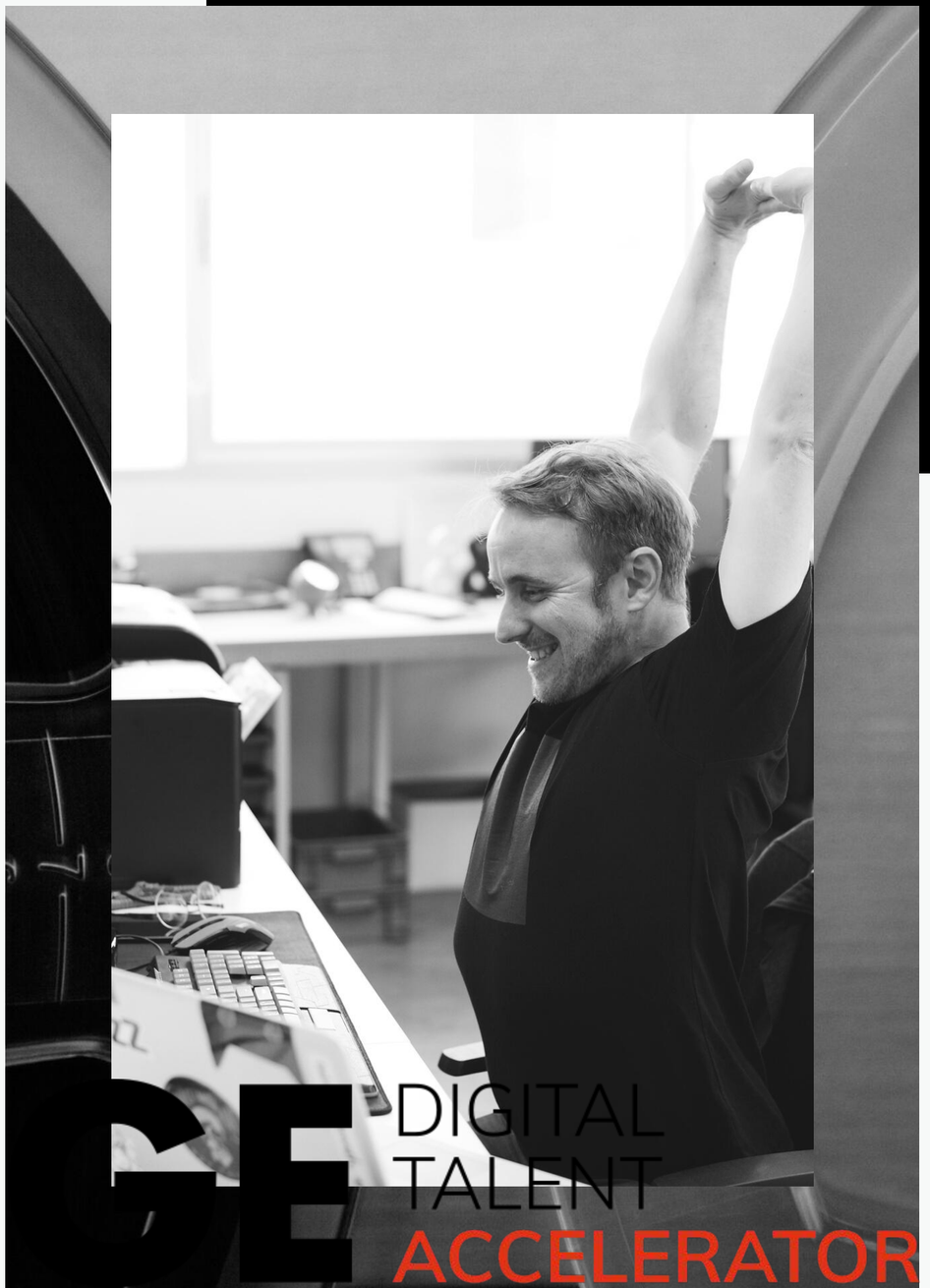
App de tareas con Context

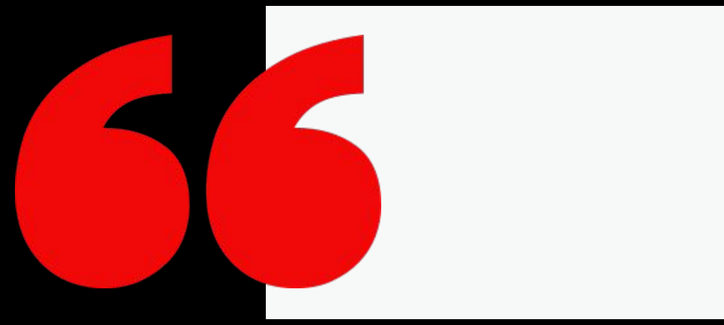
THE BRIDGE
DIGITAL TALENT ACCELERATOR

Índice

App de tareas utilizando

Context





Proyecto con Context

Vamos a crear nuestro aplicación de tareas utilizando Context

Task List

Creamos una carpeta **components** y dentro de ella la carpeta task-list donde crearemos nuestro componente **TaskList**:

```
import React from 'react'

const TaskList = () => {
  return (
    <div>TaskList</div>
  )
}

export default TaskList
```

Context

Creamos una carpeta **context** y dentro de ella el archivo **GlobalState.jsx**

```
import React, { createContext } from 'react';
import axios from 'axios'

const initialState = {
  tasks: [],
};

export const GlobalContext = createContext(initialState);
```

Definimos el estado

Creamos nuestro contexto

AppReducer

- Creamos el archivo **AppReducer.js** en la carpeta context.
- Aquí podemos ver un ejemplo de definición del **reducer** :
- Aunque generalmente se escribe usando sentencias **switch**, también puede usar if/else.

```
const tasks = (state, action) => {
    switch (action.type) {
      case "GET_TASKS":
        return {
          ...state,
          tasks: action.payload,
        };
      default:
        return state;
    }
  };
  export default tasks;
```

Provider Component

Nos importamos **useReducer** y lo inicializamos.

Nos instalamos **axios** y lo importamos.

Creamos **una función getTasks** que hará una petición para traernos las tareas, en este caso no llamaremos a una API externa sino que **llamaremos a una de nuestras API's locales**:

```
import React, { createContext, useReducer } from "react";
import AppReducer from "./AppReducer";
import axios from "axios";
. . .
export const GlobalProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);

  const getTasks = async () => {
    const res = await axios.get("http://localhost:3000/tasks");
    dispatch({
      type: "GET_TASKS",
      payload: res.data.tasks,
    });
  };

  return (
    <GlobalContext.Provider
      value={{
        tasks: state.tasks,
        getTasks,
      }}>
      {children}
    </GlobalContext.Provider>
  );
};
```

inicializamos nuestro reducer

especificamos el valor del provider

GlobalProvider

Usamos un **provider** para pasar el contexto de la aplicación actual a todos los componentes hijos.

Dentro de nuestro **GlobalProvider** ponemos nuestro **TaskList** component.

```
import TaskList from "./components/task-list/TaskList";
import { GlobalProvider } from "./context/GlobalState";

function App() {
  return (
    <div className="App">
      <GlobalProvider>
        <TaskList />
      </GlobalProvider>
    </div>
  );
}

export default App;
```

Utilizando context

Creamos nuestro componente **TaskDetail.jsx** dentro de la carpeta **task-detail,** que a su vez está en la carpeta **task-list**. Con **useContext** utilizamos el context que habíamos creado:

```jsx
import React, { useContext, useEffect } from 'react'
import { GlobalContext } from '../../../context/GlobalState'

const TaskDetail = () => {
    const { tasks, getTasks } = useContext(GlobalContext)

    useEffect(() => {
        getTasks()
    }, [])

    return tasks && tasks.map((task) => (
        <div className="task" key={task._id}>
            <h2>{task.title}</h2>
        </div>
    ))
}

export default TaskDetail
```

deleteTask action

Creamos una nueva función **deleteTask** que hará una petición para eliminar una tarea:

```
import React, { createContext, useReducer } from "react";
import AppReducer from "./AppReducer";
import axios from "axios";
. . .
export const GlobalProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
. . .
const deleteTask = async(_id) => {
    try {
        await axios.delete(`http://localhost:3000/tasks/id/${_id}`)
        dispatch({
            type: "DELETE_TASK",
            payload: _id,
          });
    } catch (error) {
        console.error(error)
    }
}

  return (
    <GlobalContext.Provider
      value={{
        tasks: state.tasks,
        getTasks,
       deleteTask,
      }}>
      {children}
    </GlobalContext.Provider>
  );
};
```

añadimos el nuevo valor al provider

AppReducer

Definimos el nuevo caso **DELETE_TASK** en nuestro reducer :

```
const tasks = (state, action) => {
    switch (action.type) {
      case "GET_TASKS":
        return {
          ...state,
          tasks: action.payload,
        };
        case "DELETE_TASK":
          return {
            ...state,
            tasks: state.tasks.filter((task) => task._id !== action.payload),
          };
      default:
        return state;
    }
  };
  export default tasks;
```

Utilizando context

- Nos traemos deleteTask al componente **TaskDetail.jsx**. Con **useContext** utilizamos el context que habíamos creado.
- Añadimos un botón que eliminará la tarea.

```jsx
import React, { useContext, useEffect } from "react";
import { GlobalContext } from "../../../context/GlobalState";

const TaskDetail = () => {
  const { tasks, getTasks, deleteTask } = useContext(GlobalContext);
  useEffect(() => {
    getTasks();
  }, []);
  const task = tasks.map((task) => {
    return (
      <div className="task" key={task._id}>
        <h1>{task.title}</h1>
        <button onClick={() => deleteTask(task._id)}>X</button>
      </div>
    );
  });
  return <>{task}</>;
};

export default TaskDetail;
```

addTask action

Creamos una nueva función **addTask** que hará una petición para crear una tarea.

```
import React, { createContext, useReducer } from "react";
import AppReducer from "./AppReducer";
import axios from "axios";
. . .
export const GlobalProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
. . .
    const addTask = async(task) => {
        try {
            const res = await axios.post('http://localhost:3000/tasks', task)
            dispatch({
                type: "ADD_TASK",
                payload: res.data.task,
              });
        } catch (error) {
            console.error(error)
        }
    }

  return (
    <GlobalContext.Provider
      value={{
        tasks: state.tasks,
        getTasks,
       deleteTask,
       addTask,
      }}>
      {children}
    </GlobalContext.Provider>
  );
};
```

añadimos el nuevo valor al provider

AppReducer

Definimos el nuevo caso **ADD_TASK** en el reducer, que cambiará el estado añadiendo la nueva tarea.

```
const tasks = (state, action) => {
    switch (action.type) {
      case "GET_TASKS":
        return {
          ...state,
          tasks: action.payload,
        };
        case "DELETE_TASK":
          return {
            ...state,
            tasks: state.tasks.filter((task) => task._id !== action.payload),
          };
        case "ADD_TASK":
          return {
            ...state,
            tasks: [...state.tasks, action.payload ],
          };
      default:
        return state;
    }
  };
  export default tasks;
```

Utilizando context

Creamos un nuevo componente **AddTask.jsx** dentro de la carpeta **AddTask** en la carpeta **TaskList**. Con **useContext** utilizamos el context que habíamos creado:

```jsx
import React, { useState, useContext } from "react";
import { GlobalContext } from "../../../context/GlobalState";

const AddTask = () => {
  const [title, setTitle] = useState("");
  const { addTask } = useContext(GlobalContext);

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log("title", title);
    addTask({ title });
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        onChange={(e) => setTitle(e.target.value)}
        name="title"
      />
      <button type="submit">Add task</button>
    </form>
  );
};
export default AddTask;
```

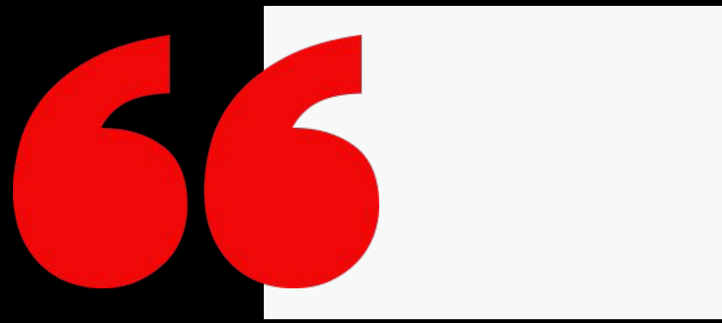
Utilizando context

Ahora en el componente **TaskList.jsx** importamos el componente **AddTask**

```
import React from 'react'
import AddTask from './AddTask/AddTask'
import Task from './Task/Task'

const TaskList = () => {
  return (
    <div>
      <AddTask/>
      <Task/>
    </div>
  )
}

export default TaskList
```



Editar una tarea

Vamos a crear un componente que será el encargado de editar una tarea

Utilizando context

Creamos un componente **EditTask.jsx** dentro de la carpeta **EditTask** en la carpeta **TaskList**, y lo importamos en **App.jsx**

```
const EditTask = () => {
  return (
    <div>EditTask</div>
  )
}

export default EditTask
```

useParams()

Los parámetros son marcadores de posición en la URL que comienzan con dos puntos, como el parámetro **:id**

React Router

Implementamos React Router en el proyecto:

npm i react-router-dom@6

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import "./App.css";
import EditTask from "./components/Tasks/EditTask/EditTask";
import TaskList from "./components/Tasks/Tasks";
import { GlobalProvider } from "./context/GlobalState";
function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <GlobalProvider>
          <Routes>
            <Route path="/" element={<TaskList />} />
            <Route path="/task/:_id" element={<EditTask />} />
          </Routes>
        </GlobalProvider>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Componente Task detail

En el componente TaskDetail **añadimos un link “edit” que nos lleve a la ruta que hemos definido** anteriormente.

Para eso necesitamos importarnos **link**

```
import React, { useContext, useEffect } from "react";
import { Link } from "react-router-dom";
import { GlobalContext } from "../../../context/GlobalState";
import "./Task.css";

const TaskDetail = () => {
  const { tasks, getTasks, deleteTask } = useContext(GlobalContext);
  useEffect(() => {
    getTasks();
  }, []);
  const task = tasks.map((task) => {
    return (
      <div className="task" key={task._id}>
        <h1>{task.title}</h1>
        <Link to={'/task/' + task._id}>Edit</Link>
        <button onClick={() => deleteTask(task._id)}>X</button>
      </div>
    );
  });
  return <>{task}</>;
};

export default TaskDetail;
```

UseParams

Podemos usar el hook **useParams** para acceder a las partes dinámicas de la URL, en este caso al **id**.

```
import { useParams } from 'react-router-dom';

const EditTask = () => {
    const {id} = useParams()
    console.log('id',id)
  return (
    <div>EditTask</div>
  )
}

export default EditTask
```

Context

Añadimos un nuevo valor a nuestro estado inicial del contexto, en este caso un objeto vacío que solo guardará una sola tarea:

```
. . .

const initialState = {
  tasks: [],
  task: {},
};

export const GlobalContext = createContext(initialState);

. . .
```

Definimos la nueva propiedad de nuestro estado

getTask action

Creamos una nueva función **getTask** que hará una petición para traernos una tarea por id:

```
import React, { createContext, useReducer } from "react";
import AppReducer from "./AppReducer";
import axios from "axios";
. . .
export const GlobalProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
. . .
    const getTask = async (_id) =>
        try {
          const res = await axios.get("http://localhost:3000/tasks/id/" + _id);
          dispatch({
            type: "GET_TASK",
            payload: res.data,
          });
        } catch (error) {
          console.error(error);
        }
      };

  return (
    <GlobalContext.Provider
      value={{
        tasks: state.tasks,
        task: state.task,
        getTask,
          . . .
      }}>
      {children}
    </GlobalContext.Provider>
  );
};
```

recibimos el _id por parámetro

le pasamos el _id de la tarea

añadimos task y getTask a value

AppReducer

Definimos el nuevo caso **GET_TASK** en el reducer

```
const tasks = (state, action) => {
    switch (action.type) {
      . . .
        case "ADD_TASK":
          return {
            ...state,
            tasks: [action.payload, ...state.tasks],
          };
        case "GET_TASK":
          return {
            ...state,
            task: action.payload,
          };

      default:
        return state;
    }
  };
  export default tasks;
```

Tarea por id

Con **useContext** usamos el context que habíamos creado y con él nos traemos la función **getTask**, y la propiedad **task** de nuestro estado que hemos definido:

```
import React , {useContext, useEffect} from 'react'
import { useParams } from 'react-router-dom';
import { GlobalContext } from '../../../context/GlobalState';

const EditTask = () => {
    const {id} = useParams()
    const { task, getTask } = useContext(GlobalContext);

    useEffect(() => {
        getTask(id)
    }, [])

    console.log('task',task)
  return (
    <div>EditTask</div>
  )
}

export default EditTask
```

editTask action

Creamos una nueva función **editTask** que hará una petición para editar una tarea por id

```
import React, { createContext, useReducer } from "react";
import AppReducer from "./AppReducer";
import axios from "axios";
. . .
export const GlobalProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
. . .
 const editTask = async (_id, task) => {
    try {
      await axios.put(`http://localhost:3000/tasks/update/id/${_id}`, task);
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <GlobalContext.Provider
      value={{
        tasks: state.tasks,
        task: state.task,
        getTask,
        editTask,
         . . .
      }}>
      {children}
    </GlobalContext.Provider>
  );
};
```

Pintamos tarea a editar

Rellenamos el formulario con la tarea que queremos editar

```
import React, { useContext, useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { GlobalContext } from "../../../context/GlobalState";

const EditTask = () => {
    const { task, getTask } = useContext(GlobalContext);
    const [title, setTitle] = useState("");
    const { _id } = useParams();

  useEffect(() =>   {
    getTask(_id)
  },[]);

  useEffect(() =>   {
    setTitle(task.title)
  },[task.title]);

  return (
    <form>
      <input
        type="text"
        onChange={(e) => setTitle(e.target.value)}
        value={ title || ""}
        name="title"
      />
      <button type="submit">Edit task</button>
    </form>
  );
};

export default EditTask;
```

Editar tarea

Añadimos el formulario en el componente para poder editar la tarea. Nos traemos la función **editTask** que habíamos creado antes y la utilizamos en el ,método **handleSubmit**

```
import React, { useContext, useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { GlobalContext } from "../../../context/GlobalState";
const EditTask = () => {
    const { task, getTask, editTask } = useContext(GlobalContext);
    const [title, setTitle] = useState("");
    const { _id } = useParams();

  const handleSubmit = (event) => {
    event.preventDefault();
    editTask(task._id, { title });
  };

  useEffect(() =>   {
    getTask(_id)
  },[]);

  useEffect(() =>   {
    setTitle(task.title)
  },[task.title]);

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        onChange={(e) => setTitle(e.target.value)}
        value={ title || ""}
        name="title"
      />
      <button type="submit">Edit task</button>
    </form>
  );
};

export default EditTask;
```

Redireccionar

Ahora con **useNavigate()** le decimos que al editar la tarea y pasado 1 segundo nos redirija a la vista principal.

```
import React, { useContext, useEffect, useState } from "react";
import { useNavigate, useParams } from "react-router-dom";
import { GlobalContext } from "../../../context/GlobalState";
const EditTask = () => {
  const { id } = useParams();
  const { task, getTask, editTask } = useContext(GlobalContext);
  const [title, setTitle] = useState("");
  let navigate = useNavigate();

  const handleSubmit = (event) => {
    event.preventDefault();
    editTask(task._id, { title });

    setTimeout(() => {
      navigate("/");
    }, 1000);
  };

...

  return (
  . . .
  );
};

export default EditTask;
```