MASTER EN REACT

[**1-** **Preparar entorno** 2](#_Toc126065018)

[**2-** **Proyecto** 3](#_Toc126065019)

[**3-** **Componente** 4](#_Toc126065020)

[**4-** **HOOK useState** 5](#_Toc126065021)

[**5-** **HOOK useEffect** 6](#_Toc126065022)

[**6-** **AJAX** 7](#_Toc126065023)

[**7-** **Formularios** 9](#_Toc126065024)

[**8-** **Router y navegación** 10](#_Toc126065025)

[**9-** **HOOK useParams** 11](#_Toc126065026)

[**10-** **HOOK useNavigate** 12](#_Toc126065027)

[**11-** **Subrutas** 13](#_Toc126065028)

[**12-** **HOOK useRef** 15](#_Toc126065029)

[**13-** **HOOK useMemo** 16](#_Toc126065030)

[**14-** **HOOK useCallBack** 16](#_Toc126065031)

[**15-** **HOOK useReducer** 17](#_Toc126065032)

[**16-** **HOOK useContext** 18](#_Toc126065033)

[**17-** **HOOK useId** 20](#_Toc126065034)

# **Preparar entorno**

1. Instalar nodejs LTS desde la web
2. Instalar react
   1. Abrir CMD.
   2. npm install -g npm@latest
   3. npm cache clean --force
3. Comprobar que está todo instalado
   1. node --version ver si node está instalado.
   2. npm --version ver si npm está instalado.

# **Proyecto**

Crear proyecto  
npx create-react-app **nombreproyecto**

Iniciar proyecto

1. cd nombreproyecto
2. npm start
3. Abro navegador y pongo localhost:3000

# **Componente**

Es un bloque de nuestra página. Tiene parte visual y parte de lógica (variables y funciones).

1. Crear la carpeta src/**components**.  
   Dentro, creo el archivo MiComponente.js y uso el atajo **rafc** para crear la estructura base.

ComponenteA.js

import React from 'react';

export const EstadoA = () => {

    return (

        <p>Este es el primer componente</p>

    );

}

MiComponente.js

import React, {Component} from 'react';

class MiComponente extends Component{

    render(){

        return(

            <h1>Hola, soy el componente MiComponente</h1>

        );

    }

}

export default MiComponente;

App.js

import { ComponenteA } from './components/ComponenteA';

import MiComponente from './components/MiComponente';

<ComponenteA />

<MiComponente />

# **HOOK useState**

Me permite **modificar** el valor de un **estado**.

const [ nombre, setNombre ] = useState('Kevin Arcas');

const cambiarNombreAle = () => {

    setNombre('Alejandra Ramos')

}

const cambiarNombre = (nombreVar) => {

    setNombre(nombreVar)

}

const escribirNombre = (e) => {

    setNombre(e.target.value)

}

<button onClick={cambiarNombreAle}>Cambiar nombre por Ale</button>

<button onClick={ () => cambiarNombre('Jose Luis') }>Cambiar nombre por variable</button>

<input type='text' onChange={ escribirNombre } />

# **HOOK useEffect**

Produce un **efecto desencadenado** cuando algo ocurre en la aplicación.

Saca un console.log, cuando **carga o** se **modifica** el **componente o algún estado**.

useEffect( () => {

    console.log('Componente cargado o cambio en él.');

} );

Para ejecutar useEffect solo al **cargar** el **componente**, pongo [ ] al final

useEffect( () => {

    console.log('Componente cargado.');

}, []);

Para ejecutar useEffect al **cargar o modificar** un **estado concreto**, pongo el estado dentro de [ ]

useEffect( () => {

console.log('Usuario modificado.');

}, [usuario]);

Ejecuto algo cuando se **monta** el componente y otra cosa cuando se **desmonta** el componente.

useEffect( () => {

    console.log('El componente Aviso está montado.');

    /\* Cuando se DESMONTA el componente \*/

    return( () => {

        console.log('Aviso DESMONTADO.')

    });

}, []);

También está el **useLayoutEffect**, que **es similar al useEffect, pero se ejecuta antes.**

# **AJAX**

**1º** Hago la petición AJAX, la cual se puede hacer de 2 formas: Promesas o por Async/await.

1. Promesas

const getUsuariosAjaxPromesa = () => {

    fetch('https://reqres.in/api/users?page=1')

        .then(respuesta => respuesta.json())

        .then(

            resultado\_final => {

                setUsuarios(resultado\_final.data)

            },

            error => {

                console.log(error)

            }

        )

}

1. Async / await

const getUsuariosAjaxAW = async() => {

    setTimeout( async() => {

        try{

            const peticion = await fetch('https://reqres.in/api/…');

            const {data} = await peticion.json();

            setUsuarios(data);

        }

        catch(error){

            console.log(error);

        }

    }, 3000);

}

**2º** Hago useEffect para cargar los datos.

useEffect( () => {

    /\* getUsuariosAjaxPromesa(); \*/

    getUsuariosAjaxAW();

}, []);

