

Luisa Fernanda Triviño Gil

CC: 1030612775

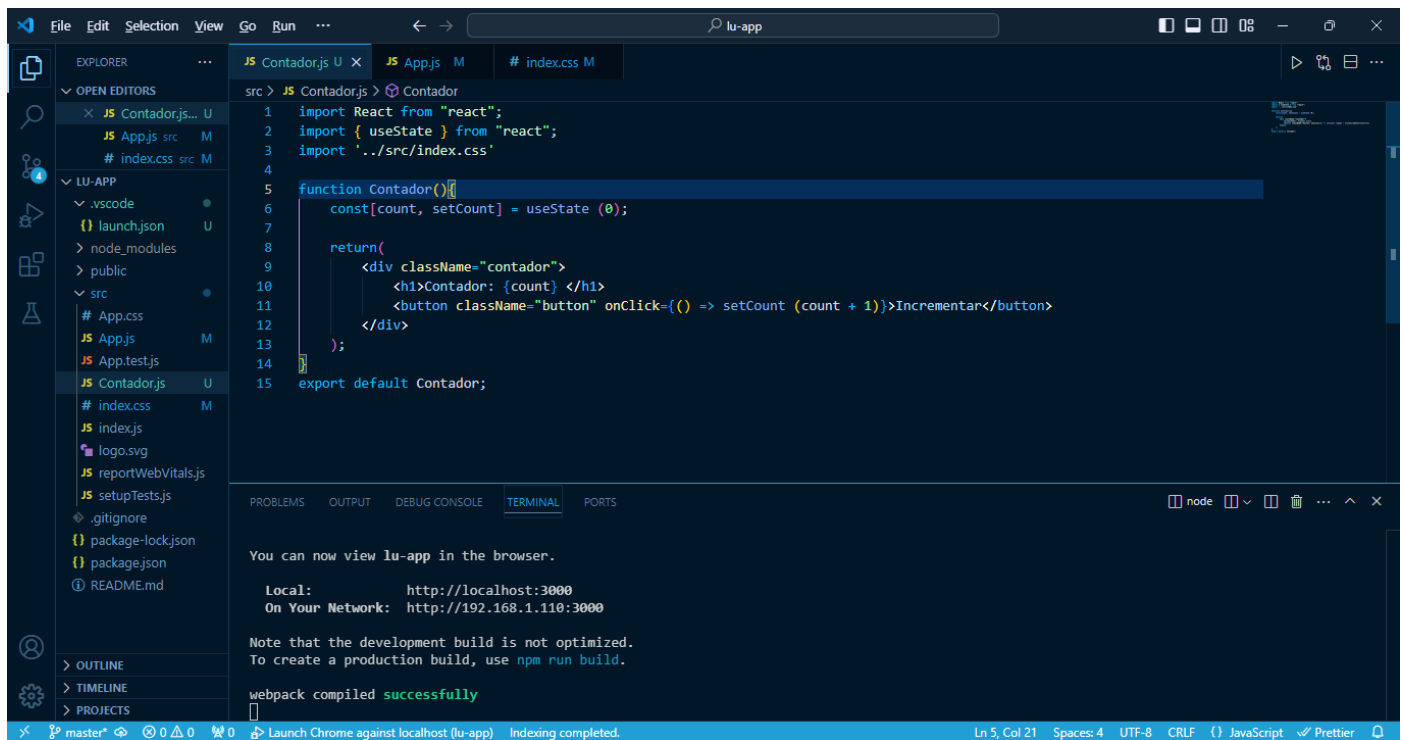
Estudiante Sena ADSO

TALLER 3 REACT

Ejercicio 1: Contador simple Crea un componente que muestre un contador. Al hacer clic en un botón, el contador debe incrementarse.

Objetivo: Que los estudiantes entiendan cómo funciona `this.state` y el método `setState` para actualizar el estado.

Contador.js



The screenshot shows a Visual Studio Code editor with a project named 'lu-app'. The Explorer sidebar on the left shows the file structure, including 'src' and 'public' folders. The main editor window displays the 'Contador.js' file with the following code:

```
1 import React from "react";
2 import { useState } from "react";
3 import '../src/index.css'
4
5 function Contador() {
6   const [count, setCount] = useState(0);
7
8   return (
9     <div className="contador">
10       <h1>Contador: {count}</h1>
11       <button className="button" onClick={() => setCount(count + 1)}>Incrementar</button>
12     </div>
13   );
14 }
15 export default Contador;
```

