Módulo 11 - React Router

Desarrolla los pasos que habrá que seguir para definir todo el código de la siguiente manera:

1. Crea un proyecto React con la herramienta Create React App.
2. Crea un directorio pages con dos componentes funcionales para inicio con un formulario de autenticación con correo electrónico y contraseña, y otro para bienvenida una vez autenticado.
3. Crea un servicio con una función que compruebe las credenciales de correo electrónico y contraseña del usuario.
4. Implementa los estilos CSS de forma global en el archivo index.css.
5. Implementa ambos componentes en el componente raíz del archivo App.js, implementando las rutas de navegación con React Router.
6. Establece la lógica para que el componente formulario reciba los valores de correo electrónico y contraseña, y, en caso de ser correctos, navegue al componente de bienvenida; en caso de error, mostrará un mensaje.
7. Comprueba el funcionamiento en el navegador.

1. Crea un proyecto React con la herramienta Create React App.

npx create-react-app test_modulo_11

2. Crea un directorio pages con dos componentes funcionales para inicio con un formulario de autenticación con correo electrónico y contraseña, y otro para bienvenida una vez autenticado.

Creamos el directorio "pages" en la raíz del proyecto.Dentro del directorio "pages", creamos 2 archivos: "LoginPage.js" y "WelcomePage.js".

En "LoginPage.js" creamos un formulario de autenticación con correo electrónico y contraseña utilizando **useNavigate()** para redirigir al usuario a la página de bienvenida:

```
import React, { useState } from 'react';
import checkCredentials from '../CheckCredentials';
import { useNavigate } from 'react-router-dom';

function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
    // Verifica las credenciales utilizando la función
checkCredentials.
    if (checkCredentials(email, password)) {
      navigate('/welcome'); // redirige al usuario a la página de
bienvenida.
       // mostraremos un mensaje de éxito.
      alert('Autenticación exitosa');
    } else{
      alert('Error de autenticación.');
    }
  };

  return (
    <div className='container_form'>

      <form className='form' onSubmit={handleSubmit}>
        <h2>Inicio de sesión</h2>
        <div>
          <label htmlFor="email">Correo Electrónico:</label>
          <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
        </div>
        <div>
          <label htmlFor="password">Contraseña:</label>
          <input
            type="password"
            id="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <button type="submit">Iniciar sesión</button>
      </form>
    </div>
  );
}
```

```
export default LoginPage;
```

En **src/pages/WelcomePage.js** creamos la página de bienvenida con una imagen de fondo:

```
import React from 'react';

function WelcomePage() {
  return (
    <div>
      <h2>Bienvenido</h2>
       <p>***** Has iniciado sesión con éxito *****</p>
      <section class="welcome_img">
            <img src="./img/cafe_welcome.jpg" alt="Cafe" />
        </section>
    </div>
  );
}

export default WelcomePage;
```

Ahora ponemos en la terminal:

npm install react-router-dom@6

3. Crea un servicio con una función que compruebe las credenciales de correo electrónico y contraseña del usuario.

Creamos **CheckCredentials.js** para simular una base de datos y así poder comprobar el correo y la contraseña:

```
const usersDatabase = [
  { email: 'us1@gmail.com', password: 'password1' },
  { email: 'us2@gmail.com', password: 'password2' },

];

// Función para comprobar las credenciales del usuario.
function checkCredentials(email, password) {
  const user = usersDatabase.find((user) => user.email === email);
```

```
  if (user && user.password === password) {
    return true; // Las credenciales son válidas.
  }

  return false; // Las credenciales no son válidas.
}

export default checkCredentials;
```

y en LoginPage.js lo utilizamos(**checkCredentials**) para comprobar los datos del usuario:

```
import React, { useState } from 'react';
import checkCredentials from '../CheckCredentials';
import { useNavigate } from 'react-router-dom';

function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault();

    // Verifica las credenciales utilizando la función
checkCredentials.
    if (checkCredentials(email, password)) {
      navigate('/welcome'); // redirige al usuario a la página de
bienvenida.
       // mostraremos un mensaje de éxito.
      alert('Autenticación exitosa');
    } else{
      alert('Error de autenticación');
    }
  };

  return (
    <div>
      <h2>Inicio de sesión</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label htmlFor="email">Correo Electrónico:</label>
```

```
          <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
        </div>
        <div>
          <label htmlFor="password">Contraseña:</label>
          <input
            type="password"
            id="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <button type="submit">Iniciar sesión</button>
      </form>
    </div>
  );
}

export default LoginPage;
```

