Universidad Juárez del Estado de Durango

Facultad de Ingeniería Ciencias y Arquitectura

Ingeniería en Tecnologías Computacionales

Programación Web

Reporte de Práctica 23

Diego Rea Morales

Ing. Fabian Gallegos Gutiérrez

7 de Noviembre del 2025

Índice

Contenido_____3

- Explicación del código de JavaScript_____4

Conclusiones_____6

Contenido

En la práctica del día de hoy realizaremos una calculadora en una pagina web, esta será realizada en React.

La calculadora constatará de un pad numérico y aparte de las operaciones aritméticas básicas, elevar un número al cuadrado, punto decimal, igual y borrar los datos.

```css
.calculator {
  width: 300px;
  margin: 50px auto;
  border: 1px solid #ccc;
  border-radius: 5px;
  background-color: #f3f3f3;
  padding: 10px;
}
.display {
  margin-bottom: 10px;
}
.display input {
  width: calc(100% - 20px);
  height: 30px;
  margin-bottom: 5px;
  padding: 5px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
}
button {
  width: calc(100% - 10px);
  height: 40px;
  margin: 5px;
  padding: 5px;
  font-size: 18px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
button:hover {
  background-color: #eaeaea;
}
button.operator {
  grid-column: span 2;
}
button.equal {
  grid-column: span 4;
}
```

```jsx
import { useState } from "react";
import "./App.css";

function App() {
  const [expression, setExpression] = useState("");
  const [result, setResult] = useState("");
  const handleClick = (value) => {
    if (value === "=") {
      try {
        const calculatedResult = eval(expression);
        setResult(calculatedResult);
      } catch (error) {
        setResult("Error");
      }
    } else if (value === "C") {
      setExpression("");
      setResult("");
    } else {
      setExpression((prevExpression) => prevExpression + value);
    }
  };

  return (
    <div className="calculator">
      <div className="display">
        <input type="text" value={expression} readOnly />
        <input type="text" value={result} readOnly />
      </div>
      <div className="buttons">
        <button onClick={() => handleClick("7")}>7</button>
        <button onClick={() => handleClick("8")}>8</button>
        <button onClick={() => handleClick("9")}>9</button>
        <button onClick={() => handleClick("+")}>+</button>
        <button onClick={() => handleClick("4")}>4</button>
        <button onClick={() => handleClick("5")}>5</button>
        <button onClick={() => handleClick("6")}>6</button>
        <button onClick={() => handleClick("-")}>-</button>
        <button onClick={() => handleClick("1")}>1</button>
        <button onClick={() => handleClick("2")}>2</button>
        <button onClick={() => handleClick("3")}>3</button>
        <button onClick={() => handleClick("*")}>*</button>
        <button onClick={() => handleClick("0")}>0</button>
        <button onClick={() => handleClick(".")}>.</button>
        <button onClick={() => handleClick("=")}>=</button>
        <button onClick={() => handleClick("/")}>/</button>
        <button onClick={() => handleClick("C")}>C</button>
        <button onClick={() => handleClick("**2")}>^2</button>
      </div>
    </div>
  );
}

export default App;
```

Explicación del código JavaScript

```
const [expression, setExpression] = useState("");
const [result, setResult] = useState("");
```

Primero, declaramos dos *hooks*, uno para la expresión que se va a ingresar y otro para le resultado.

```
const handleClick = (value) => {
    if (value === "=") {
      try {
        const calculatedResult = eval(expression);
        setResult(calculatedResult);
      } catch (error) {
        setResult("Error");
      }
    } else if (value === "C") {
      setExpression("");
      setResult("");
    } else {
      setExpression((prevExpression) => prevExpression + value);
    }
  };
```

Después, el evento que se va a ejecutar cuando, se ingrese un dato y/o se ejecute el botón de resultado o de borrar. Este tiene como parámetro el valor a evaluar.

Este valor viene dado desde los botones de los caracteres.

```
<button onClick={() => handleClick("7")}>7</button>
<button onClick={() => handleClick("8")}>8</button>
<button onClick={() => handleClick("9")}>9</button>
```

Después ese valor es verificado en un condicional.

La misión de este condicional es la de verificar si es que se trata de que se va a evaluar la expresión, es decir, realizar el calculo, borrar los datos o seguir ingresando datos.

La primera condición toma en cuenta si es que el carácter que se ingresó es el símbolo de igual. Si es así, se realiza un try catch, esto para evitar errores y notificárselo al usuario.

Si no hay errores, el programa evalúa la expresión, usando el método `eval();`,

para después, setear el valor de *result* con el resultado de evaluar la expresión.

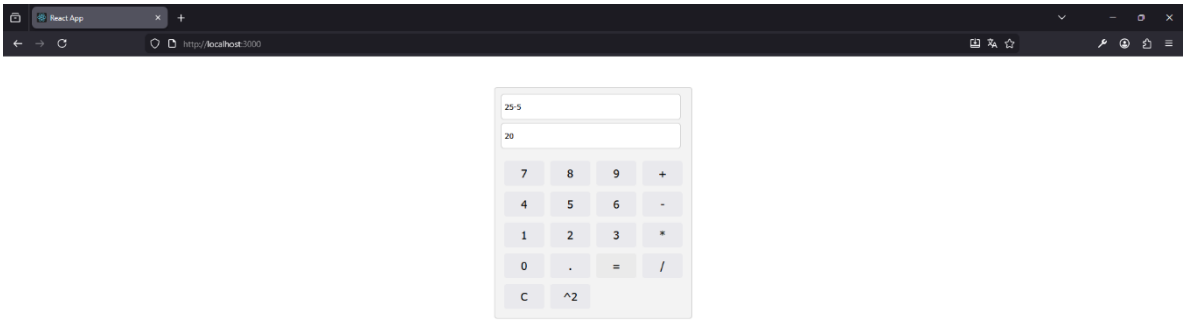
```
setResult(calculatedResult);
```

Si el carácter es igual a C, es decir el símbolo que indica que tiene que borrar los datos, este simplemente seteara los valores de la expresión y el resultado a ceros.

```
setExpression("");
setResult("");
```

Si no es ni uno de esas dos condiciones, es decir si no se va a borrar los datos o se va a calcular, simplemente se hace una concatenación de los caracteres que se hayan ingresado, esto para poder crear la expresión para que cuando el usuario quiera calcular el resultado, esta expresión ya este completa, utilizando el valor heredado por el evento de click del botón para agregar un valor.

```
setExpression((prevExpression) => prevExpression + value);
```



Conclusiones

En resumen, esta práctica reforzamos el uso de *hook*s useState para guardar la expresión y el resultado, dándole "memoria" a la calculadora. Toda la lógica se centró de forma limpia en la función handleClick, que decide qué hacer (añadir, borrar o calcular) según el botón presionado.

Aunque eval() fue un atajo útil para calcular, el try...catch fue igual de importante para manejar errores y evitar que la aplicación se rompiera. E