Below the code editor, the TERMINAL panel shows the following output:

```
You can now view lu-app in the browser.

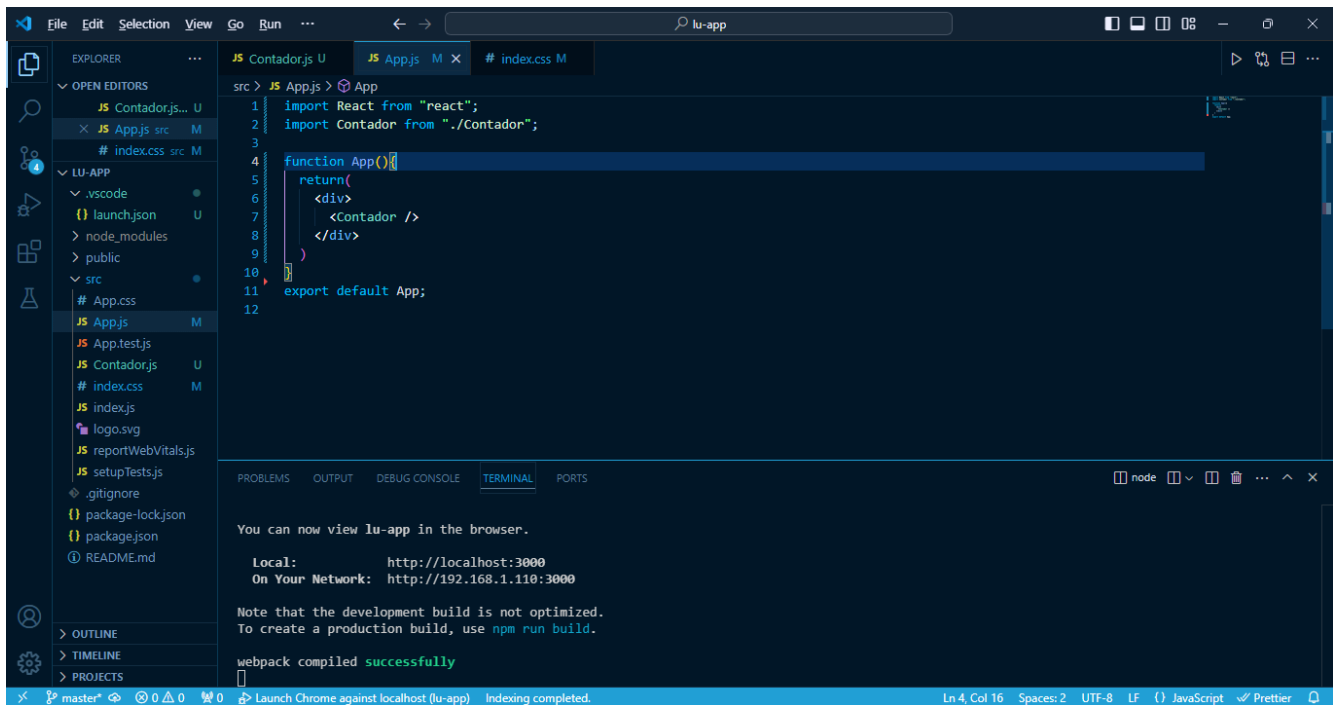
Local:      http://localhost:3000
On Your Network: http://192.168.1.110:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

The status bar at the bottom indicates the file is 'Ln 5, Col 21', uses 'UTF-8' encoding, 'CRLF' line endings, and is 'JavaScript' with 'Prettier' formatting.

App.js



The screenshot shows the VS Code editor with the file explorer on the left. The 'src' folder is expanded, showing files like App.js, App.test.js, Contador.js, index.css, index.js, logo.svg, reportWebVitals.js, and setupTests.js. The 'App.js' file is open in the editor, showing the following code:

```
1 import React from "react";
2 import Contador from "../Contador";
3
4 function App() {
5   return (
6     <div>
7       <Contador />
8     </div>
9   );
10 }
11 export default App;
```

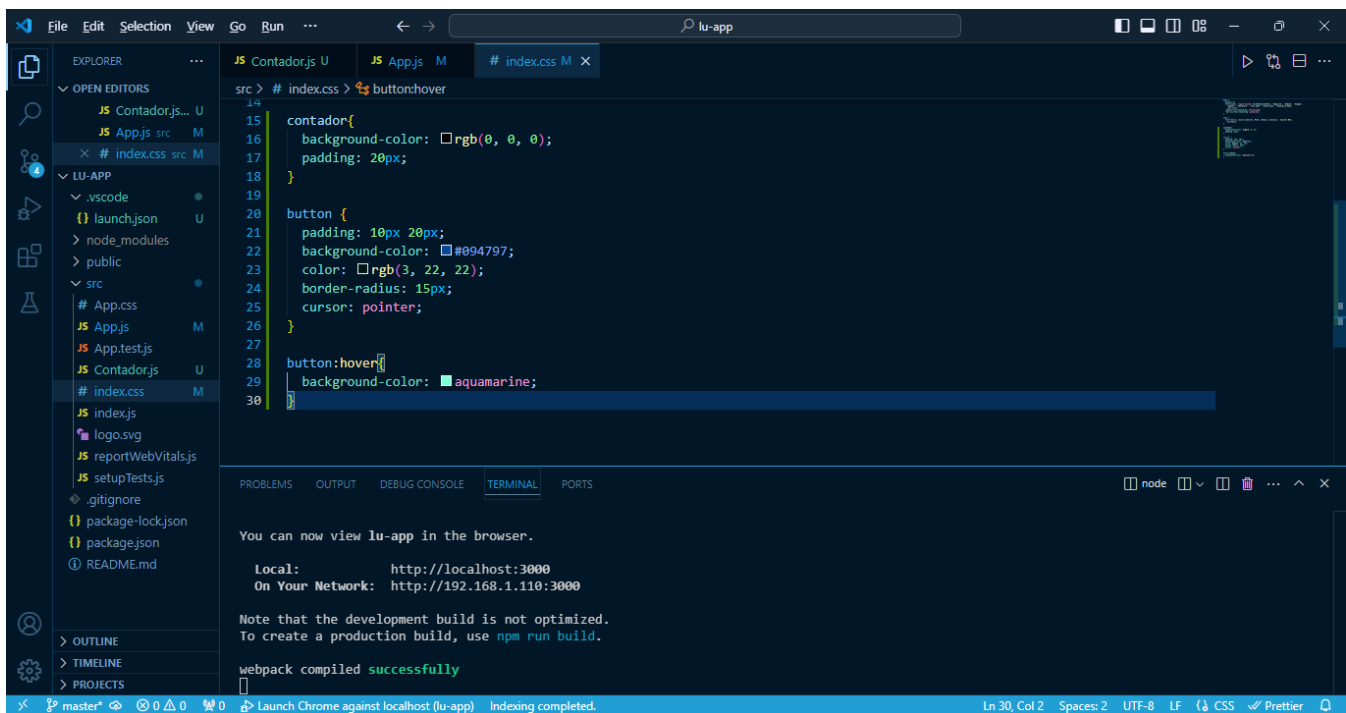
The terminal at the bottom shows the following output:

```
You can now view lu-app in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.1.110:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Index.css



The screenshot shows the VS Code editor with the file explorer on the left. The 'src' folder is expanded, showing files like App.js, App.test.js, Contador.js, index.css, index.js, logo.svg, reportWebVitals.js, and setupTests.js. The 'index.css' file is open in the editor, showing the following code:

```
14
15 contador{
16   background-color: #rgb(0, 0, 0);
17   padding: 20px;
18 }
19
20 button {
21   padding: 10px 20px;
22   background-color: #094797;
23   color: #rgb(3, 22, 22);
24   border-radius: 15px;
25   cursor: pointer;
26 }
27
28 button:hover{
29   background-color: #aquamarine;
30 }
```

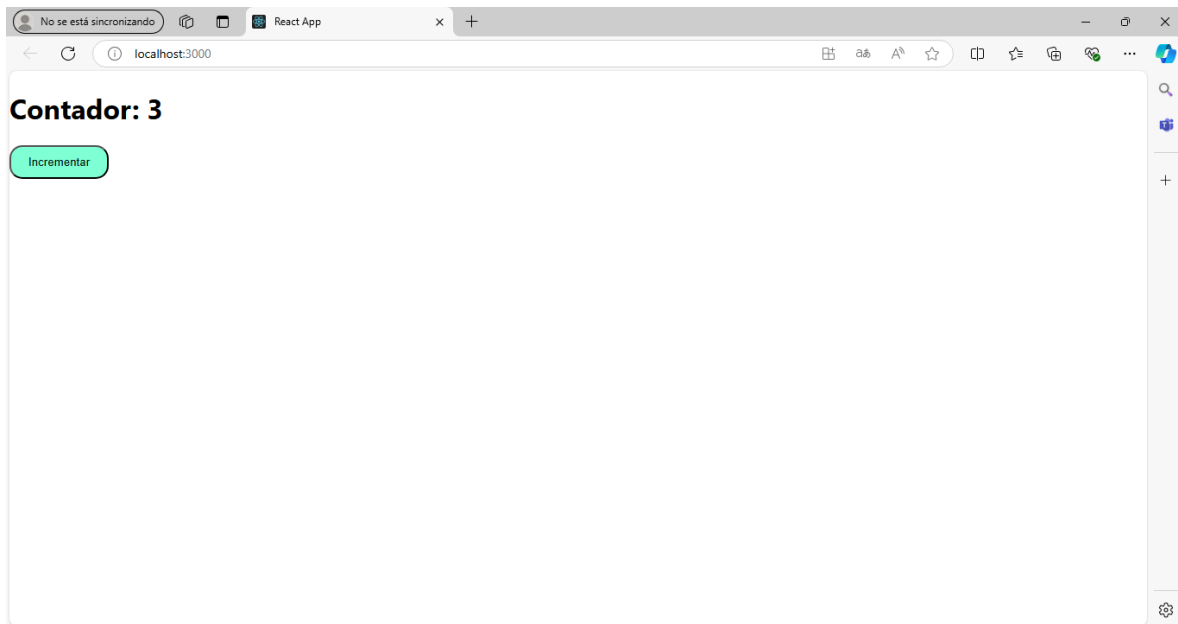
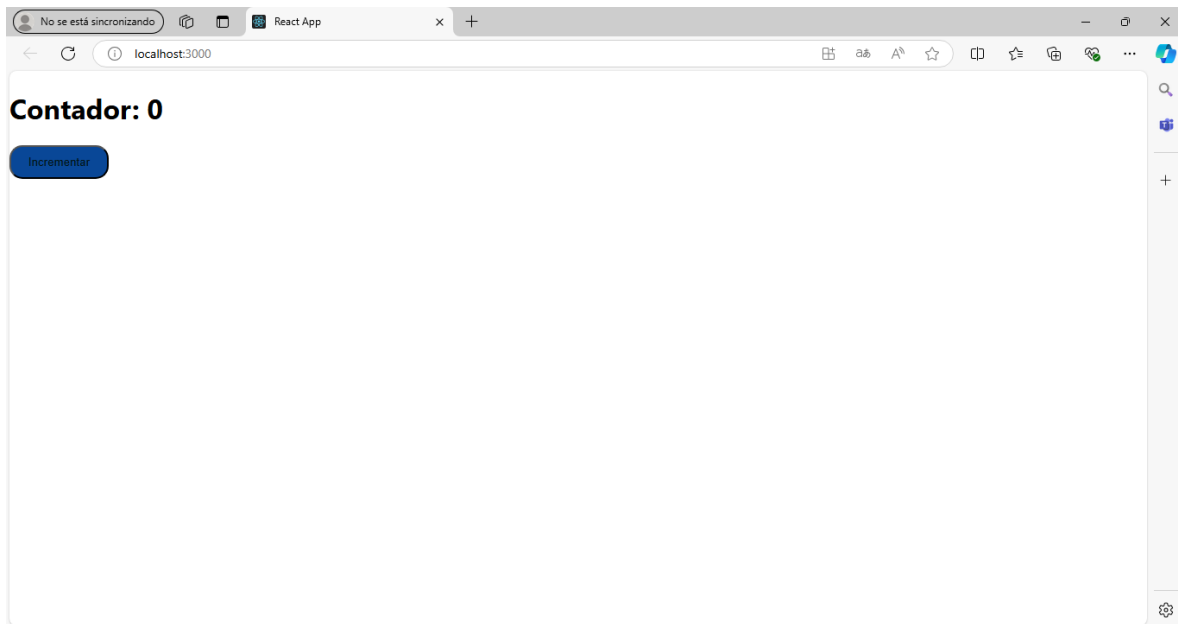
The terminal at the bottom shows the following output:

```
You can now view lu-app in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.1.110:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Resultado:



Ejercicio 2: Ciclo de vida de un componente. Crea un componente que muestre un mensaje en la consola cuando se monte, actualice y desmonte.

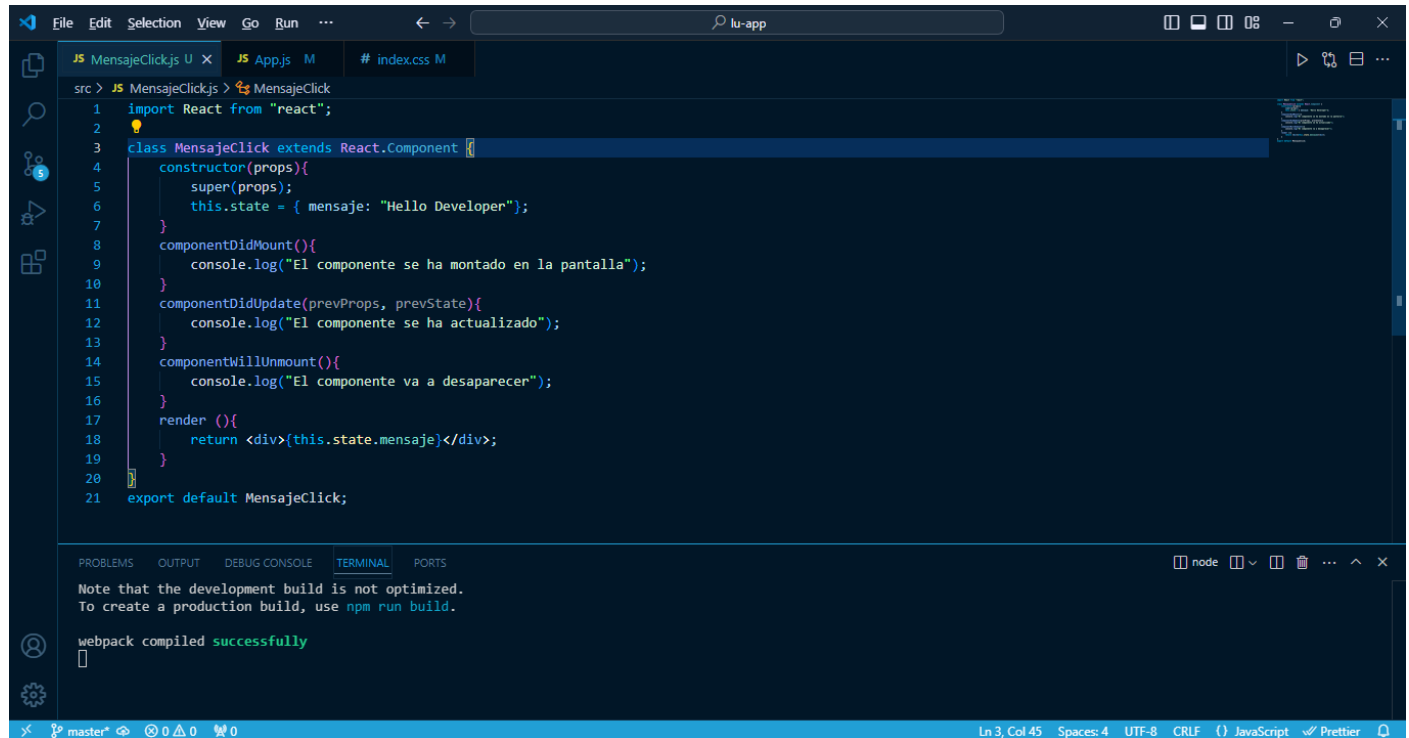
Objetivo: Que los estudiantes practiquen los métodos del ciclo de vida y vean cómo responden a cambios de estado.

Instrucciones:

1. Realiza ambos ejercicios en tu entorno de desarrollo. Intenta modificar el código para explorar cómo funcionan `setState` y los métodos del ciclo de vida.

2. Observa los mensajes en la consola cuando el componente se monta, actualiza y se desmonta.
3. Opcional: Crea un tercer componente que combine lo aprendido sobre setState y ciclo de vida (por ejemplo, un componente que cambie un valor cuando se actualice)

MensajeClick.js



The image shows a Visual Studio Code editor window with a dark theme. The editor is open to a file named `MensajeClick.js` in the `src` directory. The code defines a class `MensajeClick` that extends `React.Component`. It includes a constructor, lifecycle methods (`componentDidMount`, `componentDidUpdate`, `componentWillUnmount`), and a `render` method. The `render` method returns a `<div>` element containing the state message. The terminal at the bottom shows the output of the development server, indicating that webpack compiled successfully.