Creamos un componente **`ErrorBoundary.js`** para controlar si hay algún error y la pagina falla al cargar:

```
import React, { Component } from 'react';

class ErrorBoundary extends Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  componentDidCatch(error, errorInfo) {
    // Registra el error o realiza alguna acción personalizada aquí
```

```
    console.error(error, errorInfo);
    this.setState({ hasError: true });
  }

  render() {
    if (this.state.hasError) {
      // Puedes renderizar un mensaje de error personalizado aquí
      return <div>Ha ocurrido un error.</div>;
    }
    return this.props.children;
  }
}

export default ErrorBoundary;
```

Y ahora nos vamos a index.js para envolver la aplicación con el **`ErrorBoundary:`**

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <React.StrictMode>
      <ErrorBoundary>
        <App />
      </ErrorBoundary>
    </React.StrictMode>
  </BrowserRouter>
);
```

4. Implementa los estilos CSS de forma global en el archivo index.css.

Aquí añadimos algunos estilos CSS:

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Helvetica Neue', sans-serif;
  margin: 0;
  padding: 0;
```

```
  width: 100%;
  height: auto;
  background-color:transparent;
  background-image: url('../public/img/cafe_fondo.jpg');
  background-repeat: no-repeat;
  background-size:cover;
  background-position: center;
}
p, h1, h2{
  text-align: center;
  color: #fff;
}
header {
  background-color: #646464;
  color: #fff;
  text-align: center;
  padding: 1rem;
}
.container_form{
  display: flex;
  justify-content: space-around;
}
.form{
  background-color: #9f9f9fc2;
  width: 500px;
  color: #fffefe;
  padding: 10px;
  text-align: center;
  border: 2px solid #b4b4b4cc;
  border-radius: 30px;
}
label, input, button{
  margin: 5px;
}
main {
  padding: 2rem;
}

.welcome_img {
  text-align: center;

}
```

```css
.welcome_img img {
  max-width: 100%;
  height: auto;
  margin-top: 60px;
  width: auto;
  border-radius: 20px;
  border: 3px solid rgba(255, 255, 255, 0.874);
}

footer {
  background-color: #646464;
  color: #fff;
  text-align: center;
  padding: 1rem;
  position:relative;
  margin-top: 100%;
}
```

5. Implementa ambos componentes en el componente raíz del archivo App.js, implementando las rutas de navegación con React Router.

Aquí implementamos las dos paginas que hicimos: `LoginPage y WelcomePage`

```jsx
import React from 'react';
import { Routes, Route } from 'react-router-dom';
import LoginPage from './pages/LoginPage';
import WelcomePage from './pages/WelcomePage';


function App() {
  return (
    <>
      <Routes>
        <Route path="/login" element={<LoginPage />} />
        <Route path="/welcome" element={<WelcomePage />} />
      </Routes>
    </>
  );
}

export default App;
```

6. Establece la lógica para que el componente formulario reciba los valores de correo electrónico y contraseña, y, en caso de ser correctos, navegue al componente de bienvenida; en caso de error, mostrará un mensaje.

En LoginPage.js ponemos:

```
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault();

    // Verifica las credenciales utilizando la función
checkCredentials.
    if (checkCredentials(email, password)) {
      navigate('/welcome'); // redirige al usuario a la página de
bienvenida.
       // mostraremos un mensaje de éxito.
      alert('Autenticación exitosa');
    } else{
      alert('Error de autenticación');
    }
  };
```

