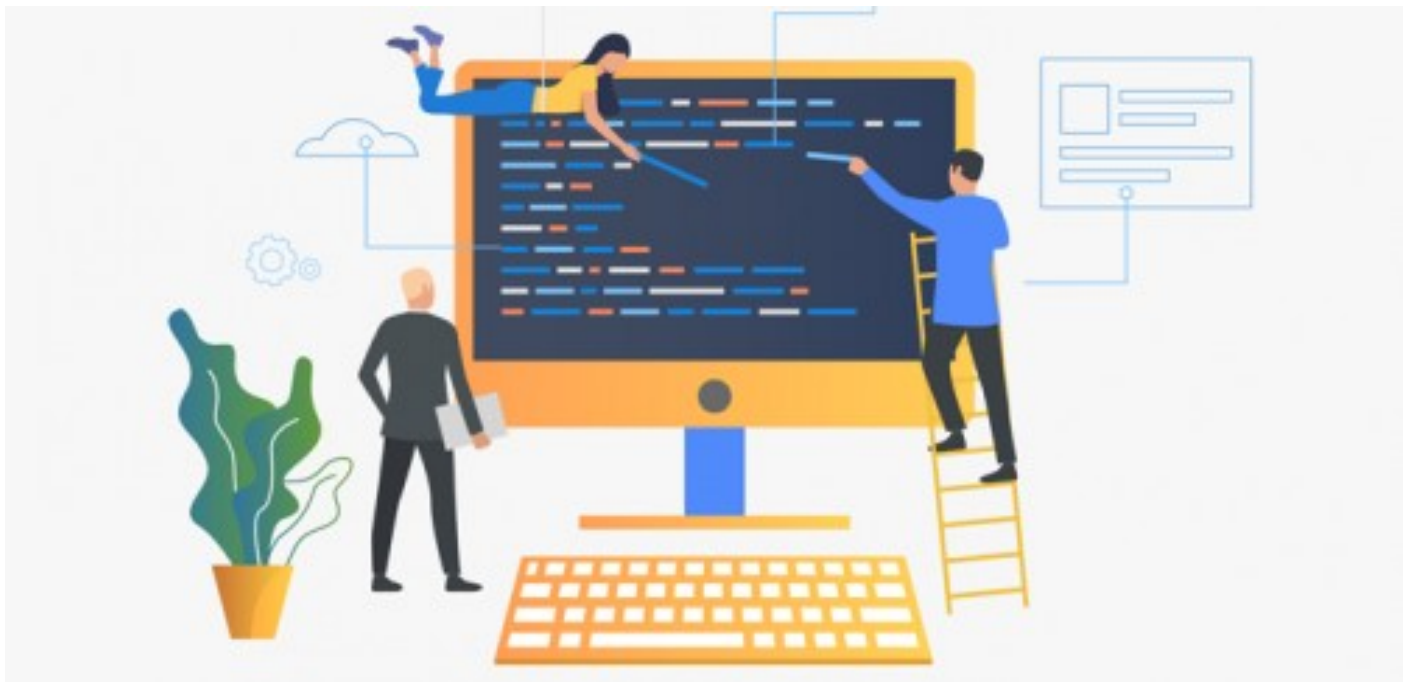


Tutorial CRUD usando Node JS, Express, React JS y MySQL (Full-Stack) (<https://mfikri.com/blog/node-express-react-mysql>)

📅 14 DE SEPTIEMBRE DE 2021 👤 M FIKRI SETIADI 📄 REACCIONARJS ([HTTPS://MFIKRI.COM/EN/CATEGORY/REACTJS](https://mfikri.com/en/category/reactjs)) 👁 23,045 VISTAS
🗨 5 COMENTARIOS



En este tutorial, aprenderá cómo crear una aplicación CRUD de pila completa utilizando node.js, express, MySQL y React JS.

No solo eso,

En este tutorial, también aprenderá a usar Bulma CSS para diseñar la interfaz.

Por lo tanto, la aplicación que se crea se vuelve más fácil de usar con una interfaz de usuario (UI) elegante y receptiva.

Este no es un tutorial para principiantes,

Si es un novato en node.js express, le sugiero que primero aprenda el " Tutorial de Express Js para principiantes (<https://mfikri.com/en/blog/express-tutorial-beginners>) ".

Y si eres nuevo en React JS, te sugiero que primero aprendas el " Tutorial de React Js para principiantes (<https://mfikri.com/en/blog/reactjs-beginners>) ".

Empecemos.

Tutorial LENGKAP CRUD dengan Node JS, Express, React JS, dan MySQL (Full Stack)



#1. Instalar: Express, MySQL2, Nodemon y Cors

Crea una carpeta en tu computadora, aquí la nombré " fullstack ".

Si crea una carpeta con el mismo nombre, eso es aún mejor.

Puede crearlo en cualquier lugar, ya sea en C, D o en el escritorio.

Luego abra la carpeta " fullstack " usando el editor de código, aquí uso Visual Studio Code .

También te sugiero que uses Visual Studio Code .

Puede descargar Visual Studio Code en el siguiente enlace y luego instalarlo en su computadora:

<https://code.visualstudio.com/> (<https://code.visualstudio.com/>)

Después de abrir la carpeta " fullstack " con Visual Studio Code, cree una subcarpeta llamada " backend " dentro de la carpeta " fullstack ".

A continuación, abra una terminal en Visual Studio Code en la barra de menú terminal => nueva terminal.

Luego, ve a la carpeta “ backend ” escribiendo el siguiente comando en la terminal:

```
1 | cd backend
```

Después de eso, escriba el siguiente comando en la terminal para crear un " paquete.json ": archivo

```
1 | npm init -y
```

A continuación, instale express , mysql2 , sequelize y cors escribiendo el siguiente comando en la terminal:

```
1 | npm install express mysql2 sequelize cors
```

A continuación, instale nodemon como una dependencia de desarrollo escribiendo el siguiente comando en la

A continuación, agregue el siguiente código al " paquete.json ": archivo

Entonces, el archivo " paquete.json " se parece a lo siguiente:

```
1 | {
2 |   {
3 |     "name": "backend",
4 |     "version": "1.0.0",
5 |     "description": "",
6 |     "type": "module",
7 |     "main": "index.js",
8 |     "scripts": {
9 |       "test": "echo \"Error: no test specified\" && exit 1"
10 |    },
11 |    "keywords": [],
12 |    "author": "",
13 |    "license": "ISC",
14 |    "dependencies": {
15 |      "cors": "^2.8.5",
16 |      "express": "^4.17.1",
17 |      "mysql2": "^2.3.0",
18 |      "sequelize": "^6.6.5"
19 |    },
20 |    "devDependencies": {
21 |      "nodemon": "^2.0.12"
22 |    }
23 |  }
```

Esto es para que podamos usar la sintaxis del módulo ES6 para exportar e importar módulos.

#2. Crear base de datos y tabla.

Para poder usar MySQL, debe instalar XAMPP, WAMP, MAMP o un software similar.

En este tutorial, estoy usando XAMPP (<https://www.apachefriends.org/index.html>) .

Luego cree una nueva base de datos en MySQL, puede usar herramientas como SQLyog, PHPMysqlAdmin o herramientas similares.

Aquí creo una base de datos con el nombre " mern_db ".

Si crea una base de datos con el mismo nombre, eso es aún mejor.

Para crear una base de datos en MySQL, se puede hacer ejecutando la siguiente consulta:

```
1 | CREATE DATABASE mern_db;
```

El comando SQL anterior creará una base de datos con el nombre " mern_db ".

A continuación, cree una tabla en la base de datos " mern_db ".

Aquí creo una tabla con el nombre " productos ".

Si crea una tabla con el mismo nombre, eso es aún mejor.

Para crear una tabla de " productos ", se puede hacer ejecutando el siguiente comando SQL:

```
1 | CREATE TABLE products(  
3 | id INT(11) PRIMARY KEY AUTO_INCREMENT,  
4 | title VARCHAR(200),  
5 | price DOUBLE,  
6 | createdAt DATE,  
7 | updatedAt DATE  
  | )ENGINE=INNODB;
```

El comando SQL anterior creará una tabla llamada "productos" con campos: id , título , precio , createdAt y updatedAt .

Sequelize requiere que tenga campos: id , createdAt y updatedAt .

Si no tiene ese campo, habrá un error.

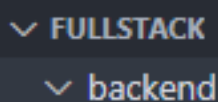
#3. Estructura de la aplicación

Para que la aplicación esté más estructurada, aplicaremos el patrón MVC (Model-View-Controllers).

Cree las carpetas " config ", " controllers ", " models " y " routes " en la carpeta " backend ".

Luego cree un " base archivo de datos.js " en la carpeta " config ", cree un archivo " Productos.js " en la carpeta " controladores ", cree un archivo " productModel.js " en la carpeta " modelos ", cree un " índice .js " en la carpeta " routes ", y cree el archivo " index.js " en la carpeta " backend ".

Mira la siguiente imagen para más detalles:



✓ FULLSTACK

✓ backend

✓ config

JS database.js

✓ controllers

JS Products.js

✓ models

JS productModel.js

> node_modules

✓ routes

JS index.js

JS index.js

{ } package-lock.json

{ } package.json

#4. Conectar a la base de datos

Abra el base archivo " de datos.js" en la carpeta " config ", luego escriba el siguiente código:

```
1 | import { Sequelize } from "sequelize";
2 |
3 |
4 | const db = new Sequelize('mern_db', 'root', '', {
5 |   host: "localhost",
6 |   dialect: "mysql"
7 | });
8 |
9 | export default db;
```

#5. Modelos

Abra el archivo del modelo " productModel.js " que se encuentra en la carpeta " modelos ", luego escriba el siguiente código:

```
1 |
2 | import { Sequelize } from "sequelize";
3 | import db from "../config/database.js";
4 |
5 | const { DataTypes } = Sequelize;
6 |
7 | const Product = db.define('products',{
8 |   title:{
9 |     type: DataTypes.STRING
10 |   },
11 |   price:{
12 |     type: DataTypes.DOUBLE
13 |   }
14 | },{
15 |   freezeTableName: true
16 | });
17 | export default Product;
```

Abra el archivo del controlador " Productos.js " que se encuentra en la carpeta " controladores ", luego escriba el

```

1  import Product from "../models/productModel.js";
2
3
4  export const getAllProducts = async (req, res) => {
5      try {
6          const products = await Product.findAll();
7          res.json(products);
8      } catch (error) {
9          res.json({ message: error.message });
10     }
11 }
12
13 export const getProductById = async (req, res) => {
14     try {
15         const product = await Product.findAll({
16             where: {
17                 id: req.params.id
18             }
19         });
20         res.json(product[0]);
21     } catch (error) {
22         res.json({ message: error.message });
23     }
24 }
25
26 export const createProduct = async (req, res) => {
27     try {
28         await Product.create(req.body);
29         res.json({
30             "message": "Product Created"
31         });
32     } catch (error) {
33         res.json({ message: error.message });
34     }
35 }
36
37 export const updateProduct = async (req, res) => {
38     try {
39         await Product.update(req.body, {
40             where: {
41                 id: req.params.id
42             }
43         });
44         res.json({
45             "message": "Product Updated"
46         });
47     } catch (error) {
48         res.json({ message: error.message });
49     }
50 }
51
52 export const deleteProduct = async (req, res) => {
53     try {
54         await Product.destroy({
55             where: {
56                 id: req.params.id
57             }
58         });
59         res.json({
60             "message": "Product Deleted"
61         });
62     } catch (error) {
63         res.json({ message: error.message });
64     }
65 }

```

#7. Rutas

Abra el archivo “ index.js ” ubicado en la carpeta “ routes ”, luego escriba el siguiente código:

```

1 |
2 | import express from "express";
3 |
4 | import {
5 |     getAllProducts,
6 |     createProduct,
7 |     getProductById,
8 |     updateProduct,
9 |     deleteProduct
10 | } from "../controllers/Products.js";
11 |
12 | const router = express.Router();
13 |
14 | router.get('/', getAllProducts);
15 | router.get('/:id', getProductById);
16 | router.post('/', createProduct);
17 | router.patch('/:id', updateProduct);
18 | router.delete('/:id', deleteProduct);
19 |
    export default router;