```
1 import React from "react";
2
3 class MensajeClick extends React.Component {
4   constructor(props) {
5     super(props);
6     this.state = { mensaje: "Hello Developer" };
7   }
8   componentDidMount() {
9     console.log("El componente se ha montado en la pantalla");
10  }
11   componentDidUpdate(prevProps, prevState) {
12     console.log("El componente se ha actualizado");
13   }
14   componentWillUnmount() {
15     console.log("El componente va a desaparecer");
16   }
17   render() {
18     return <div>{this.state.mensaje}</div>;
19   }
20 }
21 export default MensajeClick;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Note that the development build is not optimized.
To create a production build, use `npm run build`.

webpack compiled successfully

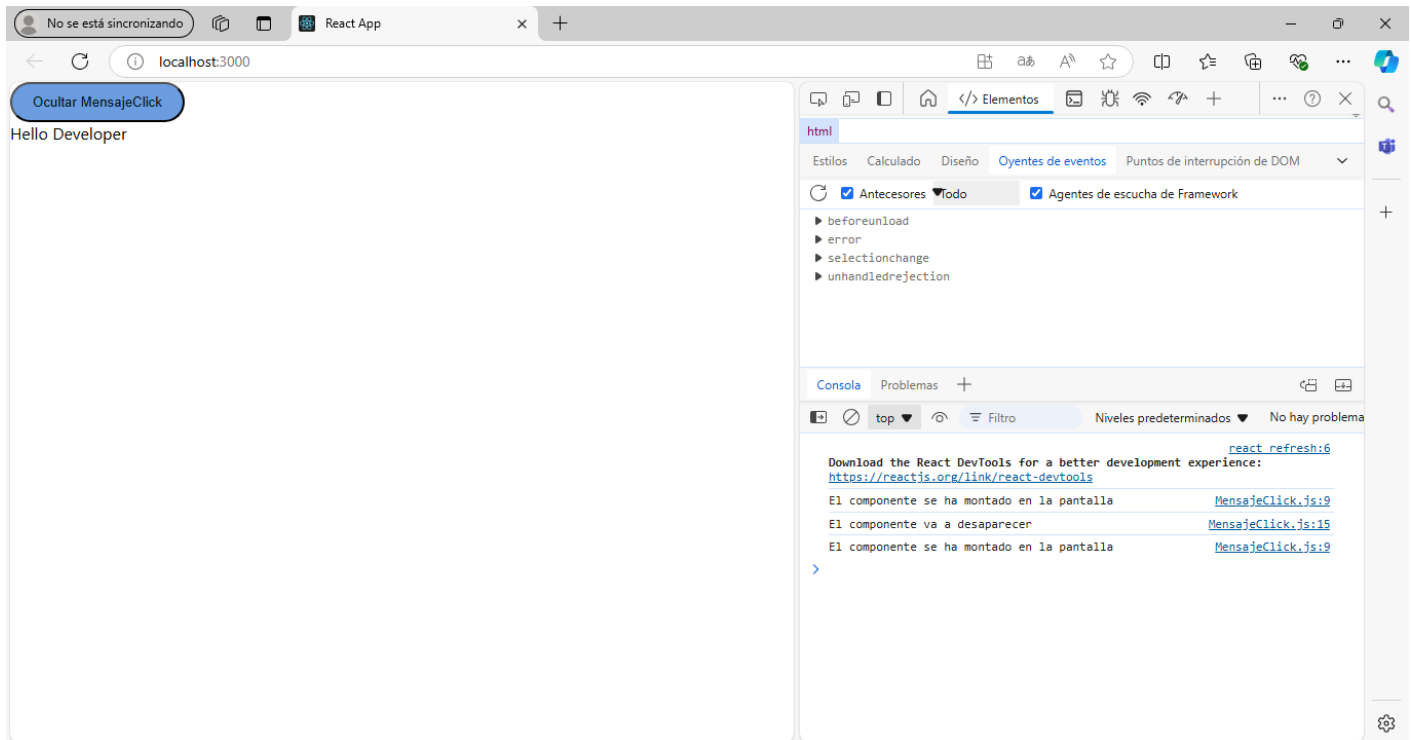
Ln 3, Col 45 Spaces: 4 UTF-8 CRLF JavaScript Prettier

