



FULL STACK

**Comenzamos en unos
minutos**

ACADEMY
by NUMEN

CRUD en React

```
const initialDb = [  
  {  
    id: 1,  
    name: "Seiya",  
    constellation: "Pegaso",  
  },  
  {  
    id: 2,  
    name: "Shiryu",  
    constellation: "Dragon",  
  },  
  {  
    id: 3,  
    name: "Hyoga",  
    constellation: "Cisne",  
  },  
  {  
    id: 4,  
    name: "Shun",  
    constellation: "Andromeda",  
  },  
]
```

Dentro colocamos nuestro arreglo de objetos desde donde consumiremos la data.

```
{  
  id: 5,  
  name: "Ikki",  
  constellation: "Fenix",  
},  
];  
  
const CrudApp = () => {  
  return (  
    <div>  
      <h2>CRUD App</h2>  
      <div>CrudForm</div>  
      <div>CrudTable</div>  
    </div>  
  )  
}  
  
export default CrudApp
```

Creamos nuestro
componente principal
CrudApps

JSX de nuestro formulario

```
const CrudForm = () => {  
  
  return (  
    <div>  
      <h3>Agregar</h3>  
      <form>  
        <input type="text" name="name" placeholder="Nombre"/>  
        <input type="text" name="constellation" placeholder="constellation"/>  
        <input type="submit" value="Enviar"/>  
        <input type="reset" value="Limpiar"/>  
      </form>  
    </div>  
  )  
}  
  
export default CrudForm
```

```
import { useState, useEffect } from "react"
```

```
const CrudForm = () => {
```

```
  const handleChange = (e) => {}
```

```
  const handleSubmit = (e) => {}
```

```
  const handleReset = (e) => {}
```

```
  return (
```

```
    <div>
```

```
      <h3>Agregar</h3>
```

```
      <form>
```

```
        <input type="text" name="name" placeholder="Nombre" onChange={handleChange} value={form.name}/>
```

```
        <input type="text" name="constellation" placeholder="constellation"
```

```
          onChange={handleChange} value={form.constellation}
```

```
        />
```

```
        <input type="submit" value="Enviar"/>
```

```
        <input type="reset" value="Limpiar" onClick={handleReset}/>
```

```
      </form>
```


```
    </div>
```

```
  )
```


```
}
```

```
export default CrudForm
```

Preparamos las funciones manejadoras.




Pasamos las funciones a los eventos.



Definimos el estado inicial



Agregamos el useState que necesitamos para manejar el Estado de nuestro formulario.



```
import { useState, useEffect } from "react"

const initialForm = {
  name: "",
  constellation: "",
  id: null
}

const CrudForm = () => {
  const [form, setForm] = useState(initialForm)

  const handleChange = (e) => {}

  const handleSubmit = (e) => {}

  const handleReset = (e) => {}

  return (
  )
  export default CrudForm
```

JSX de nuestra Tabla

```
const CrudTable = () => {  
  return (  
    <div>  
      <h3>Tabla de Datos</h3>  
      <table>  
        <thead>  
          <tr>  
            <th>Nombre</th>  
            <th>Constelacion</th>  
            <th>Acciones</th>  
          </tr>  
        </thead>  
      </div>  
    )  
  }  
}
```

```
        <tbody>  
          <tr>  
            <td>Aioria</td>  
            <td>Leo</td>  
            <td>  
              <button>Editar</button>  
              <button>Eliminar</button>  
            </td>  
          </tr>  
        </tbody>  
      </table>  
    </div>  
  )  
}  
  
export default CrudTable
```


Pasando las referencias en CrudApp

Preparamos el useState que necesitamos para pasarle la data a la tabla.

```
const CrudApp = () => {  
  
  const [db, setDb] = useState(initialDb)  
  
  return (  
    <div>  
      <h2>CRUD App</h2>  
      <CrudForm />  
      <CrudTable data={db} />  
    </div>  
  )  
}  
  
export default CrudApp
```

Pasamos la variable del estado inicial a la tabla a través de props.

JSX de nuestra fila de tabla

```
const CrudTableRow = () => {  
  return (  
    <tr>  
      <td>Aioria</td>  
      <td>Leo</td>  
      <td>  
        <button>Editar</button>  
        <button>Eliminar</button>  
      </td>  
    </tr>  
  )  
}  
  
export default CrudTableRow
```

Armamos una validación para mostrar un mensaje en caso de no recibir datos.

```
<tbody>
{
  data.length === 0 && (
    <tr>
      <td colSpan="3">Sin datos</td>
    </tr>
  )
}
{
  data.length !== 0 && (
    data.map( caballero => <CrudTableRow key={caballero.id} caballero={caballero}/> )
  )
}
</tbody>
```

Mapeamos el CrudTableRow en el body de nuestra tabla.

Le pasamos las props que arrastramos desde CrudApp a la tabla.

Destructuramos las props

```
const CrudTableRow = ({caballero}) => {  
  return (  
    <tr>  
      <td>{caballero.name}</td>  
      <td>{caballero.constellation}</td>  
      <td>  
        <button>Editar</button>  
        <button>Eliminar</button>  
      </td>  
    </tr>  
  )  
}  
  
export default CrudTableRow
```

Colocamos las props en los table data.

Agregando la programación!

```
const CrudApp = () => {  
  const [db, setDb] = useState(initialDb);  
  const [dataToEdit, setDataToEdit] = useState(null);  
  
  const createData = (data) => {};  
  const updateData = (data) => {};  
  const deleteData = (id) => {};  
  
  return (  
    <div>  
      <h2>CRUD App</h2>  
      <CrudForm  
        createData={createData}  
        updateData={updateData}  
        dataToEdit={dataToEdit}  
        setDataToEdit={setDataToEdit}  
      />  
      <CrudTable  
        data={db}  
        setDataToEdit={setDataToEdit}  
        deleteData={deleteData}  
      />  
    </div>  
  );  
};
```

Definimos nuestro
useState para controlar
la edición de la data.

Preparamos las funciones
Insertar, Editar y Borrar
data.

Pasamos la variable de
estado y todas las
funciones como props a
los componentes.

```
const CrudForm = ({ createData, updateData, dataToEdit, setDataToEdit }) => {  
  const [form, setForm] = useState(initialForm);  
  
  const handleChange = (e) => {  
    setForm({  
      ...form,  
      [e.target.name]: e.target.value,  
    });  
  };  
  
  const handleSubmit = (e) => {  
    e.preventDefault();  
    if (!form.name || !form.constellation) {  
      alert("Datos incompletos");  
      return;  
    }  
    if (form.id === null) {  
      createData(form);  
    } else {  
      updateData(form);  
    }  
    handleReset();  
  };  
};
```

```
const handleReset = (e) => {  
  setForm(initialForm);  
  setDataToEdit(null);  
};
```

Programamos todas las funciones manejadoras de nuestro Formulario.

```
return (  
  <div>  
    <h3>Agregar</h3>  
    <form onSubmit={handleSubmit}>
```

Pasamos la función de handleSubmit como evento al formulario.

```
const CrudApp = () => {  
  const [db, setDb] = useState(initialDb);  
  const [dataToEdit, setDataToEdit] =  
    useState(null);  
  
  const createData = (data) => {  
    data.id = Date.now()  
    // console.log(data)  
    setDb([...db, data])  
  };  
};
```

Programamos la función
de Inserción.

Pasamos la data deestructurada a la referencia del CrudTableRow como props.

Deestructuramos la data en la Tabla.

```
const CrudTable = ({ data, setDataToEdit, deleteData }) => {
```

```
<tbody>
  {data.length === 0 && (
    <tr>
      <td colSpan="3">Sin datos</td>
    </tr>
  )}
  {data.length !== 0 &&
    data.map((caballero) => (
      <CrudTableRow
        key={caballero.id}
        caballero={caballero}
        setDataToEdit={setDataToEdit}
        deleteData={deleteData}
      />
    ))}
</tbody>
```

```
const CrudTableRow = ({caballero, setDataToEdit, deleteData}) => {
```

```
  let {name, constellation, id} = caballero;
```

Destructuramos las propiedades de nuestros caballeros.

```
  return (
```

```
    <tr>
```

```
      <td>{name}</td>
```

```
      <td>{constellation}</td>
```

```
      <td>
```

```
        <button onClick={() => setDataToEdit(caballero)}>Editar</button>
```

```
        <button onClick={() => deleteData(id)}>Eliminar</button>
```

```
      </td>
```

```
    </tr>
```

```
  )
```

```
}
```

Pasamos las propiedades de los caballeros al JSX

```
export default CrudTableRow
```

Pasamos las funciones como eventos.

```
const CrudForm = ({ createData, updateData, dataToEdit, setDataToEdit }) => {  
  const [form, setForm] = useState(initialForm);  
  
  useEffect(() => {  
    if (dataToEdit) {  
      setForm(dataToEdit)  
    } else {  
      setForm(initialForm)  
    }  
  }, [dataToEdit])  
}
```

Programamos el useEffect para actualizar los datos de los inputs al editar.

```
const CrudApp = () => {  
  const [db, setDb] = useState(initialDb);  
  const [dataToEdit, setDataToEdit] = useState(null);  
  
  const createData = (data) => {  
    data.id = Date.now()  
    // console.log(data)  
    setDb([...db, data])  
  };  
  
  const updateData = (data) => {  
    let newData = db.map(el => el.id === data.id ? data  
: el)  
    setDb(newData);  
  };  
};
```

Programamos la
Inserción

Programamos la
Edición.

Programamos el cambio de
titulo de Inserción a Edición.

```
<h3>{dataToEdit ? "Editar" : "Agregar"}</h3>
```

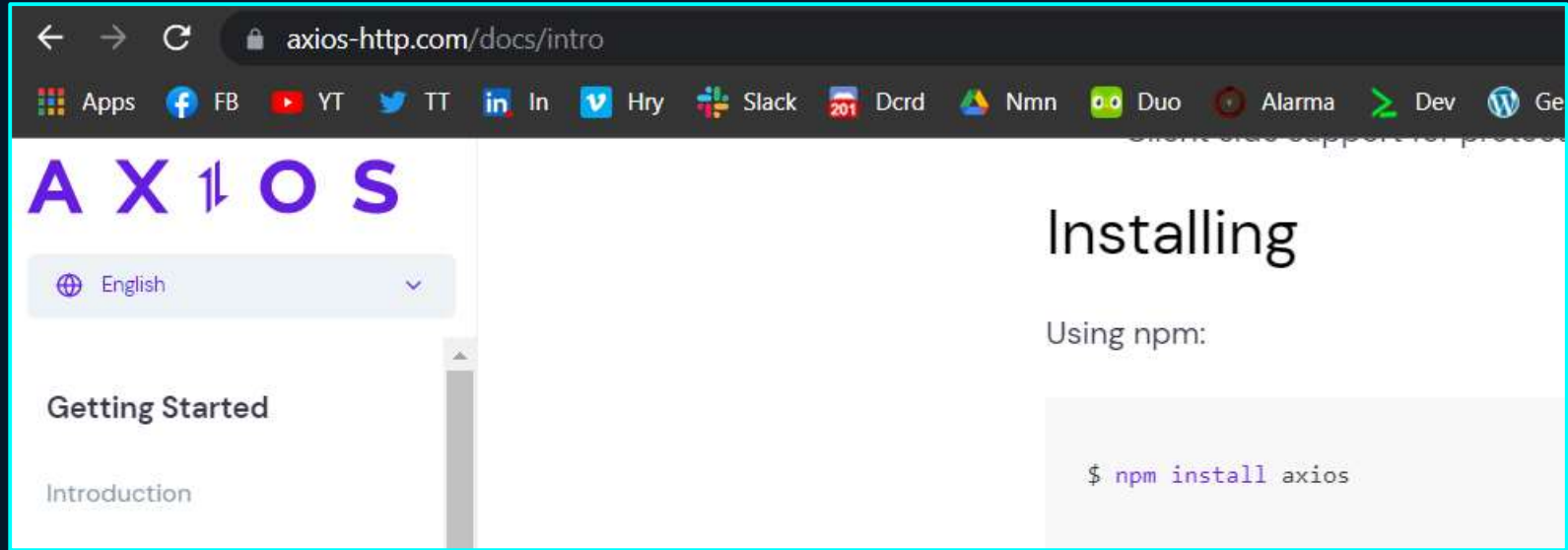
```
const CrudApp = () => {  
  const [db, setDb] = useState(initialDb);  
  const [dataToEdit, setDataToEdit] = useState(null);  
  
  const createData = (data) => {  
    data.id = Date.now()  
    // console.log(data)  
    setDb([...db, data])  
  };  
  
  const updateData = (data) => {  
    let newData = db.map(el => el.id === data.id ? data : el)  
    setDb(newData);  
  };  
  
  const deleteData = (id) => {  
    let isDelete = window.confirm(`Estás seguro de eliminar el registro con el id '${id}'`);  
  
    if (isDelete) {  
      let newData = db.filter(el => el.id !== id);  
      setDb(newData);  
    } else {  
      return  
    }  
  };  
};
```

Programamos la Eliminación.



Agregando Axios

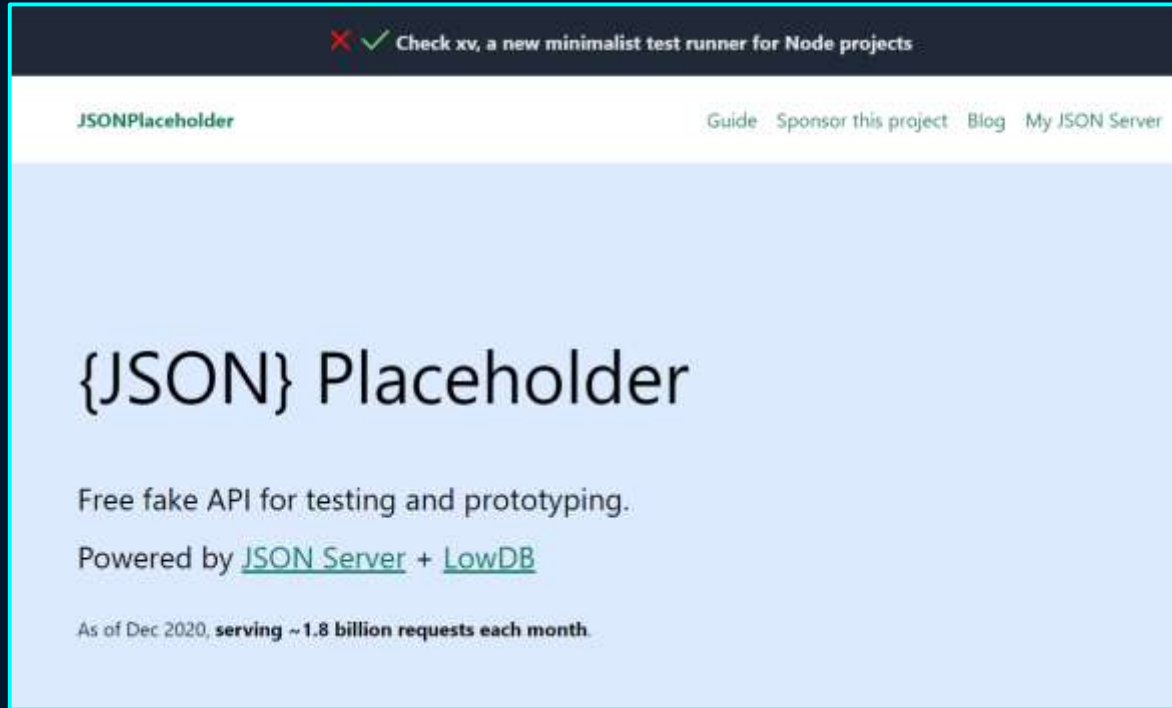
Instalando Axios



```
PS C:\Users\NicoD\OneDrive\Escritorio\jsx> npm install axios
```

```
import axios from "axios";
```

JSON Server



Enlace: <https://github.com/typicode/json-server>

Instalando JSON Server

Teniendo instalado Node, ejecutamos el siguiente comando para instalar JSON Server en nuestro proyecto.

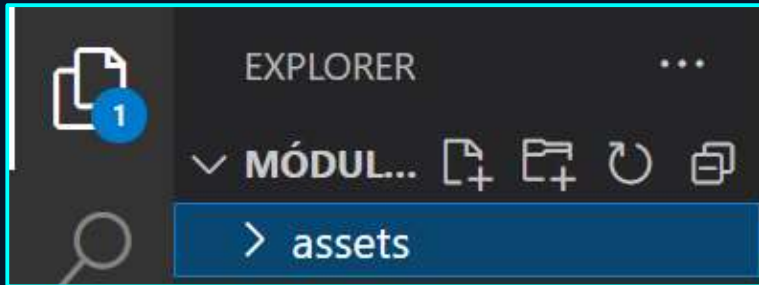
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

powershell + - [] [] ^ X

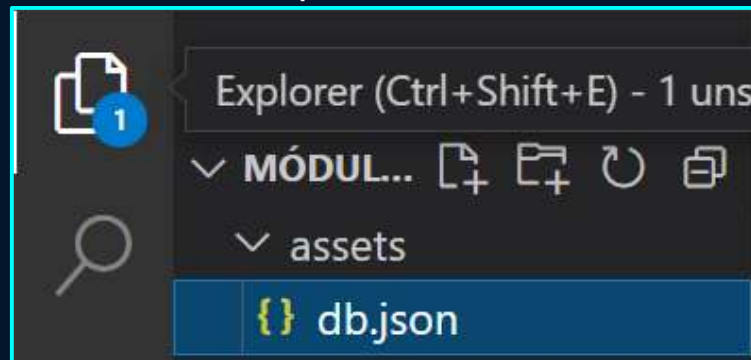
```
PS C:\Users\NicoD\OneDrive\Escritorio\Módulo Javascript> npm install -g json-server
```

Conectando nuestra base de datos con JSON Server

Paso 1: Creamos una carpeta llamada assets



Paso 2: Dentro de assets creamos un archivo json llamado db.json con la información que deseamos manipular.



Paso 3: Abrimos la terminal y escribimos el siguiente comando

```
PS C:\Users\NicoD\OneDrive\Escritorio\Módulo Javascript> json-server --watch db.json
```

El Arreglo de objeto que utilizaremos en db.json

```
{
  "santos": [
    {
      "id": 1,
      "name": "Seiya",
      "constellation": "Pegaso"
    },
    {
      "id": 2,
      "name": "Shiryu",
      "constellation": "Dragon"
    },
    {
      "id": 3,
      "name": "Shun",
      "constellation": "Andromeda"
    },
  ],
}
```

```
{
  {
    "id": 4,
    "name": "Ikki",
    "constellation": "Fenix"
  },
  {
    "id": 5,
    "name": "Hyoga",
    "constellation": "Cisne"
  }
]
}
```

Programamos la petición GET

```
const CrudApp = () => {  
  const [db, setDb] = useState(initialDB);  
  const [dataToEdit, setDataToEdit] = useState(null);  
  
  const getData = async () => {  
    const res = await  
    axios.get("http://localhost:3000/santos"),  
    json = await res.data  
    setDb(json);  
  }  
  useEffect(() => {  
    getData()  
  }, []);  
}
```

Programamos el
conditional render

```
{db && (  
  <CrudTable  
    data={db}  
    setDataToEdit={setDataToEdit}  
    deleteData={deleteData}  
  />  
)}
```

Creamos el
componente de Loader

```
const Loader = () => {  
  return (  
    <div>Cargando...</div>  
  )  
}  
export default Loader
```

```
{loading && <Loader />}  
{db && (  
  <CrudTable  
    data={db}  
    setDataToEdit={setDataToEdit}  
    deleteData={deleteData}  
  />  
)}  
))
```

Programamos el conditional render del Loader.

Programamos el useState del Loader

```
const CrudApp = () => {  
  
  const [db, setDb] = useState(initialDB);  
  const [dataToEdit, setDataToEdit] = useState(null);  
  const [loading, setLoading] = useState(false);  
  
  const getData = async () => {  
    const res = await axios.get("http://localhost:3000/santos"),  
    json = await res.data  
    setDb(json);  
  }  
  useEffect(() => {  
    setLoading(true);  
    getData();  
    setLoading(false);  
  }, []);  
}
```

Insertamos el
setLoading en el
useEffect

```
const createData = async (data) => {  
  data.id = Date.now();  
  
  let options = {  
    method: "POST",  
    headers: { "content-type": "application/json" },  
    data: JSON.stringify(data)  
  };  
  
  let res = await axios("http://localhost:3000/santos",  
options),  
  caballero = await res.data  
  setDb([...db, caballero]);  
};
```

Reemplazamos la programación del createData por una petición POST

Reemplazamos la programación del `updateData` por una petición PUT.

```
const updateData = async (data) => {  
  let endpoint = `http://localhost:5000/santos/${data.id}`;  
  
  let options = {  
    method: "PUT",  
    headers: { "content-type": "application/json" },  
    data: JSON.stringify(data)  
  },  
  res = await axios(endpoint, options),  
  caballero = await res.data  
  setDb([...db, caballero]);  
};
```

```
const deleteData = async (id) => {  
  let isDelete = window.confirm(  
    `Estás seguro que deseas eliminar el registro "${id}"?`  
  );  
  
  if (isDelete) {  
    let endpoint = `http://localhost:5000/santos/${id}`;  
  
    let options = {  
      method: "DELETE",  
      headers: { "content-type": "application/json" },  
    },  
    res = await axios(endpoint, options),  
    caballero = await res.data  
    setDb([...db, caballero]);  
  } else {  
    return;  
  }  
};
```

Reemplazamos la programación del updateData por una petición DELETE.