**M3S24: React: Integración con Backend (Autenticación, autorización)**

**ÍNDICE**

* [Autenticación y autorización con React](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)
  + [Rutas de SIGNUP, LOGIN y PROFILE](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)
  + [Desarrollo de "Context" para usuarios](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)
  + [Creación de usuario](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)
  + [Inicio de sesión de usuario](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)
  + [Verificación de usuarios en todas las rutas y permiso de cierre de sesión](https://github.com/U-Camp/BOOT-M3-SEM24/blob/main)

**Autenticación y autorización con React**

En esta última sesión de consolidación, estaremos vinculando nuestro Backend y Frontend, en relación a nuestros usuarios.

Seremos capaces desde el cliente:

* Crear un usuario
* Realizar un inicio de sesión
* Cerrar sesión
* Acceder a áreas privadas

Tomando en cuenta que nuestros accesos serán a través de JSON Web Tokens, es importante considerar que utilizaremos Local Storage como parte de nuestra estrategia de autenticación para guardar el token recibido.

Sin más, recuerda, activar tu Backend, el mismo que utilizaste en la sesión anterior, y comencemos.

Abre dos terminales. El primero sitúa tu localización en la raiz de la carpeta de tu Backend y ejecuta:

$ npm run start

En el segundo terminal, te sitúas sobre tu carpeta de React y hacemos un llamado adicional:

$ npm run start

**Rutas de SIGNUP, LOGIN y PROFILE**

Comenzaremos estableciendo nuestras rutas de una manera muy básica.

App.js

import './App.css';

import {

Switch,

Route,

BrowserRouter as Router

} from 'react-router-dom'

import Home from './components/Home'

import Header from './components/Layout/Header'

import Profile from './components/Profile';

import Register from './components/Register';

import Login from './components/Login';

function App() {

return (

<>

<Router>

<Header />

<Switch>

{/\* RUTAS PRIVADAS \*/}

<Route path="/perfil" component={Profile} />

{/\* RUTAS DE AUTENTICACIÓN \*/}

<Route path="/registro" component={Register} />

<Route path="/iniciar-sesion" component={Login} />

{/\* RUTAS PÚBLICAS \*/}

<Route path="/" component={Home} />

</Switch>

</Router>

</>

);

}

export default App;

Perfecto. Observa que tenemos tres tipos de rutas:

* **Rutas privadas.** Consistirán en áreas donde el usuario, **a menos que** esté autorizado, no podrá pasar. Debe contar con un token que sea validado por nuestro servidor.
* **Rutas de autenticación.** Se refieren a nuestras áreas donde el usuario realiza autenticación (registro, inicio de sesión, cierre de sesión).
* **Rutas públicas.** Son las áreas en las cuales el usuario puede acceder esté o no autorizado. Abierta a cualquier usuario.

Nos movemos ahora a nuestra sección de Register, Login y Profile. Vamos a dejar un texto simple. Esto es sólo para conectar nuestros componentes.

Register/index.js

import React from 'react'

export default function Register() {

return (

<div>

Página de registro

</div>

)

}

Login/index.js

import React from 'react'

export default function Login() {

return (

<div>

Inicia sesión

</div>

)

}

Profile/index.js

import React from 'react'

export default function Profile() {

return (

<div>

Tu perfil

</div>

)

}

Listo, revisa tus diferentes rutas.

* <http://localhost:3000/registro>
* <http://localhost:3000/perfil>
* <http://localhost:3000/iniciar-sesion>

Si aparecen los textos, avancemos.

**Creación de usuario**

Comenzamos antes de todo, generando el contexto (UserContext.js``), nuestro estado global (UserState.js) y el reducer (UserReducer.js`) para consumirse en todos nuestros componentes.

A. UserContext.js

import { createContext } from 'react'

const UserContext = createContext()

export default UserContext

* Únicamente estamos creando nuestro contexto usando createContext desde la librería de React.

B. UserState.js

import React, { useReducer } from 'react'

import UserContext from './UserContext'

import UserReducer from './UserReducer'

import axiosClient from './../config/axios'

const UserState = (props) => {

const initialState = {

user: {

username: null,

email: null,

},

authStatus: false,

loading: true

}

const [ globalState, dispatch ] = useReducer(UserReducer, initialState)

const registerUser = async (dataForm) => {

try {

const res = await axiosClient.post("/usuario/crear", dataForm)

dispatch({

type: "REGISTRO\_EXITOSO",

payload: res.data

})

} catch (error) {

}

}

return (

<UserContext.Provider value={{

user: globalState.user,

authStatus: globalState.authStatus,

loading: globalState.loading,

registerUser,

}}>

{props.children}

</UserContext.Provider>

)