**3º** Pinto los datos.

<ul>

    {

        usuarios.map( (usuario) => {

            return (

                <li key={usuario.id}>

                    <img src={usuario.avatar} alt={usuario.id} />

                    {usuario.first\_name} {usuario.last\_name}

                </li>

            );

        })

    }

</ul>

# **Formularios**

<form onSubmit={enviaForm}>

    <input onChange={cambiarDatos} name='nombre' type='text' />

const enviaForm = (e) => {

    e.preventDefault(); /\* Con esto no se manda el formulario. \*/

    let target = e.target;

    let usuario = {

        nombre: target.nombre.value,

Voy actualizando el estado. Cojo el estado\_previo y añado el campo que está cambiando.

const cambiarDatos = (e) => {

    let campo = e.target.name;

    let valor = e.target.value;

    setUsuario( (estado\_previo) => ({

        ...estado\_previo,

        [campo]: valor

    }));

}

# **Router y navegación**

Escribo en terminal: npm install react-router-dom@6

Creo en src la carpeta **routers**  
Creo el componente **Rutas.js**

App.js

function App() {

  return (

    <Rutas />

  );

}

Nav.js

/\* Si el NavLink está activo, pone la clase 'activado' \*/

const activado = (datos) => {  
    return datos.isActive ? 'activado' : ''  
}

<nav>

  <ul>  
      <NavLink to='/' className={ activado }>Inicio</NavLink>  
      <NavLink to='/articulos' className={ activado }>Artículos</NavLink>  
  </ul>  
</nav>

Rutas.js

import { BrowserRouter, Routes, Route, NavLink } from 'react-router-dom'

...  
  return (  
    <BrowserRouter>  
      <section>  
          <Routes>  
              <Route path='/' element={ <Navigate to='/inicio' /> } />  
              <Route path='/inicio' element={ <Inicio /> } />  
              <Route path='/articulos' element={ <Articulos /> } />  
              <Route path='/proyecto/:id' element={ <Proyecto /> } />  
              <Route path='\*' element={ <Error />} />  
          </Routes>  
      </section>  
    </BrowserRouter>  
  )  
}

# **HOOK useParams**

Con este hook obtengo parámetros de la URL y los utilizo.

RouterPrincipal.js

Poniendo varias rutas, la 1ª es si viene sin parámetros, la 2ª con 1 y la 3º con 2 parámetros.

<Route path='/persona' element={ <Persona /> } />

<Route path='/persona/:nombre' element={ <Persona /> } />

<Route path='/persona/:nombre/:apellido' element={ <Persona /> } />

Persona.js

const {nombre, apellido} = useParams();

return (

    <div>  
        {  
            !nombre && !apellido &&  
            <p className='texto rojo'>No hay param1 ni param2</p>  
        }  
        {  
            nombre &&  
            <p className='texto'>{nombre}</p>  
        }  
        {  
            apellido &&  
            <p className='texto'>{apellido}</p>  
        }  
    </div>  
)

# **HOOK useNavigate**

Redirige a la subruta que pongamos entre paréntesis.

Persona.js

const redirigir = useNavigate();

const enviar = (e) => {

    e.preventDefault();

    let nombre = e.target.nombre.value;

    let apellido = e.target.apellido.value;

    if(nombre.length <=0 && apellido.length <=0){

        redirigir('/');

    }

    else{

        redirigir(`/persona/${nombre}/${apellido}`);

    }

}

<form onSubmit={enviar}>

    <input type='text' name='nombre' /><br />

    <input type='text' name='apellido' /><br />

    <input type='submit' name='enviar' value='enviar' />

</form>

Los valores que ponga en el formulario, al darle a enviar me redirige a /persona/nombre/apellido

Si no tiene valores, me redirige a la página de inicio.

# **Subrutas**

Creo la carpeta components/panel y dentro pongo varios componentes.  
En components creo el componente ControlPanel.js en el que tengo un menú enlazando a los componentes que están dentro de la carpeta panel.

En el RouterPrincipal pongo subrutas dentro del Route ControlPanel.

ControlPanel.js

<li>

    <NavLink to='/panel/inicio' >Inicio</NavLink>  
</li>  
<li>

    <NavLink to='/panel/about' >About</NavLink>  
</li>

<div>

    {/\* Aquí van los componentes del módulo panel. \*/}

    <Outlet />

</div>

RouterPrincipal.js

<Route path='/panel/\*' element={ <ControlPanel />} >

    <Route path='inicio' element={ <PanelInicio /> } />  
    <Route path='about' element={ <PanelAbout /> } />  
</Route>

**CONCLUSIÓN**

* The NavLink is used when you want to highlight a link as active. So, on every routing to a page, the link is highlighted according to the activeClassName.
* Link is for links that need no highlighting.
* a is for external links.

<NavLink to="/home" activeClassName="active" >Home</NavLink><br/>  
.active{  
  color:red;  
}

<Link to="/about">About</Link><br/>

<a href="https://www.educative.io/edpresso">Visit Edpresso </a>

# **HOOK useRef**

**Devuelve una referencia hacia un elemento.**

Creo la ref ‘nombre’, se la asigno al input, y puedo mostrar su current.value por ejemplo.

const nombre = useRef();

console.log(nombre.current.value)

<input type='text' placeholder='Nombre' ref={nombre} /> <br />

# **HOOK useMemo**

**Memoriza el resultado que imprime por pantalla un componente. Sólo se vuelve a ejecutar el componente memorizado cuando sus propiedades se modifiquen.**

Antes de ponerle el React.memo

export const Empleados = ({page}) => {

Después

export const Empleados = React.memo( ({page}) => {

Cierra paréntesis al final del componente (después del return).