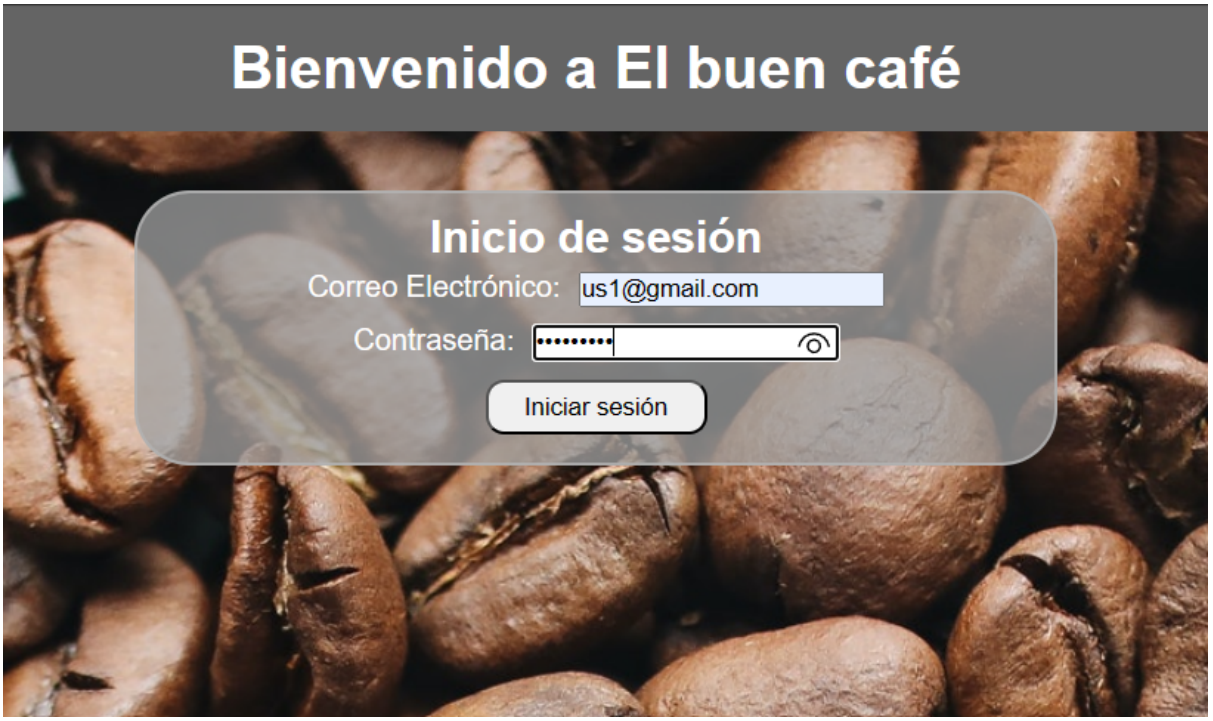
7. Comprueba el funcionamiento en el navegador.

Ahora vamos a comprobar en el navegador con el usuario y contraseña:

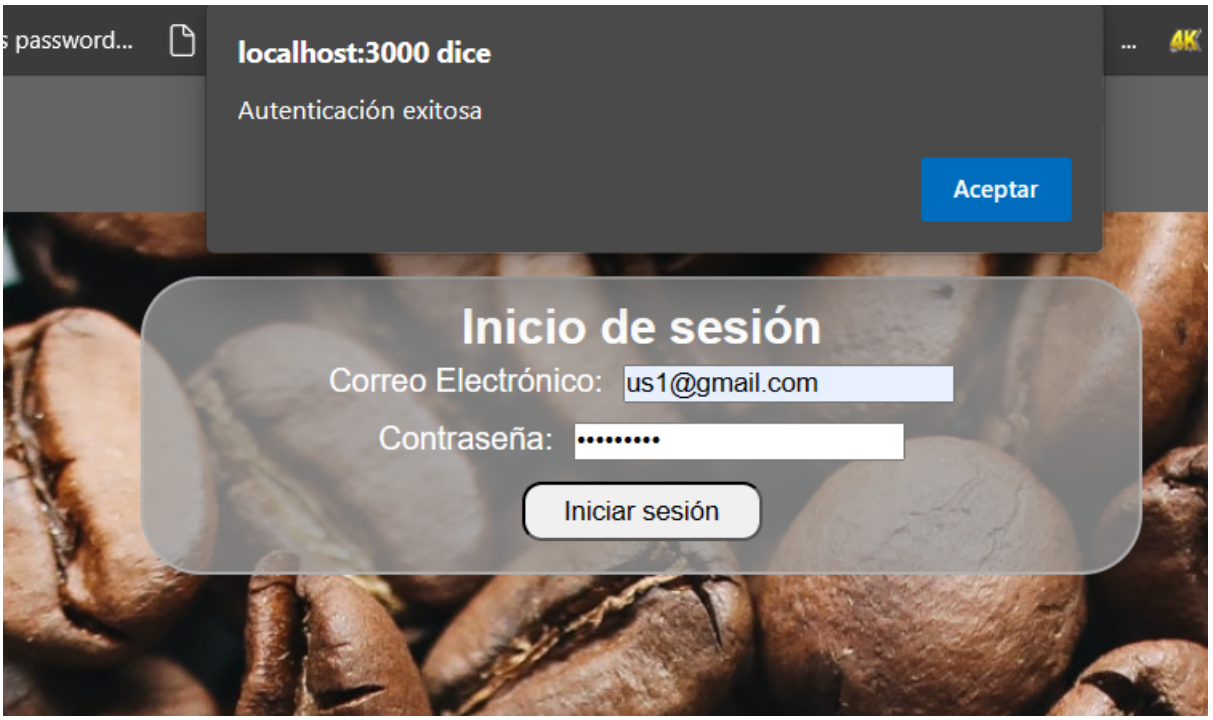
```
const usersDatabase = [
  { email: 'us1@gmail.com', password: 'password1' },
  { email: 'us2@gmail.com', password: 'password2' },

];
```