}

export default UserState

En nuestro estado global, hay algunas consideraciones adicionales:

* Nuestro estado inicial cuenta con más propiedades:
  + **user.** Referido a los datos del usuario que esté autenticado.
  + **authStatus.** Me confirmará si el usuario está autenticado o no.
  + **loading.** Una propiedad que podrá indicarme si aún se está obteniendo y estableciendo la información del usuario.
* Generamos una función llamada registerUser, el cual, su objetivo, es crear un usuario con los datos que vienen del formulario, a través de nuestro cliente axios.
* Ejecutamos un dispatch que seleccione el caso de uso REGISTRO\_EXITOSO en caso de que todo haya salido correctamente.

C. UserReducer.js

const reducers = (globalState, action) => {

switch(action.type){

case "REGISTRO\_EXITOSO":

localStorage.setItem("token", action.payload.token)

return {

...globalState,

authStatus: true,

}

default:

return globalState

}

}

export default reducers

Con respecto a nuestro reducer:

* Establecemos un caso llamado REGISTRO\_EXITOSO, el cual enviará la instrucción para realizar un cambio en el estado global.
* Si ninguno coincide, contamos con un caso default que devuelve el valor tal cual se encontraba inicialmente.

Con esto listo, avanzamos a realizar los cambios en App.js para que puedan acceder todos los componentes a este estado global.

Anexaremos <UserState> y su cierre, envolviendo a todos los componentes, incluyendo <GuitarState>.

App.js

import './App.css';

import {

Switch,

Route,

BrowserRouter as Router

} from 'react-router-dom'

import Home from './components/Home'

import Header from './components/Layout/Header'

import GuitarState from './context/guitars/GuitarState'

import Profile from './components/Profile';

import Register from './components/Register';

import Login from './components/Login';

import UserState from './context/users/UserState';

function App() {

return (

<>

<UserState>

<GuitarState>

<Router>

<Header />

<Switch>

{/\* RUTAS PRIVADAS \*/}

<Route path="/perfil" component={Profile} />

{/\* RUTAS DE AUTENTICACIÓN \*/}

<Route path="/registro" component={Register} />

<Route path="/iniciar-sesion" component={Login} />

{/\* RUTAS PÚBLICAS \*/}

<Route path="/" component={Home} />

</Switch>

</Router>

</GuitarState>

</UserState>

</>

);

}

export default App;

Bien, ahora entremos al componente.

import React, { useState, useContext } from 'react'

import UserContext from './../../context/users/UserContext'

export default function Register() {

const userCtx = useContext(UserContext)

const {

registerUser

} = userCtx

const [data, setData] = useState({

username: "",

email: "",

password: ""

})

const handleChange = (event) => {

event.preventDefault()

setData({

...data,

[event.target.name]: event.target.value

})

}

const sendData = (event) => {

event.preventDefault()

registerUser(data)

}

return (

<>

<div>

<div>

<h2>

Crear cuenta

</h2>

</div>

<div>

<div>

<form onSubmit={(e) => { sendData(e) }}>

<div>

<label htmlFor="email">

Nombre de usuario

</label>

<div>

<input

id="username"

name="username"

type="text"

required

onChange={(e) => { handleChange(e) }}

/>

</div>

</div>

<div>

<label htmlFor="email">

Email

</label>

<div>

<input

id="email"

name="email"

type="email"

autoComplete="email"

required

onChange={(e) => { handleChange(e) }}

/>

</div>

</div>

<div>

<label htmlFor="password">

Password

</label>

<div>

<input

id="password"

name="password"

type="password"

required

onChange={(e) => { handleChange(e) }}

/>

</div>

</div>

<div>

<button type="submit">

Registrarme

</button>

</div>

</form>

</div>

</div>

</div>

</>

)

}

Perfecto. Llenemos el formulario.

Revisa en tu base de datos que se haya generado correctamente el registro.

Ahora vamos con el inicio de sesión.

**Inicio de sesión de usuario**

Vamos a tener una sección para generar un inicio de sesión.

Empezamos considerando el estado global (UserState.js).

