**M3S22: React: Context API**

**Hola, espero hayas tenido un muy buen día, te doy la bienvenida a la semana 22 de tu Bootcamp, ya cada vez estamos más cerca de finalizar, ¡que emoción!. Para esta semana estaremos trabajando sobre React: Context API, te dejo por aquí el listado con el contenido a trabajar.**

**ÍNDICE**

* [React: Context API](https://github.com/U-Camp/BOOT-M3-SEM22#react-context-api)
  + [Preparación de ambiente](https://github.com/U-Camp/BOOT-M3-SEM22#preparaci%C3%B3n-de-ambiente)
  + [Context API (useContext) y estado global](https://github.com/U-Camp/BOOT-M3-SEM22#context-api-usecontext-y-estado-global)
  + [Reducers (useReducer)](https://github.com/U-Camp/BOOT-M3-SEM22#reducers-usereducer)

**React: Context API**

Una vez construido nuestro Backend, es momento de consumir sus datos y que nuestros usuarios puedan leerlos y manipularlos.

Es por ello, que utilizaremos React como nuestra aplicación de Frontend, el cual se conectará a nuestros servidores y extraerán los datos.

Sin embargo, antes de realizar estas acciones, prepararemos nuestro contexto global para su uso dentro de la aplicación con Context API, realizando lo siguiente:

**Preparación de ambiente**

Empecemos abriendo tu navegador (puntualmente Firefox o Google Chrome) y descargaremos este plugin.

* Chrome: <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=es>
* Firefox: <https://addons.mozilla.org/es/firefox/addon/react-devtools/>

Bien. Ahora crearemos una nueva carpeta uclass-client y en nuestra terminal accederemos a la carpeta para posteriormente crear una aplicación en React, como lo hemos hecho anteriormente.

npx create-react-app .

Ahora bien, una vez instalado, corremos la aplicación y nos aparecerá el átomo de entrada.

npm run start

Ok. Vamos a comenzar estableciendo nuestras dependencias.

* react-router-dom. Para el manejo de rutas dentro de la aplicación de React.
* axios. Para la extracción de datos accediendo a través de una API.

Las instalamos:

npm install react-router-dom axios

Crearemos carpetas y archivos dentro de React.

Generaremos este conjunto de archivos y carpetas en la estructura de nuestra aplicación.

**Notas importantes**

* Hay un archivo que debemos crear para nuestras variables de entorno, el cual es .env, en la raiz del proyecto, afuera del src.
* Todo se coloca dentro de la carpeta src. Ten cuidado de no dejar nada fuera.

|- .env

|- src/

|- components/

|- Layout/

|- Header.js

|- Profile/

|- index.js

|- Register/

|- index.js

|- Login/

|- index.js

|- Home/

|- index.js

|- GuitarList.js

|- Auth/

|- AuthRoute.js

|- PrivateRoute.js

|- config/

|- axios.js

|- token.js

|- context/

|- guitars/

|- GuitarContext.js

|- GuitarState.js

|- GuitarReducer.js

|- users/

|- UserContext.js

|- UserState.js

|- UserReducer.js

Te comparto una imagen para que veas cómo debe verse:

Observarás que hemos incluido varios archivos nuevos que hasta el momento no tendrás conocimiento de su funcionamiento. **No te preocupes.**

Conforme vayamos avanzando, las exploraremos.

Por ahora, avancemos.

Entremos a App.js y generemos nuestras rutas.

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'

function App() {

return (

<>

<Router>

<Header />

<Switch>

{/\* RUTAS PÚBLICAS \*/}

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

</Switch>

</Router>

</>

);

}

export default App;

Bien. Generamos el contenido de tres componentes (Home, GuitarList y Header).

Home.js

import React from 'react'

import GuitarList from './GuitarList'

export default function Home () {

return (

<>

<p>Este es el home</p>

<GuitarList />

</>

)

}

GuitarList.js

import React from 'react'

export default function GuitarList() {

return (

<div>

Esta es la lista de guitarras

</div>

)

}

Header.js

import React from 'react'

export default function Header() {

return (

<div>

Este es el header

</div>

)

}

Después de agregar nuestros tres componentes, estamos listos para obtener esta pantalla:

**Context API (useContext) y estado global**

Bien, comencemos a establecer nuestros estados globales.

¿Qué es un estado global? El estado global es una propuesta en la cual guardamos valores en un área de la aplicación disponible a todos los componentes de manera directa, sin necesidad de pasar por props (prop drilling). Es extremadamente útil para aplicaciones en las cuales estamos extrayendo múltiples tipos de datos y debemos entregarlas a muchos componentes.

En esta sesión, nos enfocaremos únicamente en extraer los datos de las guitarras con React.

Entremos a nuestro archivo de GuitarContext.js y realicemos

GuitarContext.js

import { createContext } from 'react'

const GuitarContext = createContext()

export default GuitarContext

* En este archivo, crearemos un contexto. Proveniente de la librería de react.
* Exportamos para darle disponibilidad al archivo.

Ahora bien, accedemos al archivo de GuitarState.js para generar nuestro estado global. Importaremos el contexto también.

GuitarState.js

import React from 'react'

import GuitarContext from './GuitarContext'

const GuitarState = (props) => {

// 1. ESTADO INICIAL

const initialState = {

guitars: [

{

id: 0,

nombre: "Blaze MGR",

precio: 2619

}

]

}

// 2. DISPATCHING Y REDUCERS

// 3. FUNCIONES

// NOS VAN A AYUDAR A CAPTURAR LOS EVENTOS DE LOS COMPONENTES

// 4. RETORNO DE ESTADO GLOBAL

return (

<GuitarContext.Provider

value={{

guitars: initialState.guitars

}}

>

{ props.children }

</GuitarContext.Provider>

)

}

export default GuitarState

Observa que contamos con una función que está dividida en 4 secciones:

* **Estado Inicial.** Es el valor inicial (usualmente un objeto) que contiene diferentes propiedades que contendrán nuestros datos. Estos datos React los entregará a los componentes.
* **Dispatching & Reducers**. A través de un Hook que utilizaremos más adelante, construiremos un enlace entre los eventos de los usuarios y funciones que disparen cambios en el estado inicial, convirtiéndolo en un estado global manipulable por el usuario.
* **Eventos**. Son las funciones que al ejecutarse por los componentes o nuestros usuarios, las vincularemos con otro tipo de funciones (llamados Reducers) que alterarán el estado global.
* **Retorno del estado global**. Como parte del retorno, estableceremos un Provider, el cual proveerá todos estos datos a los componentes, para ser consumidos cuando el usuario o el cliente los necesite.

Finalmente, proveeremos este estado a nuestro App.js.

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'

function App() {

return (

<>

<GuitarState>

<Router>

<Header />

<Switch>

{/\* RUTAS PÚBLICAS \*/}

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

</Switch>

</Router>

</GuitarState>

</>

);

}

export default App;

Excelente. Abre tu navegador, luego inspecciona el documento y verás que tienes una etiqueta nueva que se llama Components. Le das click y sitúate en GuitarState.

Del lado derecho, verás que tienes una propiedad value, derivada de nuestro contexto y dentro un arreglo de guitarras.

Bien. Esto significa que nuestros datos globales serán controlados desde el archivo GuitarState.js, el cual conecta GuitarContext.js.

Vamos ahora al componente para poder mostrar ese valor.

Abrimos en la carpeta components, el archivo de GuitarList.js

GuitarList.js

import React, {useContext} from 'react'

import GuitarContext from '../../context/guitars/GuitarContext'

export default function GuitarList() {

// Obtenemos el contexto

const ctx = useContext(GuitarContext)

// Extraemos los datos a través de desestructuración de objetos.

const { guitars } = ctx

return (

<div>

Esta es la lista de guitarras

{/\* Mostramos los datos \*/}

{

guitars.map((e) => {

return (

<div key={e.id}>

<h1>{e.nombre}</h1>

<p>{e.precio}</p>

</div>

)

})

}

</div>

)

}

El resultado será este:

* Aparecerá el valor directamente en la pantalla.
* Podemos descargar del contexto cualquier propiedad que necesitemos. Y podemos apreciarlo desde la sección de Components en nuestro inspector.

Por ahora, contamos con una guitarra como valor inicial escrita por nosotros.

**Reducers (useReducer)**

Como último concepto de esta sesión, es momento de explorar useReducer.

useReducer es un Hook que permite aceptar una función reducer y un estado inicial. Al momento de su invocación, retornará un arreglo con dos valores. El primero será nuestro estado global y el segundo será una otra función llamada dispatch, que gestionará y hará los cambios hacia el estado global.

Entramos a GuitarState.js y realizamos los cambios correspondientes.

import React, {useReducer} from 'react'

import GuitarContext from './GuitarContext'

import guitarReducer from './GuitarReducer'

const GuitarState = (props) => {

// 1. ESTADO INICIAL

const initialState = {

guitars: [

{

id: 0,

nombre: "Blaze MGR",

precio: 2619

}

]

}

// 2. DISPATCHING Y REDUCERS

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

// 3. FUNCIONES

// NOS VAN A AYUDAR A CAPTURAR LOS EVENTOS DE LOS COMPONENTES

// 4. RETORNO DE ESTADO GLOBAL

return (

<GuitarContext.Provider

value={{

guitars: globalState.guitars

}}

>

{ props.children }

</GuitarContext.Provider>

)

}

export default GuitarState

Listo. Entramos a GuitarReducer.js y agregamos:

const reducer = (globalState, action) => {

switch(action.type) {

default:

return globalState

}

}

export default reducer

**¿Qué ocurrió?**

* Si observamos nuestro contenido muestra exactamente igual todo. La única diferencia es que, ahora, ya podemos a través de los eventos cambiar nuestro estado global de una manera más efectiva.
* Cuando entremos a la siguiente sesión a la extracción de datos, será mucho más fácil y organizada hacerlo.

Toma tu tiempo para ver toda esta sesión. Fue una configuración extensa.

**¿Lograste la configuración solicitada?, recuerda que puedes consultar el contenido cada vez que lo consideres y de igual forma puedes hacerle preguntas a tus coaches en caso de ser necesario.**

**Por ahora, sería todo el contenido nuevo por aprender en esta semana, ¡nos vemos en la UClass!**