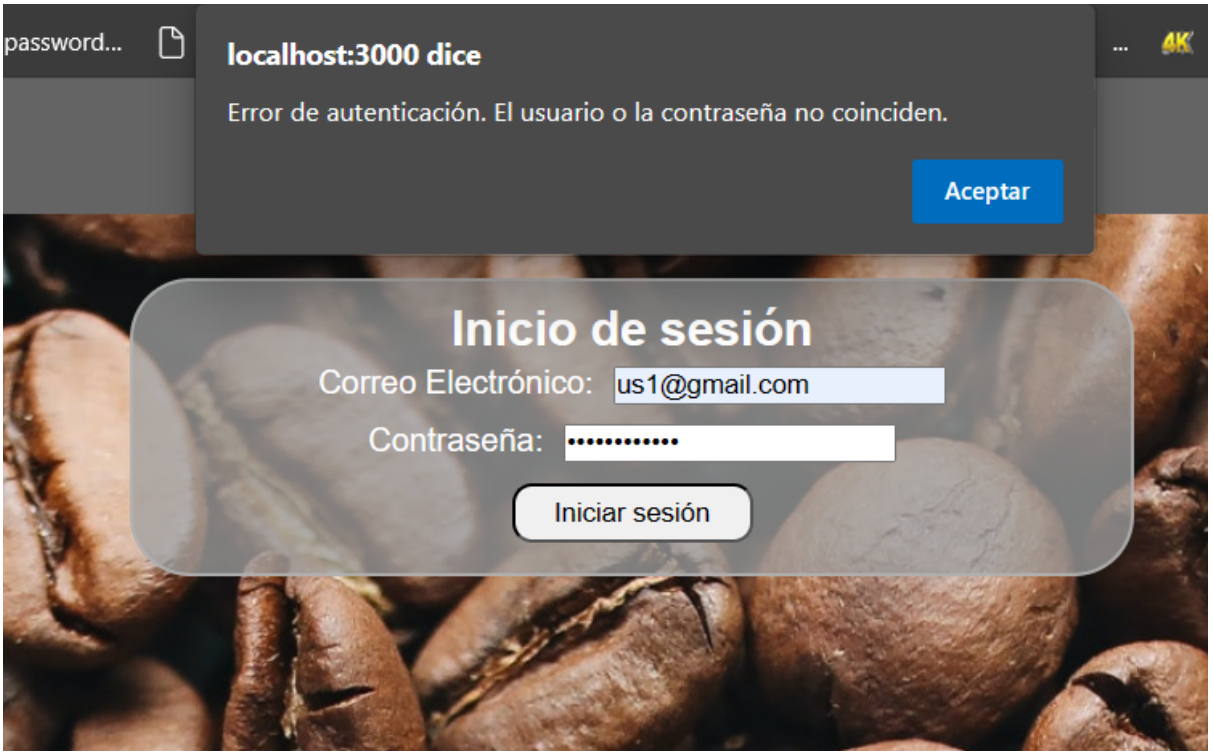
Aquí escribimos el usuario y la contraseña:



Ahora nos tiene que aparecer un alert para comprobar los credenciales:



Si no coinciden los credenciales no va ha salir otro alert:



Si todo ha ido bien no aparece la pantalla de bienvenida: