.getByLabelText(/password/i);

    fireEvent.change(userNameField, {target: {value: 'pepe'}})
    fireEvent.change(accountNameField, {target: {value: '@pepe'}})
    fireEvent.change(emailField, {target: {value: 'pepe@gmail.com'}})
    fireEvent.change(passwordField, {target: {value: '1234'}})

    const buttonsign = screen.getByRole('button', {name: /sign me up/i});
    fireEvent.click(buttonsign);

    const resultREgister = await screen.findByText(/ user created successfully/i)

    screen.debug()
    expect(resultREgister).toBeInTheDocument()
  });
});
```

Figura 13: test de funcionamiento de registro.

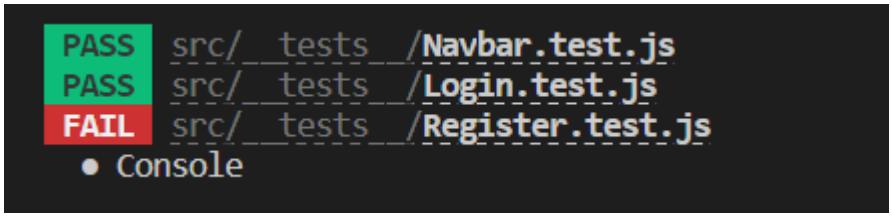

Figura 14: Resultado del test.

Este último test falla ya que actualmente nuestro formulario no notifica al usuario que se ha registrado correctamente aunque se procese la petición al servidor, este test fallido se solucionará en posteriores iteraciones,

4. Pivotal Tracker

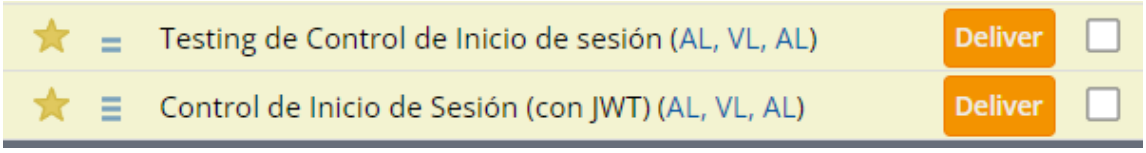

Figura 15: Tareas finalizadas en Pivotal Tracker

Algunas de estas tareas han quedado todavía por mejorar. Se seguirá refactorizando el inicio de sesión en posteriores iteraciones.

Tareas a realizar en la iteración 4

Además de la refactorización del código para la siguiente iteración, se plantean además las siguientes tareas a completar:

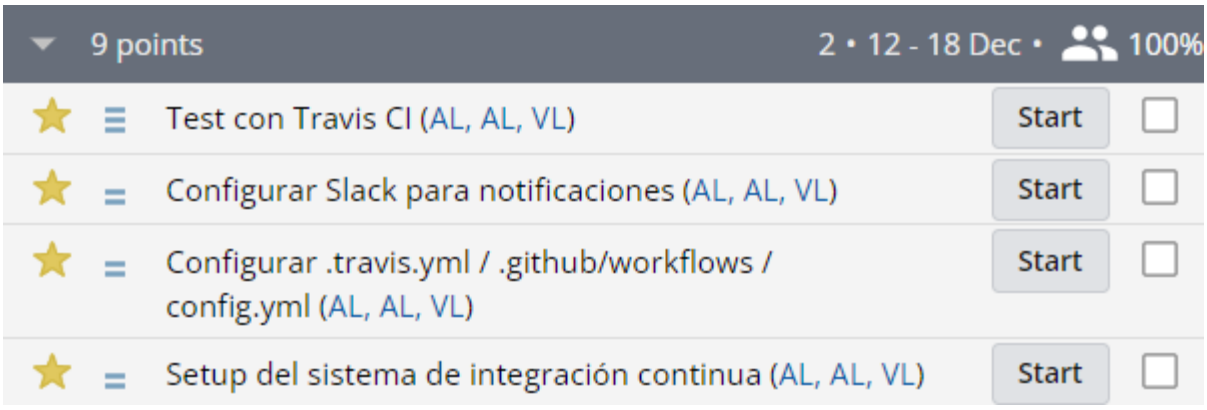

Figura 16: Tareas a realizar en la próxima iteración

Referencias

[1] Herramienta de Pivotal Tracker: https://www.pivotaltracker.com/
[2] Repositorio del proyecto: https://github.com/SyTW2223/E08.git