UserState.js

import React, { useReducer } from 'react'

import UserContext from './UserContext'

import UserReducer from './UserReducer'

import axiosClient from './../../config/axios'

const UserState = (props) => {

const initialState = {

user: {

username: null,

email: null,

},

authStatus: false,

loading: true

}

const [ globalState, dispatch ] = useReducer(UserReducer, initialState)

const registerUser = async (dataForm) => {

try {

const res = await axiosClient.post("/usuario/crear", dataForm)

dispatch({

type: "REGISTRO\_EXITOSO",

payload: res.data

})

} catch (error) {

console.log(error)

}

}

const loginUser = async (dataForm) => {

console.log("dataForm", dataForm)

try {

const respuesta = await axiosClient.post("/api/auth", dataForm)

console.log(respuesta)

dispatch({

type: "LOGIN\_EXITOSO",

payload: respuesta.data

})

} catch (error) {

console.log(error)

}

}

return (

<UserContext.Provider value={{

user: globalState.user,

authStatus: globalState.authStatus,

loading: globalState.loading,

registerUser,

loginUser

}}>

{props.children}

</UserContext.Provider>

)

}

export default UserState

Creamos un nuevo caso de uso llamado LOGIN\_EXITOSO, en el reducer.

UserReducer.js

const reducers = (globalState, action) => {

switch(action.type){

case "LOGIN\_EXITOSO":

case "REGISTRO\_EXITOSO":

localStorage.setItem("token", action.payload.token)

return {

...globalState,

authStatus: true,

}

default:

return globalState

}

}

export default reducers

Como último movimiento, nos vamos directamente al componente de Login/index.js.

Login/index.js

import React, { useState, useEffect, useContext } from 'react'

import UserContext from './../../context/users/UserContext'

export default function Login(props) {

const userCtx = useContext(UserContext)

const {

loginUser,

authStatus,

verifyingToken

} = userCtx

const [data, setData] = useState({

email: "",

password: ""

})

useEffect(() => {

verifyingToken()

if(authStatus){

props.history.push("/perfil")

}

}, [authStatus])

if(authStatus) return null

const handleChange = (event) => {

setData({

...data,

[event.target.name]: event.target.value

})

}

const sendData = (event) => {

event.preventDefault()

loginUser(data)

}

return (

<>

<div>

<div>

<div>

<h2>

Iniciar sesión

</h2>

</div>

<form onSubmit={(e) => { sendData(e) }}>

<input type="hidden" name="remember" value="true" />

<div>

<div>

<label for="email-address">Tu correo</label>

<input

id="email-address"

onChange={(e) => { handleChange(e) }}

name="email" type="email" autocomplete="email" required placeholder="Tu correo" />

</div>

<div>

<label for="password">Password</label>

<input id="password"

name="password"

onChange={(e) => { handleChange(e) }}

type="password" autocomplete="current-password"

required

placeholder="Password" />

</div>

</div>

<div>

<button type="submit">

Comenzar

</button>

</div>

</form>

</div>

</div>

</>

)

}

**¿Qué es lo que va a pasar?**

* Tan pronto el usuario verifique su identidad en el formulario, se ejecutará un hook llamado useEffect, el cual hará un empuje hacia /perfil, a través de la librería de react-router-dom.

Perfil.js

import React from 'react'

export default function Profile() {

return (

<>

<main>

<section aria-labelledby="primary-heading">

<h1 id="primary-heading">Home</h1>

</section>

<aside>

<div>

Aquí va tu perfil

</div>

</aside>

</main>

</>

)

}

* Posteriormente, la zona del perfil saldrá con este mensaje, y acabamos de verificar al usuario.

**Verificación de usuarios en todas las rutas y permiso de cierre de sesión**

Para el manejo de sesión, necesitamos establecer antes la verificación de la autenticación en **toda** la app.

Primero, generemos una función, con habilidad de ser adquirido por los componentes a través del Context.

También escribiremos la función de cierre de sesión.

UserState.js

import React, { useReducer } from 'react'

import UserContext from './UserContext'

import UserReducer from './UserReducer'

import axiosClient from './../../config/axios'

const UserState = (props) => {

const initialState = {

user: {

username: null,

email: null,

},

authStatus: false,

loading: true

}

const [ globalState, dispatch ] = useReducer(UserReducer, initialState)

const registerUser = async (dataForm) => {

try {

const res = await axiosClient.post("/usuario/crear", dataForm)

dispatch({

type: "REGISTRO\_EXITOSO",

payload: res.data

})

} catch (error) {

console.log(error)

}

}

const verifyingToken = async () => {

const token = localStorage.getItem('token')

if(token){

axiosClient.defaults.headers.common['x-auth-token'] = token

} else{

delete axiosClient.defaults.headers.common['x-auth-token']

}

try {

const respuesta = await axiosClient.get("/usuario/verificar-usuario")

dispatch({

type: "OBTENER\_USUARIO",

payload: respuesta.data.usuario

})

} catch (error) {

console.log(error)

}

}

const loginUser = async (dataForm) => {

console.log("dataForm", dataForm)

try {

const respuesta = await axiosClient.post("/usuario/iniciar-sesion", dataForm)

console.log(respuesta)

dispatch({

type: "LOGIN\_EXITOSO",

payload: respuesta.data

})