```

#8. Punto de entrada

Abra el archivo “ index.js ” ubicado en la carpeta “ backend ”, luego escriba el siguiente código:

```

2 | import express from "express";
3 | import db from "../config/database.js";
4 | import productRoutes from "../routes/index.js";
5 | import cors from "cors";
6 |
7 | const app = express();
8 |
9 | try {
10 |     await db.authenticate();
11 |     console.log('Database connected...');
12 | } catch (error) {
13 |     console.error('Connection error:', error);
14 | }
15 |
16 | app.use(cors());
17 | app.use(express.json());
18 | app.use('/products', productRoutes);
19 |
    app.listen(5000, () => console.log('Server running at port 5000'));

```

Para asegurarse de que la aplicación se ejecute correctamente, ejecútela escribiendo el siguiente comando en la terminal:

```

1 | |
    nodemon index

```

Si va bien, se verá como la siguiente imagen:

The screenshot shows a terminal window with the following content:

```

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

M Fikri@DESKTOP-MQCEKPJ MINGW64 ~/Documents/Tutorials/fullstack/backend
$ nodemon index
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node index index.js`
Executing (default): SELECT 1+1 AS result
Database connected...
Server running at port 5000

```

Hasta aquí ha creado con éxito un " backend ".

Para asegurarse de que el backend funcione correctamente, puede usar POSTMAN (<https://www.postman.com/downloads/>) o las extensiones de cliente REST en VS Code para realizar algunas pruebas.

#9. Interfaz

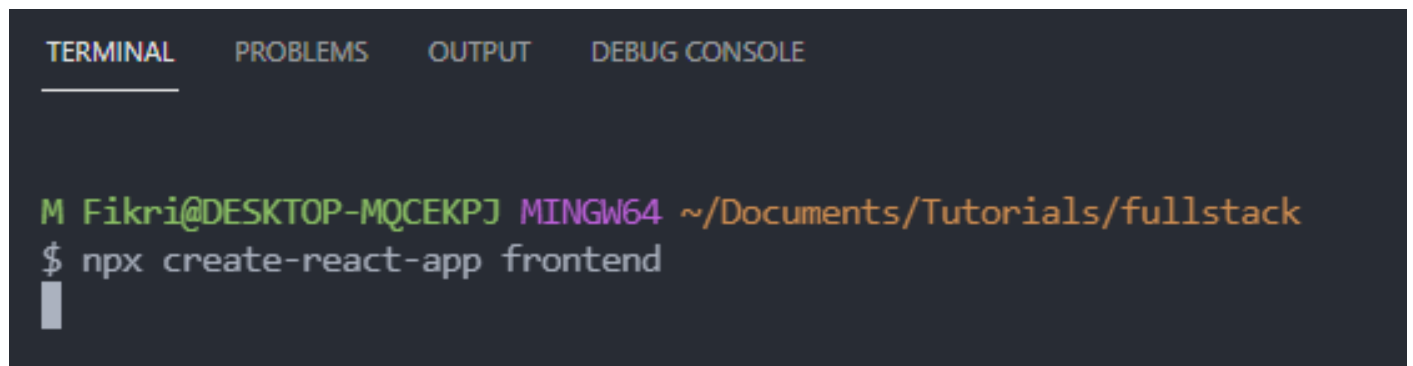
Si no está familiarizado con React JS, le sugiero que primero aprenda el " Tutorial de React JS para principiantes (<https://medium.com/en/blog/reactjs-beginners>) ".

Hay muchas maneras de crear un proyecto React. Sin embargo, la forma más fácil es " crear una aplicación de reacción ".

Abra una nueva terminal y cree un nuevo proyecto de reacción escribiendo el siguiente comando en la terminal:

```
1 | npx create-react-app frontend
```

Y asegúrese de estar en la carpeta " fullstack ", como en la siguiente imagen:

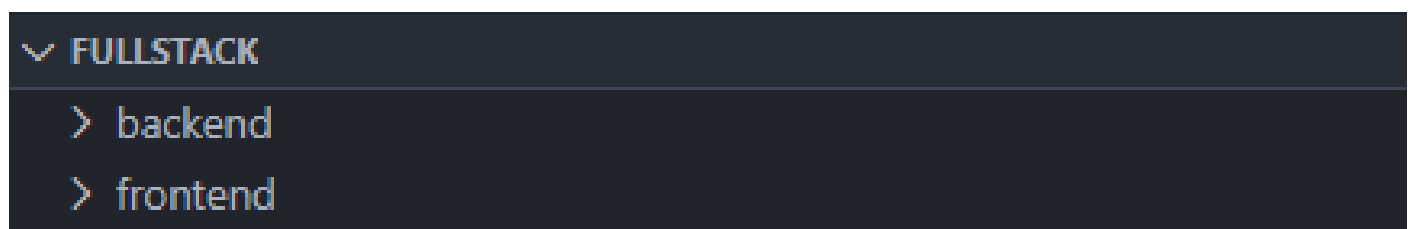


```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

M Fikri@DESKTOP-MQCEKPJ MINGW64 ~/Documents/Tutorials/fullstack
$ npx create-react-app frontend
```

Si la instalación está completa, habrá una carpeta " frontend " en la carpeta " fullstack ".

De modo que en la carpeta " fullstack " hay dos carpetas, a saber: " backend " y " frontend " como se muestra a continuación:



```
▼ FULLSTACK
  > backend
  > frontend
```

✓ FULLSTACK

> backend

> frontend

La carpeta " backend " es la carpeta de la aplicación que se creó previamente con node.js express, mientras que la " frontend " es la carpeta de la aplicación que se creó con React JS.

A continuación, vaya a la carpeta " frontend " escribiendo el siguiente comando en la terminal:

```
1 | cd frontend
```

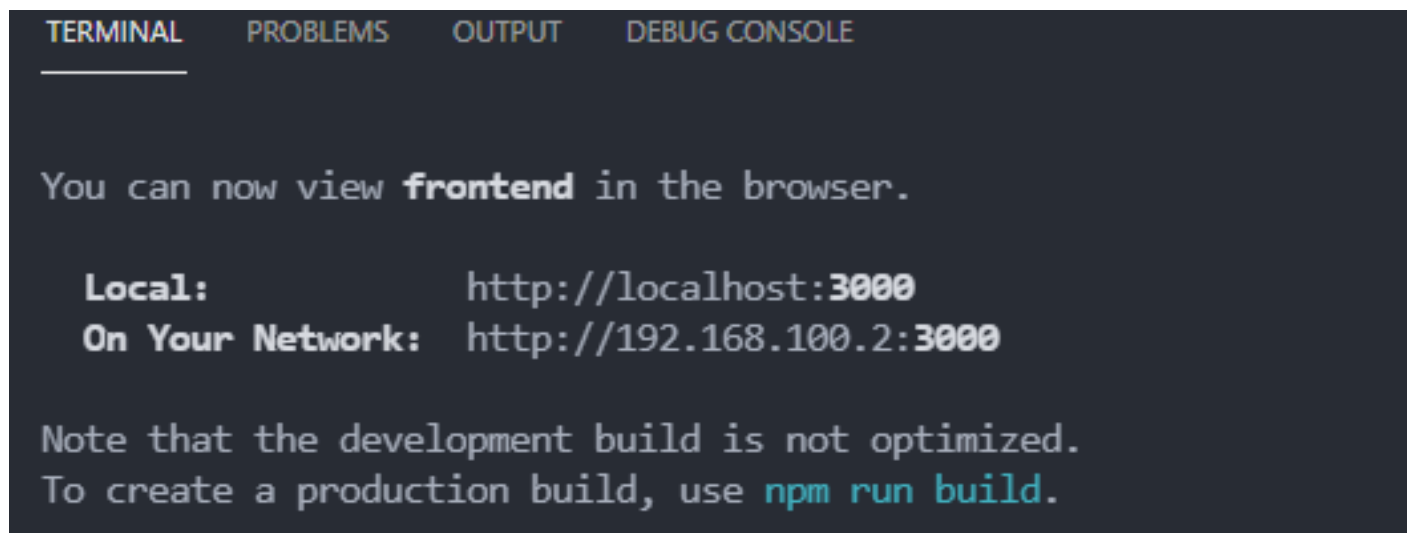
Después de eso, instala react-router-dom , axios y bulma escribiendo el siguiente comando en la terminal:

```
npm install react-router-dom axios bulma
```

Una vez completada la instalación, y para asegurarse de que todo vaya bien, escriba el siguiente comando para ejecutar el proyecto:

```
1 | npm start
```

Si va bien, se verá como la siguiente imagen:



Luego, abra su navegador y visite la siguiente URL:

```
1 | http://localhost:3000 (http://localhost:3000)
```

Entonces aparecerá de la siguiente manera:





Edit `src/App.js` and save to reload.

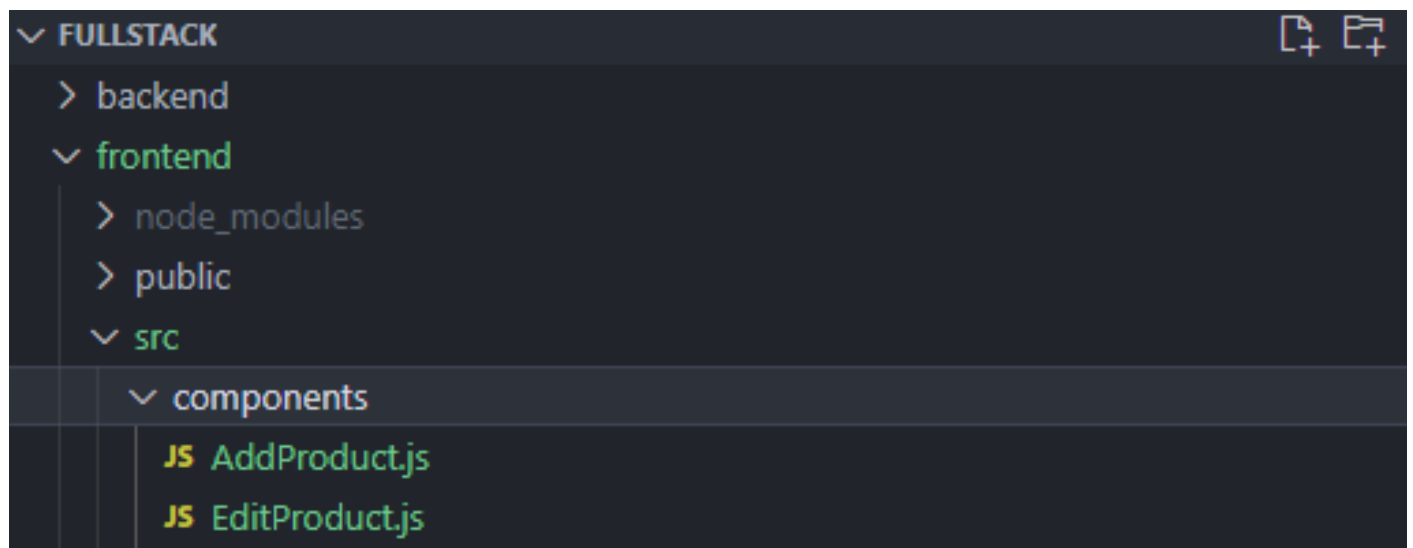
[Learn React](#)

#10. Componentes

Cree una carpeta llamada " componentes " dentro de la carpeta " frontend/src " .

Luego, cree los archivos de componentes " ProductList.js " , " AddProduct.js " y " EditProduct.js " en la carpeta " frontend/src/components " .

Como la siguiente imagen:



▼ components

JS AddProduct.js

JS EditProduct.js

JS ProductList.js

App.css

JS App.js

Luego abra el archivo " ProductList.js ", luego escriba el siguiente código:

```
1
2 import { useState, useEffect } from 'react'
3 import axios from "axios";
4 import { Link } from "react-router-dom";
5
6 const ProductList = () => {
7   const [products, setProduct] = useState([]);
8
9   useEffect(() => {
10     getProducts();
11   }, []);
12
13   const getProducts = async () => {
14     const response = await axios.get('http://localhost:5000/products (http://localhost:5000/products)');
15     setProduct(response.data);
16   }
17
18   const deleteProduct = async (id) => {
19     await axios.delete(`http://localhost:5000/products/${id}`);
20     getProducts();
21   }
22
23   return (
24     <div>
25       <Link to="/add" className="button is-primary mt-2">Add New</Link>
26       <table className="table is-striped is-fullwidth">
27         <thead>
28           <tr>
29             <th>No</th>
30             <th>Title</th>
31             <th>Price</th>
32             <th>Actions</th>
33           </tr>
34         </thead>
35         <tbody>
36           { products.map((product, index) => (
37             <tr key={ product.id }>
38               <td>{ index + 1 }</td>
39               <td>{ product.title }</td>
40               <td>{ product.price }</td>
41               <td>
42                 <Link to={`/edit/${product.id}`} className="button is-small is-info">Edit</Link>
43                 <button onClick={ () => deleteProduct(product.id) } className="button is-small is-danger"
44               </td>
45             </tr>
46           )) }
47         </tbody>
48       </table>
49     </div>
50   )
51 }
52
53 export default ProductList
```

A continuación, abra el archivo " AddProduct.js ", luego escriba el siguiente código:

```

1  import { useState } from 'react'
2  import axios from "axios";
3  import { useHistory } from 'react-router-dom';
4
5  const AddProduct = () => {
6    const [title, setTitle] = useState('');
7    const [price, setPrice] = useState('');
8    const history = useHistory();
9
10
11    const saveProduct = async (e) => {
12      e.preventDefault();
13      await axios.post('http://localhost:5000/products (http://localhost:5000/products)', {
14        title: title,
15        price: price
16      });
17      history.push("/");
18    }
19
20    return (
21      <div>
22        <form onSubmit={ saveProduct }>
23          <div className="field">
24            <label className="label">Title</label>
25            <input
26              className="input"
27              type="text"
28              placeholder="Title"
29              value={ title }
30              onChange={ (e) => setTitle(e.target.value) }
31            />
32          </div>
33
34          <div className="field">
35            <label className="label">Price</label>
36            <input
37              className="input"
38              type="text"
39              placeholder="Price"
40              value={ price }
41              onChange={ (e) => setPrice(e.target.value) }
42            />
43          </div>
44
45          <div className="field">
46            <button className="button is-primary">Save</button>
47          </div>
48        </form>
49      </div>
50    )
51  }
52  export default AddProduct

```

A continuación, abra el archivo " EditProduct.js ", luego escriba el siguiente código:

```

1  /* eslint-disable react-hooks/exhaustive-deps */
2  import { useState, useEffect } from 'react'
3  import axios from "axios";
4  import { useHistory, useParams } from 'react-router-dom';
5
6  const EditProduct = () => {
7    const [title, setTitle] = useState('');
8    const [price, setPrice] = useState('');
9    const history = useHistory();
10   const { id } = useParams();
11
12   const updateProduct = async (e) => {
13     e.preventDefault();
14     await axios.patch(`http://localhost:5000/products/${id}`, {
15       title: title,
16       price: price
17     });
18     history.push("/");
19   }
20
21   useEffect(() => {
22     getProductById();
23   }, []);
24
25   const getProductById = async () => {
26     const response = await axios.get(`http://localhost:5000/products/${id}`);
27     setTitle(response.data.title);
28     setPrice(response.data.price);
29   }
30
31   return (
32     <div>
33       <form onSubmit={ updateProduct }>
34         <div className="field">
35           <label className="label">Title</label>
36           <input
37             className="input"
38             type="text"
39             placeholder="Title"
40             value={ title }
41             onChange={ (e) => setTitle(e.target.value) }
42           />
43         </div>
44         <div className="field">
45           <label className="label">Price</label>
46           <input
47             className="input"
48             type="text"
49             placeholder="Price"
50             value={ price }
51             onChange={ (e) => setPrice(e.target.value) }
52           />
53         </div>
54         <div className="field">
55           <button className="button is-primary">Update</button>
56         </div>
57       </form>
58     </div>
59   )
60 }
61
62 export default EditProduct
63
64

```

#11 Aplicación.js

Abra el archivo " App.js " en la carpeta " frontend/src ", luego cámbielo a lo siguiente:

```

1  import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
2  import ProductList from "../components/ProductList";
3  import AddProduct from "../components/AddProduct";
4  import EditProduct from "../components/EditProduct";
5
6
7  function App() {
8    return (
9      <Router>
10     <div className="container">
11       <div className="columns">
12         <div className="column is-half is-offset-one-quarter">
13           <Switch>
14             <Route exact path="/">
15               <ProductList />
16             </Route>
17             <Route path="/add">
18               <AddProduct />
19             </Route>
20             <Route path="/edit/:id">
21               <EditProduct />
22             </Route>
23           </Switch>
24         </div>
25       </div>
26     </div>
27   </Router>
28 );
29 }
30 export default App;

```

#12. Pruebas

Regrese al navegador y visite la siguiente URL:

1 | <http://localhost:3000> (<http://localhost:3000>)

Si va bien, se verá como la siguiente imagen:

Add New			
No	Title	Price	Actions
1	Product 1	873	Edit Delete
2	Product 2	993	Edit Delete
3	Product 3	979	Edit Delete
4	Product 4	165	Edit Delete
5	Product 5	872	Edit Delete

Haga clic en el botón " Agregar nuevo ", aparecerá un formulario como el siguiente:

Title

Product New

Price

982

Save

Ingrese " Título " y " Precio ", luego haga clic en el botón " GUARDAR ".

Si la inserción es exitosa, verá la adición de datos a la lista de productos como se muestra a continuación:

Add New

No	Title	Price	Actions	
1	Product 1	873	Edit	Delete
2	Product 2	993	Edit	Delete
3	Product 3	979	Edit	Delete
4	Product 4	165	Edit	Delete
5	Product 5	872	Edit	Delete
6	Product New	982	Edit	Delete

Para actualizar los datos, haga clic en uno de los " Editar ", aparecerá un formulario como el siguiente: botones

Title

Product Updated

Price

731

Update

Cambie el " Título " o el " Precio ", luego haga clic en el botón " ACTUALIZAR ".

Si la actualización es exitosa, verá cambios en los datos en la lista de productos como se muestra a continuación:

Add New

No	Title	Price	Actions	
1	Product 1	873	Edit	Delete
2	Product 2	993	Edit	Delete
3	Product 3	979	Edit	Delete
4	Product 4	165	Edit	Delete
5	Product 5	872	Edit	Delete
6	Product Updated	731	Edit	Delete

Para eliminar datos, haga clic en uno de los " Eliminar ". botones

Si la eliminación es exitosa, los datos se perderán de la lista de productos.

Conclusión:

La discusión esta vez es cómo crear una aplicación de pila completa con un node.js express en el backend y un frontend usando React JS.

De esa manera, tendrá una idea de cómo crear una aplicación web moderna que separe el backend y el frontend.

Entonces, ¿qué estás esperando? ¡Vamos a codificar!

📄 DESCARGAR CÓDIGO FUENTE (HTTPS://PASSIONEAGLE.COM/DL/MTYZ)

BUSCAR

88

ÚLTIMOS ARTÍCULOS



(<https://mfikri.com/en/blog/express-mysql-react>) Tutorial CRUD usando Node JS, Express, React JS y MongoDB (MERN Stack) (<https://mfikri.com/en/blog/express-mysql-react>)
M Fikri Setiadi 21 de septiembre



(<https://mfikri.com/en/blog/express-reactjs-mysql>) Tutorial CRUD usando Node JS, Express, React JS y MySQL (ACTUALIZADO) (<https://mfikri.com/en/blog/express-reactjs-mysql>)
M Fikri Setiadi 09 de septiembre



(<https://mfikri.com/en/blog/react-express-mysql-authentication>) Regístrese e inicie sesión con autenticación JWT (React JS + Express + MySQL + Node JS) (<https://mfikri.com/en/blog/react-express-mysql-authentication>)
M Fikri Setiadi 23 de diciembre



(<https://mfikri.com/en/blog/react-crud-codeigniter>) Tutorial completo de CRUD CodeIgniter 4 y React JS (Full-Stack) (<https://mfikri.com/en/blog/react-crud-codeigniter>)
M Fikri Setiadi 09 de octubre



(<https://mfikri.com/en/blog/reactjs-beginners>) Tutorial de React JS para principiantes (React Hooks) (<https://mfikri.com/en/blog/reactjs-beginners>)
M Fikri Setiadi 09 de septiembre



(<https://mfikri.com/en/blog/codeigniter-login-jwt>) CodeIgniter 4 Inicie sesión y regístrese con JWT (JSON Web Token) (<https://mfikri.com/en/blog/codeigniter-login-jwt>)
M Fikri Setiadi 01 de septiembre