Ejemplo 2:

/\* Con useMemo solo hace console cuando pulso el botón.  \*/

const memoCont = useMemo( () => contadoresPasados(contador), [contador] )

# **HOOK useCallBack**

**SOLO FUNCIONA A NIVEL DE FUNCIONES. Nos permite memorizar una función y que sólo se ejecute al realizar un cambio en el componente que yo le diga.**

Sólo ejecuta el console cuando cambio la página, no cuando cambian Gestión o Empleado.  
  
Gestion.js

const mostrarMensaje = useCallback(() => {

      console.log('mensaje');

}, [pagina])

Empleado.js

useEffect(() => {

      mensaje();  
}, [page]);

# **HOOK useReducer**

**Es una función a la que se le pasa un estado y dentro se hacen los cambios en el estado.  
Es otra forma de usar estados.**

Creo la carpeta src/reducers/ y dentro creo JuegoReducer.js

Escribo rafc y quito import React, quito return y escribo en su lugar switch.

export const JuegoReducer = (state = [], action) => {

  switch (action.type) {  
    case 'crear':  
        return [...state, action.payload]  
    default:  
        return state  
  }  
}

En components/Juego.js, importo JuegoReducer.

export const Juegos = () => {

    const init = () => {  
        return JSON.parse(localStorage.getItem('juegos')) || []  
    }  
      
    const [juegos, dispatch] = useReducer(JuegoReducer, [], init)  
  
    useEffect(() => {  
        localStorage.setItem('juegos', JSON.stringify(juegos))  
    }, [juegos]);  
  
    const getDatos = (e) => {  
        e.preventDefault()  
  
        let juego = {  
            id: new Date().getTime(),  
            titulo: e.target.titulo.value   
        }  
  
        const action = {  
            type: 'crear',  
            payload: juego  
        }  
  
        dispatch(action)  
    }

# **HOOK useContext**

**Nos ayuda con el intercambio de información entre componentes cuando tenemos un sistema de routing en nuestro proyecto.**

Creo un proyecto  
Instalo el react-router-dom con: **npm i react-router-dom@6**  
Creo un router con un menú básico:  
 1- Creo en src/router  
 2- Dentro creo el componente Rutas.js  
 3- Creo en src/components varios componentes donde irá enlazado el router.

Creo en src la carpeta **router**  
Creo el componente **Rutas.js**

App.js

function App() {

  return (

    <Rutas />

  );

}

Rutas.js

import { BrowserRouter, Routes, Route, NavLink } from 'react-router-dom'

...  
  return (  
    <BrowserRouter>  
      <section>  
          <Routes>  
              <Route path='/' element={ <Navigate to='/inicio' /> } />  
              <Route path='/inicio' element={ <Inicio /> } />  
              <Route path='/articulos' element={ <Articulos /> } />  
              <Route path='/proyecto/:id' element={ <Proyecto /> } />  
              <Route path='\*' element={ <Error />} />  
          </Routes>  
      </section>  
    </BrowserRouter>  
  )  
}

Creo la carpeta src/**context**  
Dentro creo el archivo **PruebaContext.js**

import { createContext } from 'react';

export const PruebaContext = createContext(null);

En App.js, importo el PruebaContext

<PruebaContext.Provider value='Kevin Arcas'>

      <Rutas />

    </PruebaContext.Provider>

Voy a utilizar la información compartida con el useContext en Inicio.js

const compartida = useContext(PruebaContext)

  return (

    <p>{compartida}</p>

  )

# **HOOK useId**

**Nos permite crear un ID único para el cliente y el servidor.**

De siempre, se ha utilizado para asignar un id la función:  
let id = new Date().getTime()

Con esto, podía darse el caso en el que hubiera dos id idénticos.

Con **useId** se generan id diferentes siempre, es imposible que haya 2 iguales.

# **HOOK personalizados**

Creo una carpeta en src/**hooks**.

Voy a crear un hook personalizado. Dentro de la carpeta hooks, creo el fichero **useMayus.js**  
Los hooks deben empezar por **use**.

useMayus.js

export const useMayus = (texto) => {

    const mayusculas = () => {  
        return texto.toUpperCase()  
    }  
      
    const minusculas = () => {  
        return texto.toLowerCase()  
    }  
      
    const concatenar = (anadido) => {  
        return texto+' '+añadido  
    }  
  
    return { mayusculas, minusculas, concatenar }  
}

MiComponente.js

const { mayusculas, minusculas, concatenar } = useMayus('Kevin Arcas')

    console.log(mayusculas()) //KEVIN ARCAS  
    console.log(minusculas()) //kevin arcas  
    console.log(concatenar('Monteiro') ) //Kevin Arcas Monteiro