} catch (error) {

console.log(error)

}

}

const logout = () => {

dispatch({

type: "CERRAR\_SESION"

})

}

return (

<UserContext.Provider value={{

user: globalState.user,

authStatus: globalState.authStatus,

loading: globalState.loading,

registerUser,

verifyingToken,

loginUser,

logout

}}>

{props.children}

</UserContext.Provider>

)

}

export default UserState

Genial. Trabajemos el reducer nuevo en UserReducer.js

const reducers = (globalState, action) => {

switch(action.type){

case "LOGIN\_EXITOSO":

case "REGISTRO\_EXITOSO":

localStorage.setItem("token", action.payload.token)

return {

...globalState,

authStatus: true,

}

case "OBTENER\_USUARIO":

return {

...globalState,

authStatus: true,

user: action.payload

}

case "CERRAR\_SESION":

localStorage.removeItem('token')

return {

...globalState,

user: null,

authStatus: null,

loading: false

}

default:

return globalState

}

}

export default reducers

Ahora bien, comencemos estableciendo las rutas privadas y de autenticación.

PrivateRoute.js

import React, {useContext, useEffect, useState} from 'react'

import { Route, Redirect } from 'react-router-dom'

import UserContext from '../../context/users/UserContext'

export default function PrivateRoute({ component: Component, ...props }) {

const userCtx = useContext(UserContext)

const { authStatus, verifyingToken } = userCtx

const [loading, setLoading] = useState(true)

useEffect(async () => {

await verifyingToken()

setLoading(false)

}, [authStatus])

return (

<Route {...props} render={ props => {

if(loading) return null

return authStatus ?

(<Component {...props} />)

:

(<Redirect to="/" />)

}

} />

)

}

AuthRoute.js

import React, {useContext, useEffect, useState} from 'react'

import { Route, Redirect } from 'react-router-dom'

import UserContext from '../../context/users/UserContext'

export default function AuthRoute({ component: Component, ...props }) {

const userCtx = useContext(UserContext)

const { authStatus, verifyingToken } = userCtx

const [loading, setLoading] = useState(true)

useEffect(async () => {

await verifyingToken()

setLoading(false)

}, [authStatus])

return (

<Route {...props} render={ props => {

if(loading) return null

return authStatus ?

(<Redirect to="/perfil" />)

:

(<Component {...props} />)

}

} />

)

}

Vamos a App.js y coloquemos este ruteo específico en nuestros componentes.

import './App.css';

import {

Switch,

Route,

BrowserRouter as Router

} from 'react-router-dom'

import Home from './components/Home'

import Header from './components/Layout/Header'

import GuitarState from './context/guitars/GuitarState'

import Profile from './components/Profile';

import Register from './components/Register';

import Login from './components/Login';

import UserState from './context/users/UserState';

import PrivateRoute from './components/Auth/PrivateRoute'

import AuthRoute from './components/Auth/AuthRoute'

function App() {

return (

<>

<UserState>

<GuitarState>

<Router>

<Header />

<Switch>

{/\* RUTAS PRIVADAS \*/}

<PrivateRoute path="/perfil" component={Profile} />

{/\* RUTAS DE AUTENTICACIÓN \*/}

<AuthRoute path="/registro" component={Register} />

<AuthRoute path="/iniciar-sesion" component={Login} />

{/\* RUTAS PÚBLICAS \*/}

<Route path="/" component={Home} />

</Switch>

</Router>

</GuitarState>

</UserState>

</>

);

}

export default App;

Bien. Vamos a manejar nuestro header para saber exactamente dónde estamos ubicados.

Adicionalmente, crearemos un enlace que me permita cerrar sesión.

Header.js

import React, { useContext } from 'react'

import {

Link

} from 'react-router-dom'

import UserContext from '../../context/users/UserContext'

export default function Header() {

const ctx = useContext(UserContext)

const { logout, user } = ctx

return (

<div>

<ul>

<li>

<Link to="/">

Inicio

</Link>

</li>

{

user?.username ?

<>

<li>

<Link to="/perfil">

Perfil

</Link>

</li>

<li onClick={() => { logout() }}>

<Link to="#">

Cerrar sesión

</Link>

</li>

</> :

<>

<li>

<Link to="/registro">

Registro

</Link>

</li>

<li>

<Link to="/iniciar-sesion">

Iniciar sesión

</Link>

</li>

</>

}

</ul>

</div>

)

}

Maravilloso. El resultado permitirá aparecer el iniciar sesión, registro y cerrar sesión, dependiendo si el usuario cuenta con un token, o no.