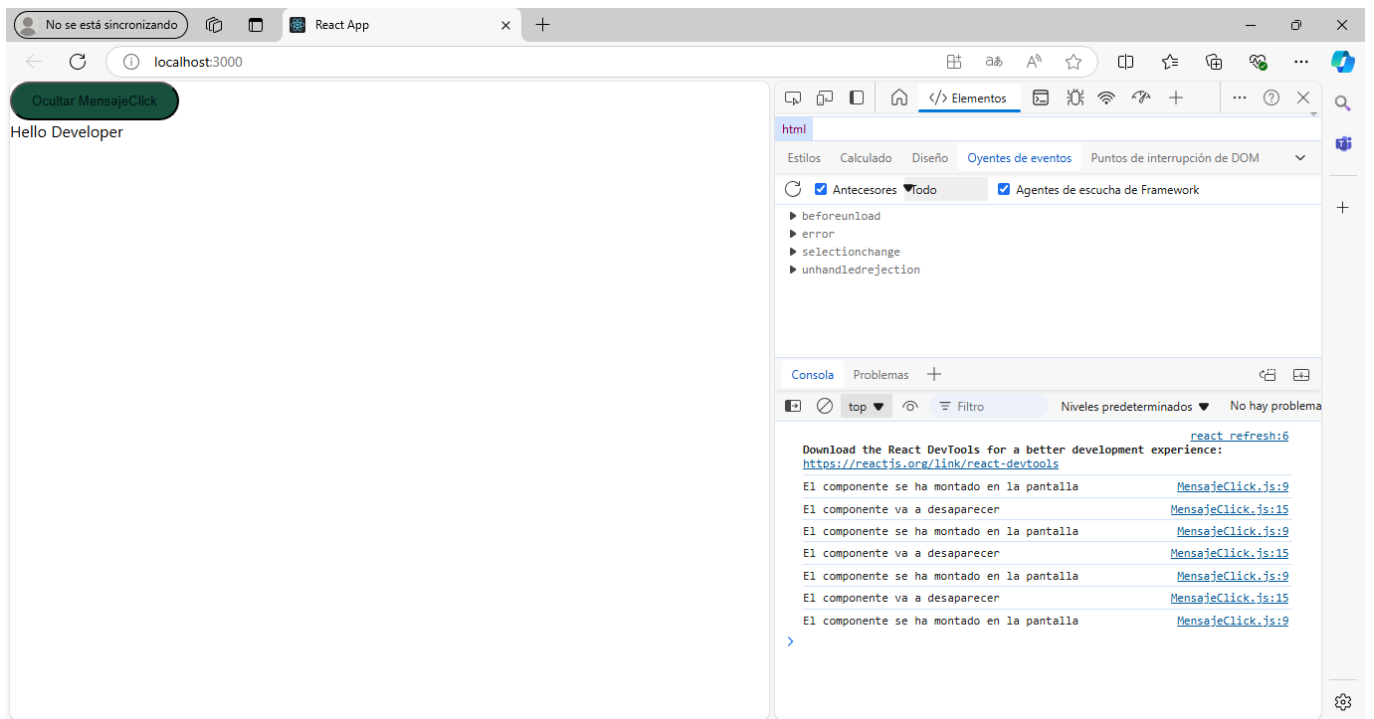
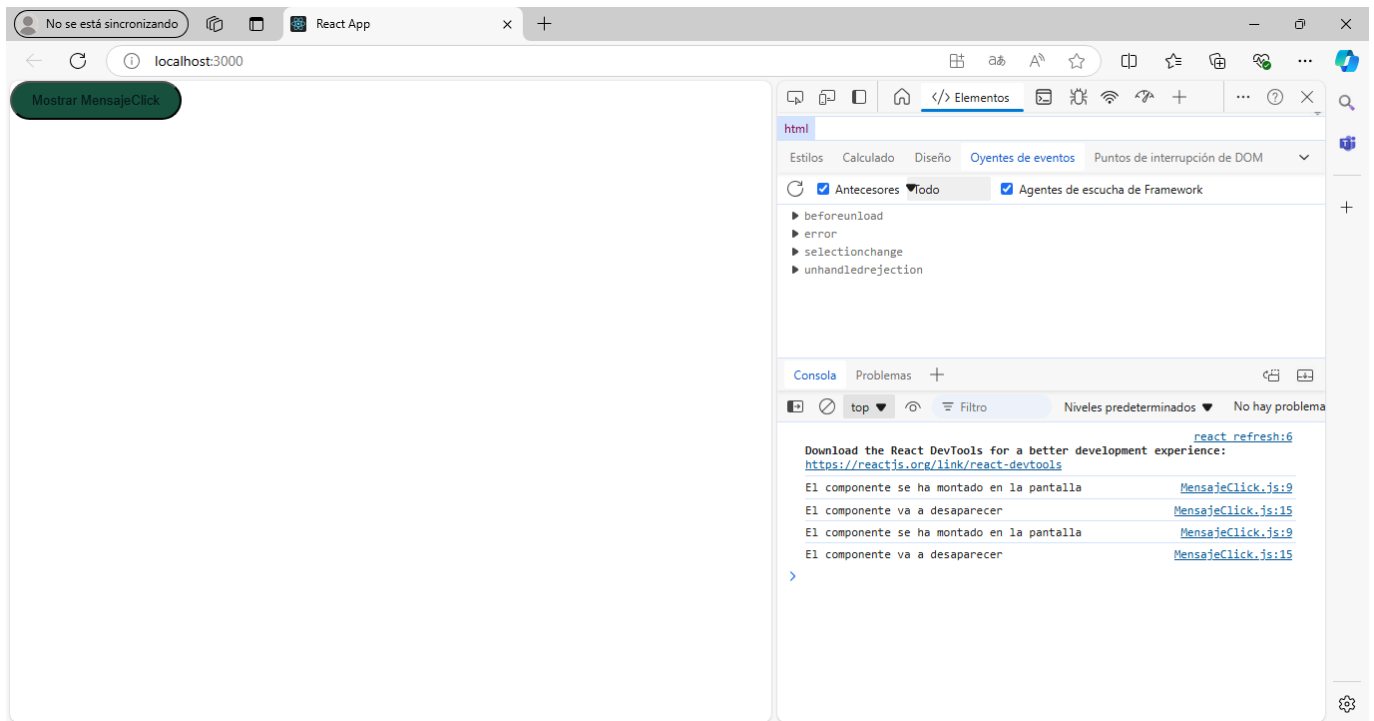
App.js

```
1 import React from "react";
2 import MensajeClick from './MensajeClick';
3
4 class App extends React.Component {
5   constructor(props){
6     super(props);
7     this.state = {mostrarComponente: true};
8   }
9   toggleComponent=() => {
10     this.setState((prevState =>({mostrarComponente: !prevState.mostrarComponente})))
11   }
12   render(){
13     return(
14       <div>
15         <button onClick={this.toggleComponent}>
16           {this.state.mostrarComponente
17             ? "Ocultar MensajeClick"
18             : "Mostrar MensajeClick"}
19         </button>
20         {this.state.mostrarComponente && <MensajeClick/>}
21       </div>
22     );
23   }
24 }
25 export default App;
26
```

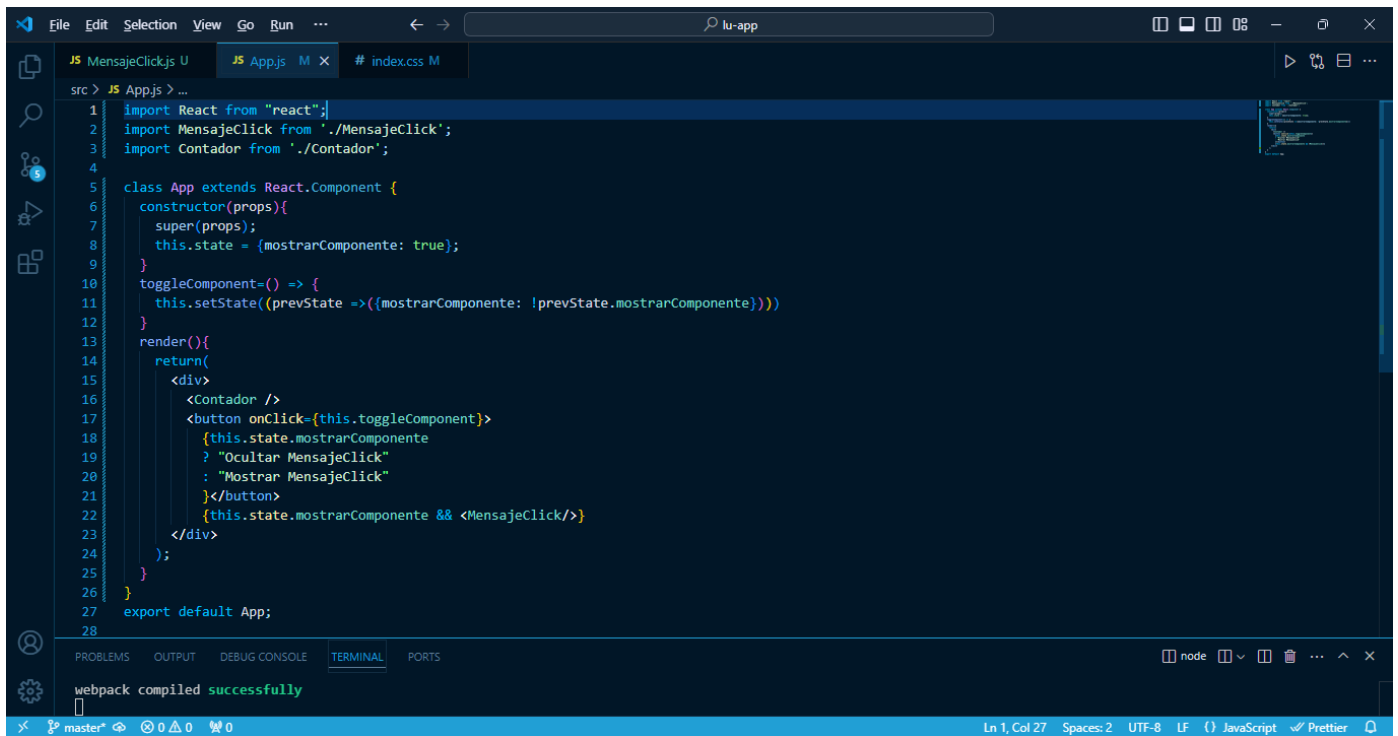
webpack compiled successfully

Resultado:





Ejercicio 1 y 2 en App.js



```
1 import React from "react";
2 import MensajeClick from "../MensajeClick";
3 import Contador from "../Contador";
4
5 class App extends React.Component {
6   constructor(props) {
7     super(props);
8     this.state = {mostrarComponente: true};
9   }
10  toggleComponent() => {
11    this.setState((prevState => ({mostrarComponente: !prevState.mostrarComponente})));
12  }
13  render() {
14    return (
15      <div>
16        <Contador />
17        <button onClick={this.toggleComponent}>
18          {this.state.mostrarComponente
19            ? "Ocultar MensajeClick"
20            : "Mostrar MensajeClick"}
21        </button>
22        {this.state.mostrarComponente && <MensajeClick />}
23      </div>
24    );
25  }
26 }
27 export default App;
28
```

webpack compiled successfully

Resultado:

