

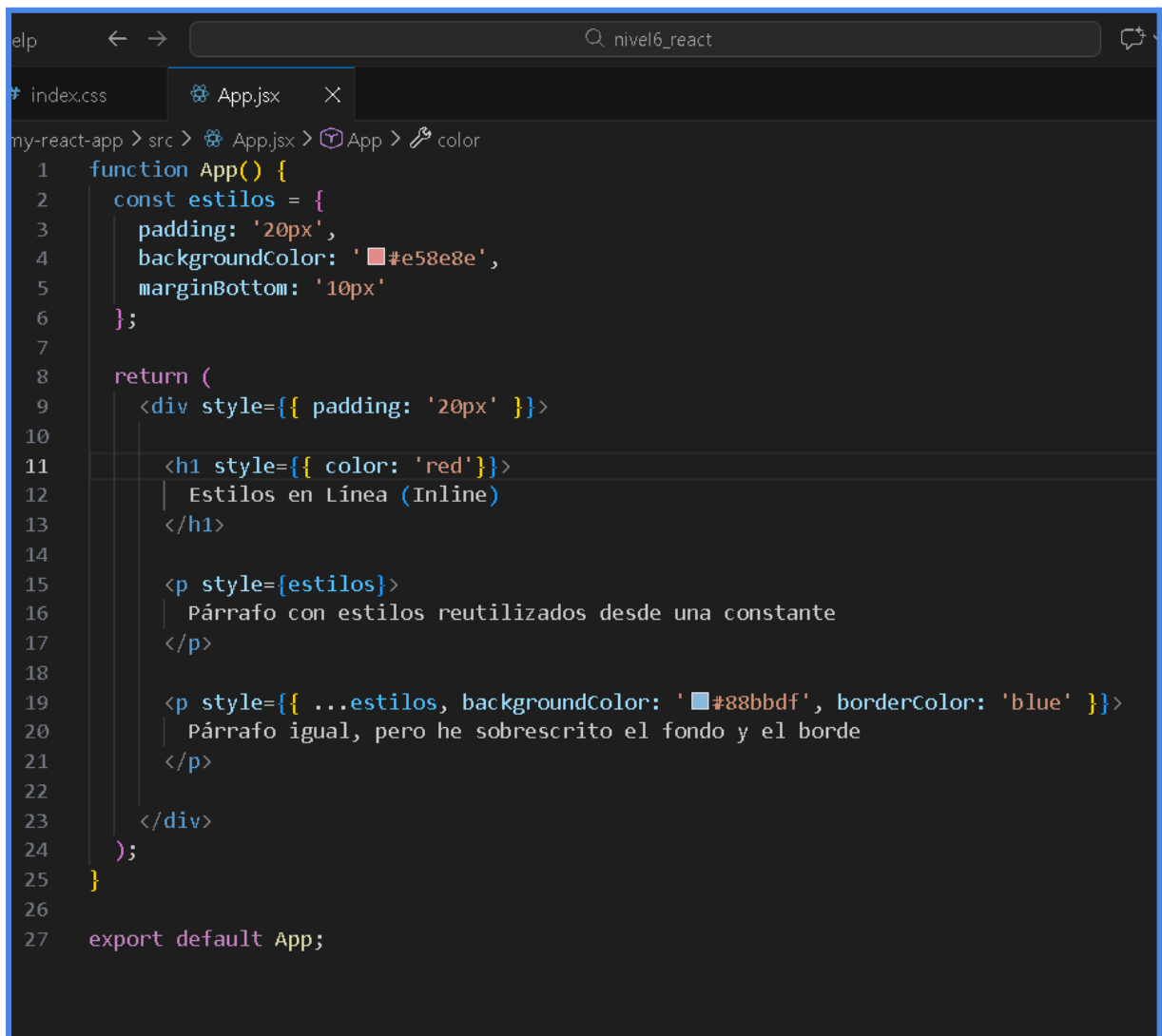
Nivel 7 React

DAW2-DWEC

Índice

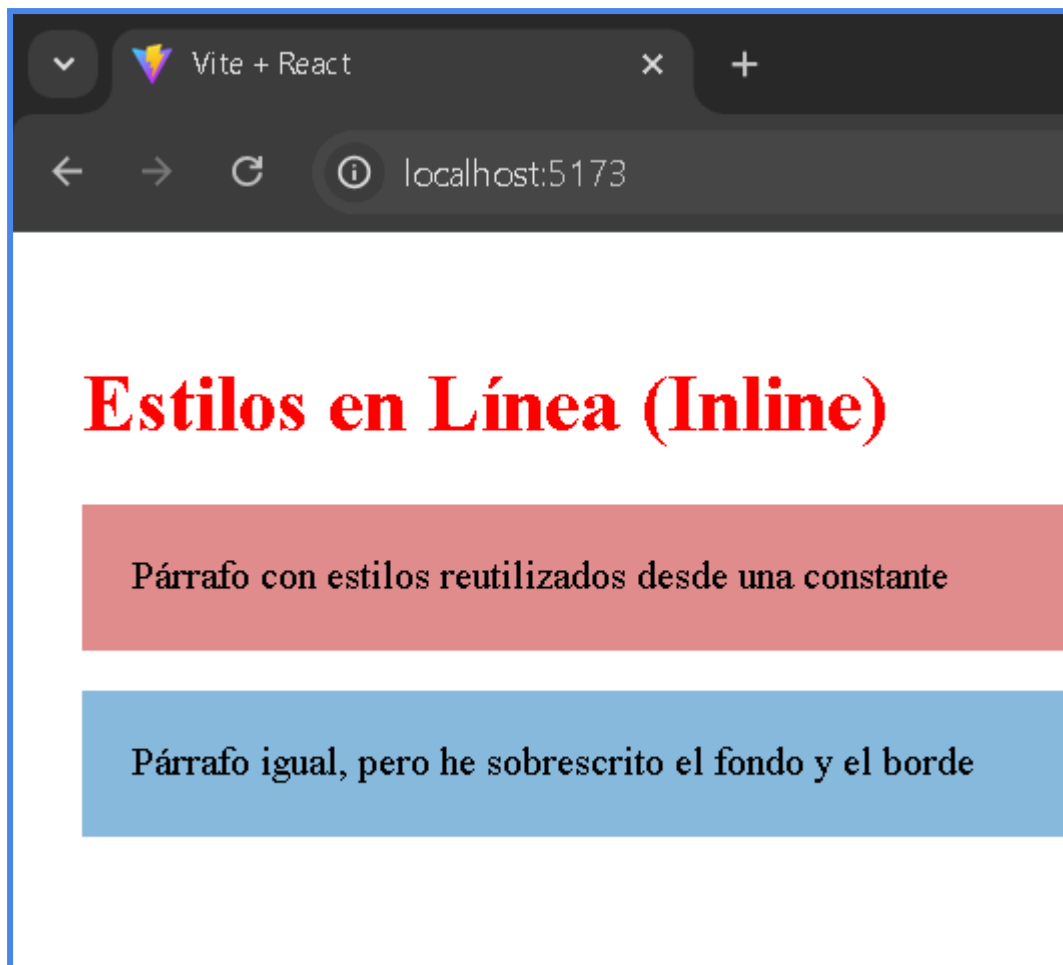
Parte A: Estilo en línea (inline styles).....	2
Parte B: Archivos CSS (global o por componente).....	2
Parte C: CSS Modules (alcance local por defecto).....	2
Parte D: Tailwind CSS (utilidades).....	2
Miniretillo (pero esta vez sin ayuda).....	2
Preguntas extra para el PDF.....	3

Parte A: Estilo en línea (inline styles)



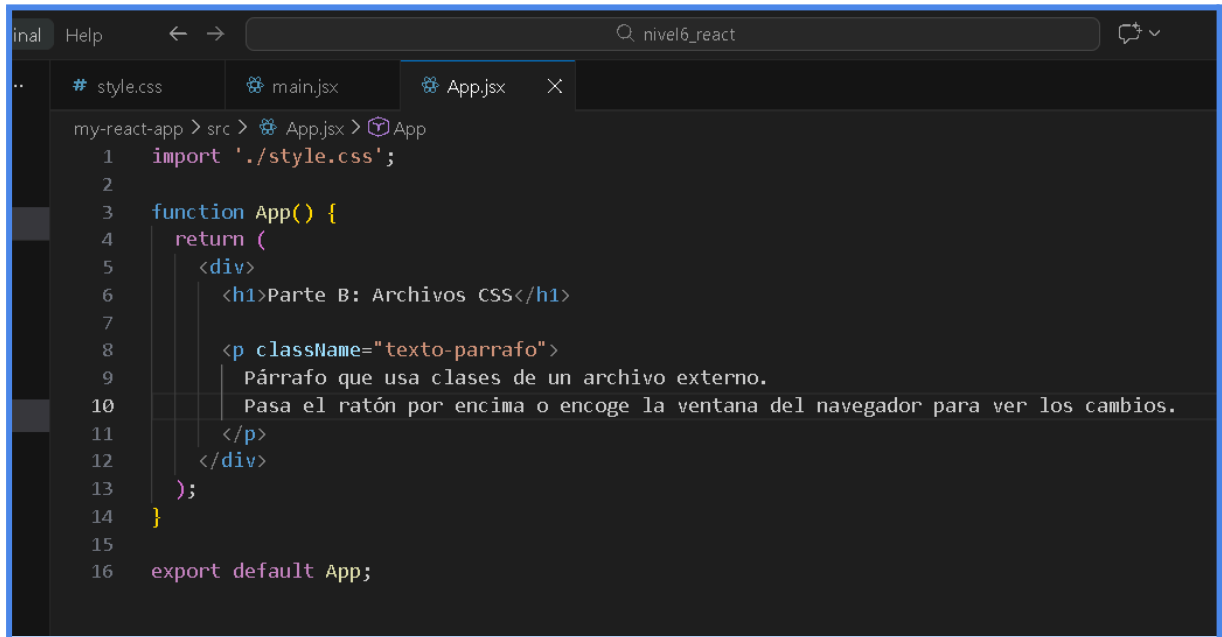
```
1  function App() {
2    const estilos = {
3      padding: '20px',
4      backgroundColor: '#e58e8e',
5      marginBottom: '10px'
6    };
7
8    return (
9      <div style={{ padding: '20px' }}>
10
11        <h1 style={{ color: 'red' }}>
12          Estilos en Línea (Inline)
13        </h1>
14
15        <p style={estilos}>
16          Párrafo con estilos reutilizados desde una constante
17        </p>
18
19        <p style={{ ...estilos, backgroundColor: '#88bbdf', borderColor: 'blue' }}>
20          Párrafo igual, pero he sobrescrito el fondo y el borde
21        </p>
22      </div>
23    );
24  }
25
26
27  export default App;
```

“Figura 01, estilo en línea, elaboración propia”



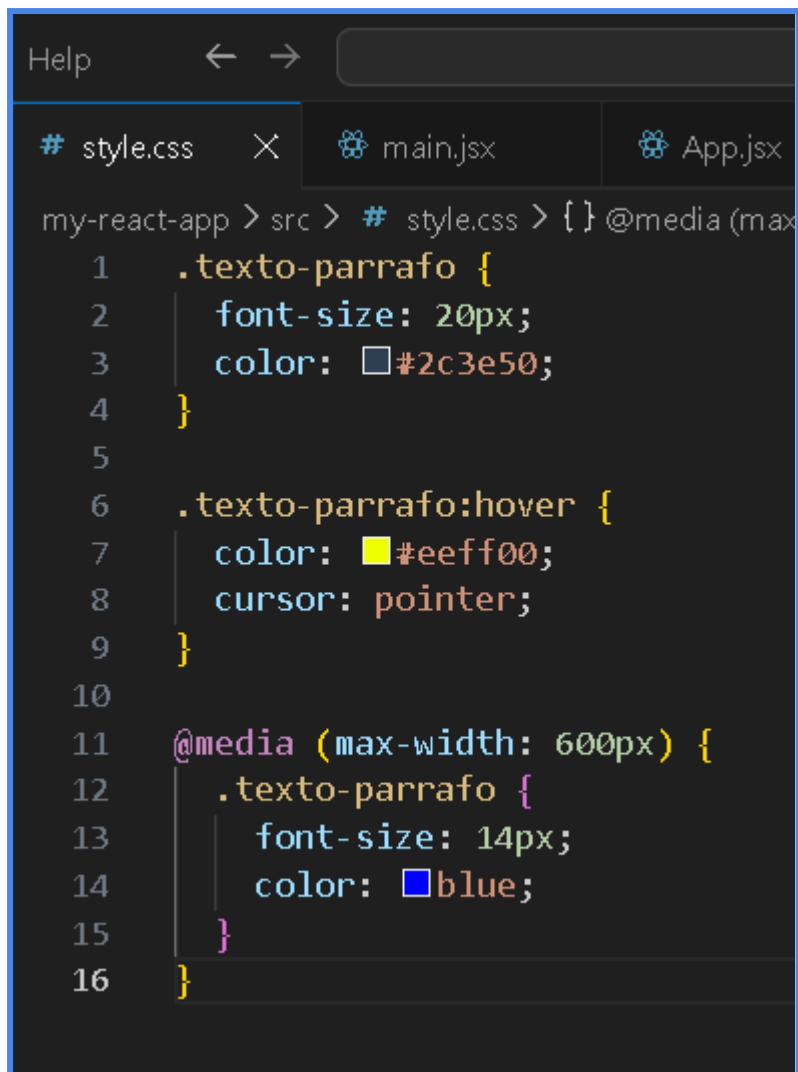
“Figura 02, muestra en pantalla, elaboración propia”

Parte B: Archivos CSS (global o por componente)



```
my-react-app > src > App.jsx > App
1  import './style.css';
2
3  function App() {
4    return (
5      <div>
6        <h1>Parte B: Archivos CSS</h1>
7
8        <p className="texto-parrafo">
9          Párrafo que usa clases de un archivo externo.
10         Pasa el ratón por encima o encoge la ventana del navegador para ver los cambios.
11       </p>
12     </div>
13   );
14 }
15
16 export default App;
```

“Figura 03, frase que cambia de color, elaboración propia”

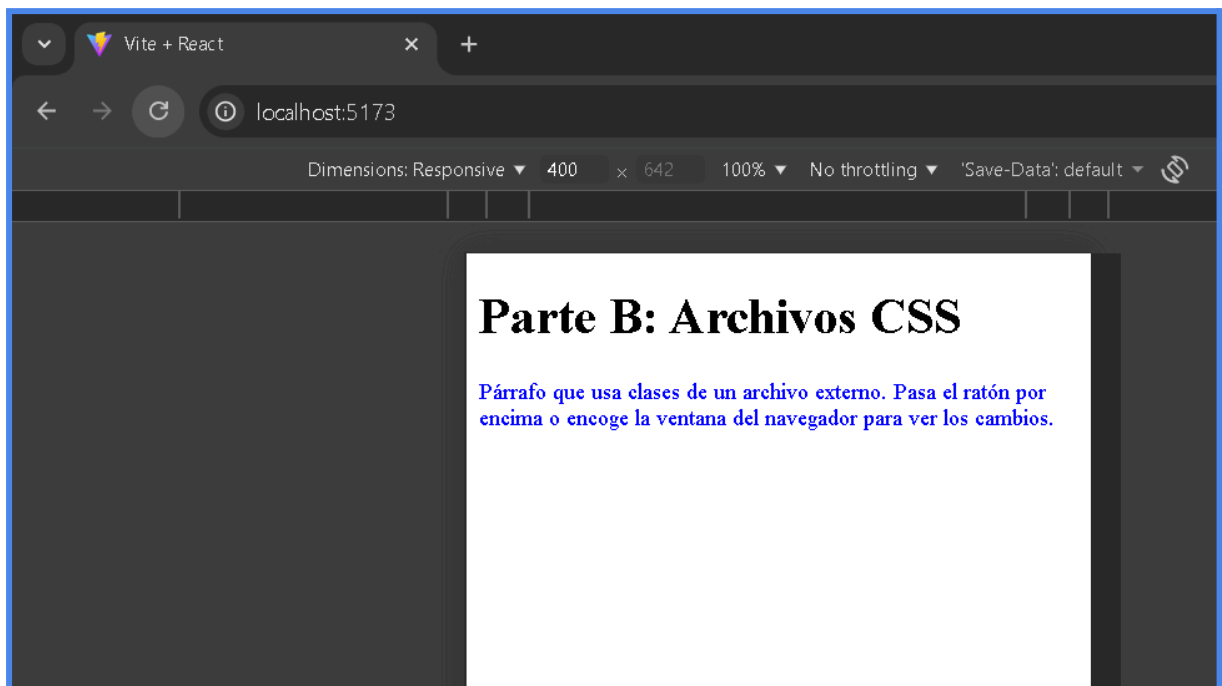


```
my-react-app > src > # style.css > {} @media (max-width: 600px) {
1  .texto-parrafo {
2    font-size: 20px;
3    color: #2c3e50;
4  }
5
6  .texto-parrafo:hover {
7    color: #eeff00;
8    cursor: pointer;
9  }
10
11 @media (max-width: 600px) {
12   .texto-parrafo {
13     font-size: 14px;
14     color: blue;
15   }
16 }
```

“Figura 04, estilos.css con una clase, elaboración propia”

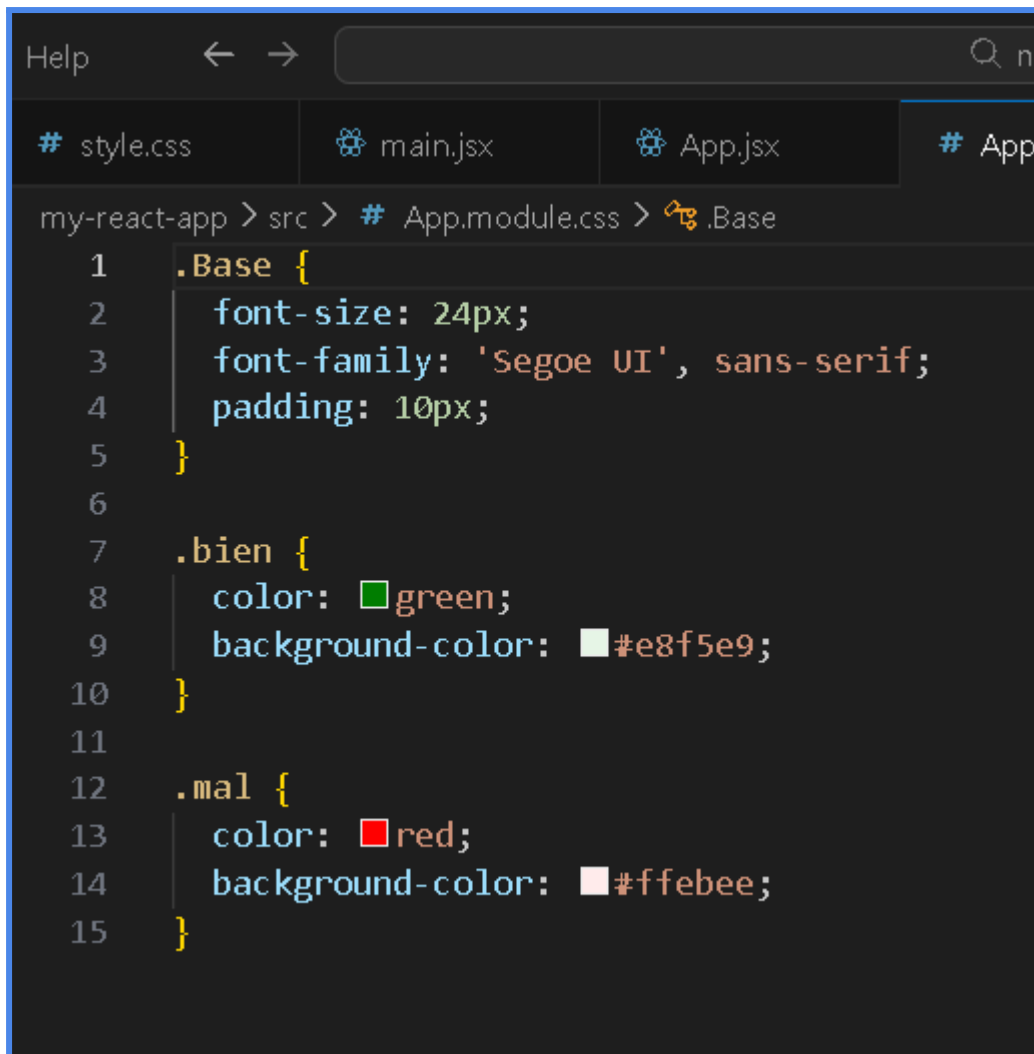


“Figura 05, muestra en pantalla a tamaño normal, elaboración propia”



“Figura 06, muestra en pantalla a tamaño móvil (media queries), elaboración propia”

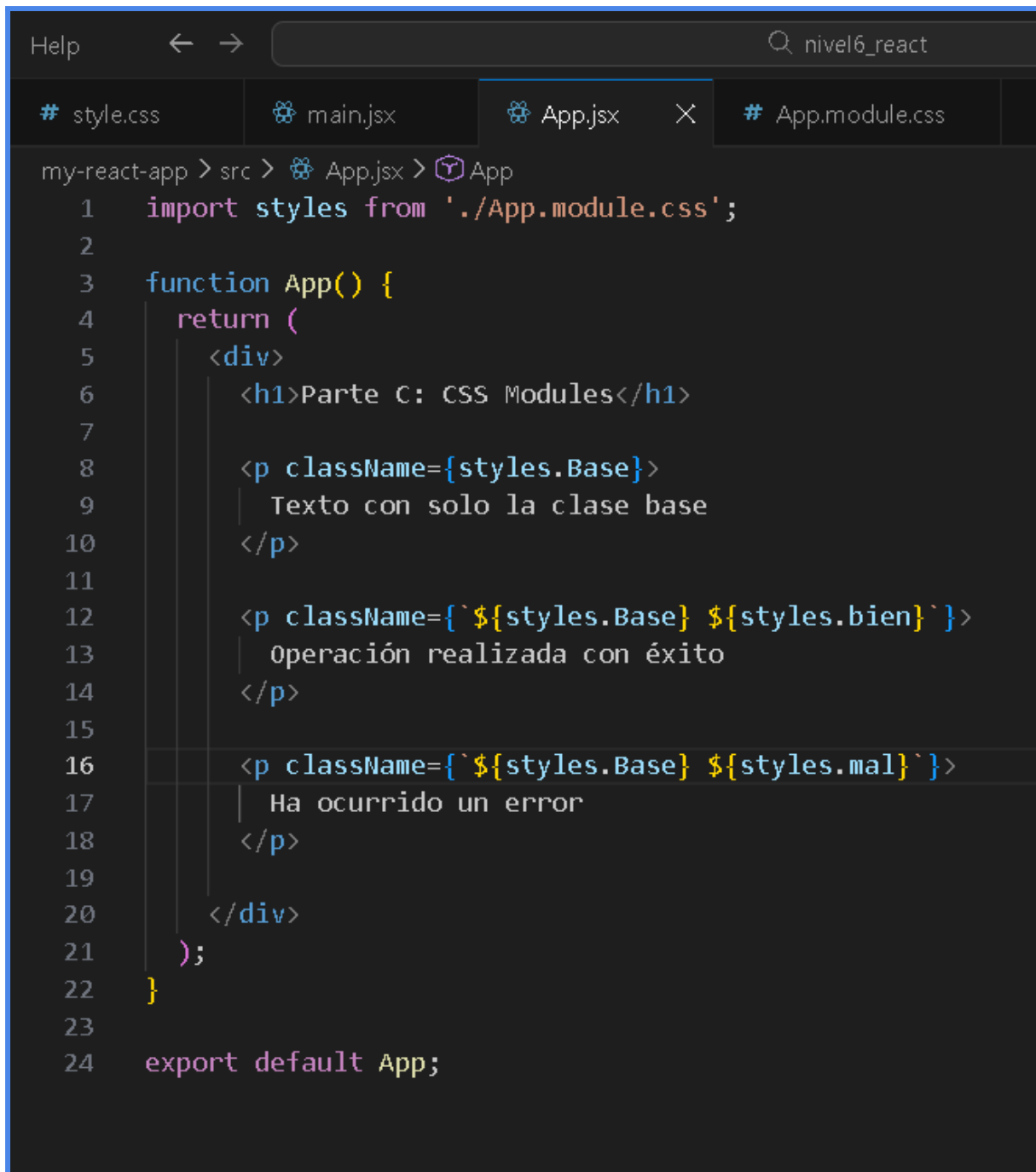
Parte C: CSS Modules (alcance local por defecto)



The image shows a code editor window with a dark theme. At the top, there is a 'Help' menu, navigation arrows, and a search bar. Below the menu, there are tabs for '# style.css', 'main.jsx', 'App.jsx', and '# App'. The active tab is '# App'. The breadcrumb path is 'my-react-app > src > # App.module.css > .Base'. The code is as follows:

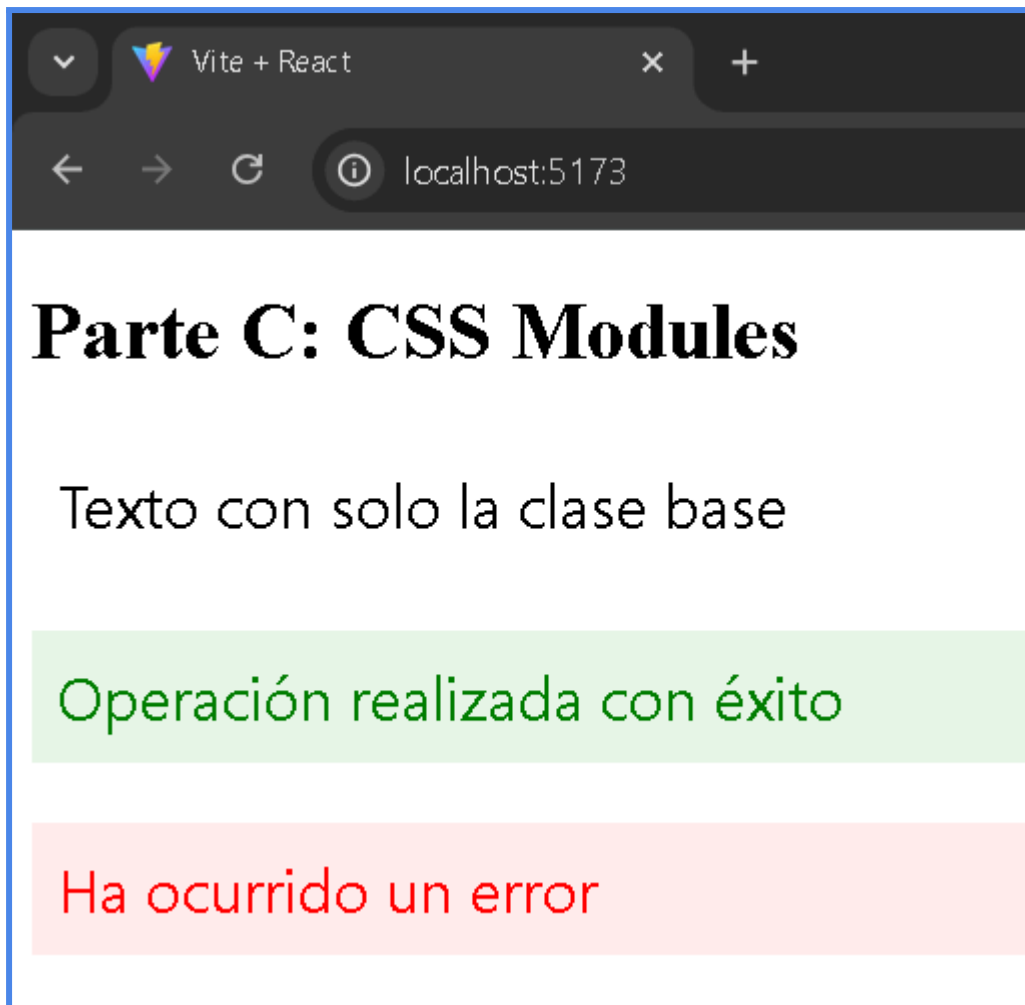
```
1  .Base {
2    font-size: 24px;
3    font-family: 'Segoe UI', sans-serif;
4    padding: 10px;
5  }
6
7  .bien {
8    color: green;
9    background-color: #e8f5e9;
10 }
11
12 .mal {
13   color: red;
14   background-color: #ffebee;
15 }
```

“Figura 07, css personalizado, elaboración propia”



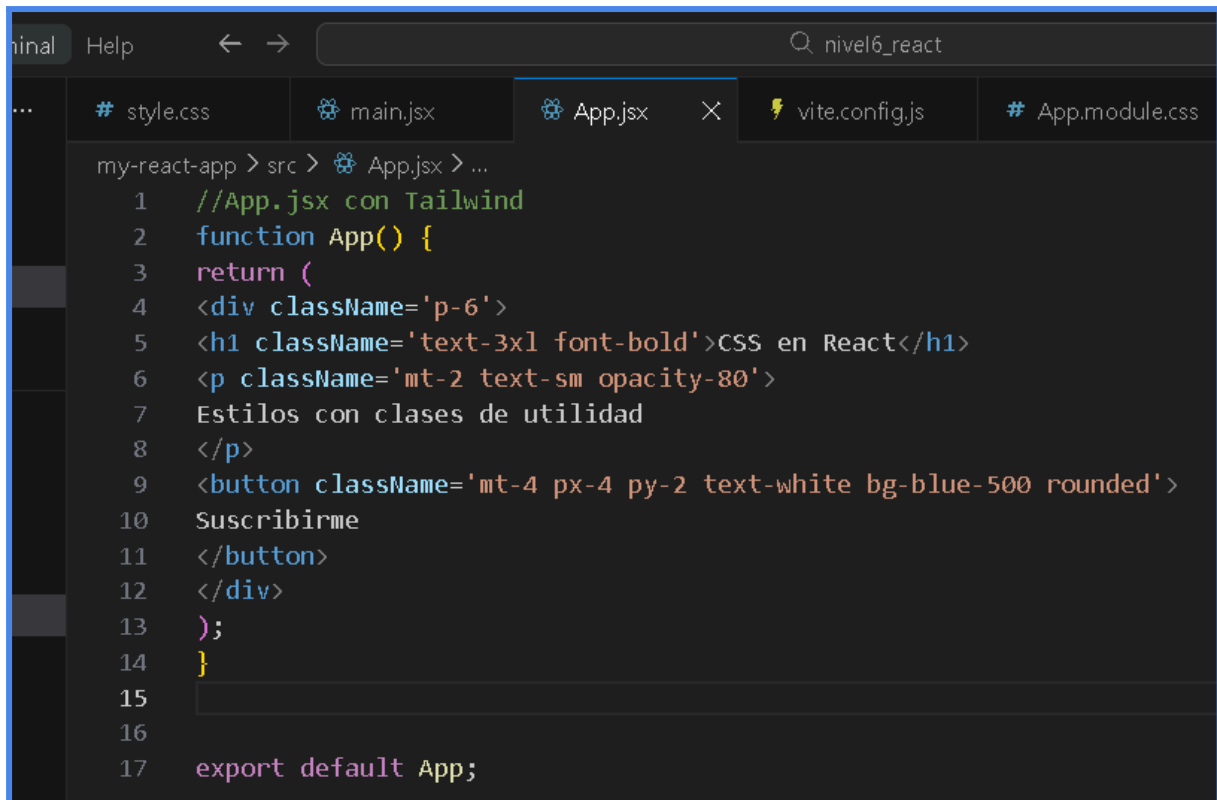
```
my-react-app > src > App.jsx > App
1  import styles from './App.module.css';
2
3  function App() {
4    return (
5      <div>
6        <h1>Parte C: CSS Modules</h1>
7
8        <p className={styles.Base}>
9          | Texto con solo la clase base
10       </p>
11
12       <p className={` ${styles.Base} ${styles.bien}`}>
13         | Operación realizada con éxito
14       </p>
15
16       <p className={` ${styles.Base} ${styles.mal}`}>
17         | Ha ocurrido un error
18       </p>
19     </div>
20   );
21 }
22
23
24 export default App;
```

“Figura 08, parte app principal con éxito y error, elaboración propia”



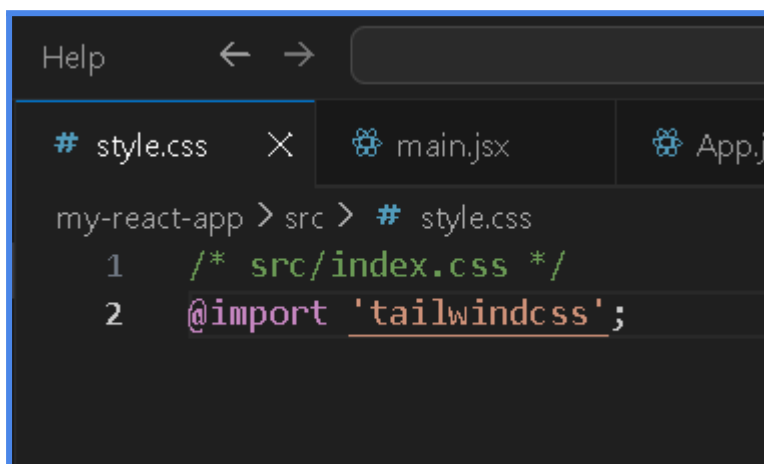
“Figura 09, muestra en pantalla, elaboración propia”

Parte D: Tailwind CSS (utilidades)



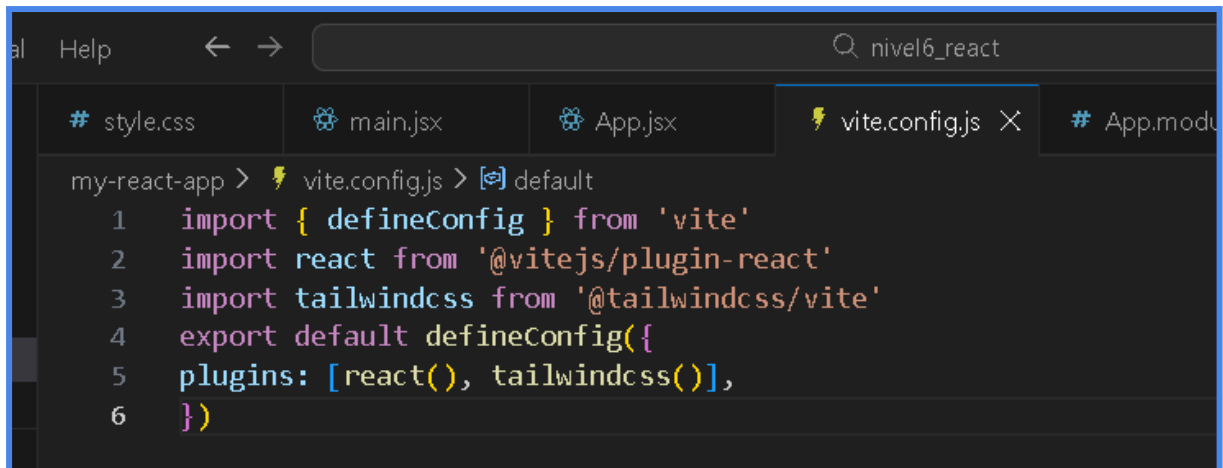
```
my-react-app > src > App.jsx > ...
1 //App.jsx con Tailwind
2 function App() {
3   return (
4     <div className='p-6'>
5       <h1 className='text-3xl font-bold'>CSS en React</h1>
6       <p className='mt-2 text-sm opacity-80'>
7         Estilos con clases de utilidad
8       </p>
9       <button className='mt-4 px-4 py-2 text-white bg-blue-500 rounded'>
10        Suscribirse
11      </button>
12    </div>
13  );
14 }
15
16
17 export default App;
```

“Figura 10, app con tailwind, elaboración propia”



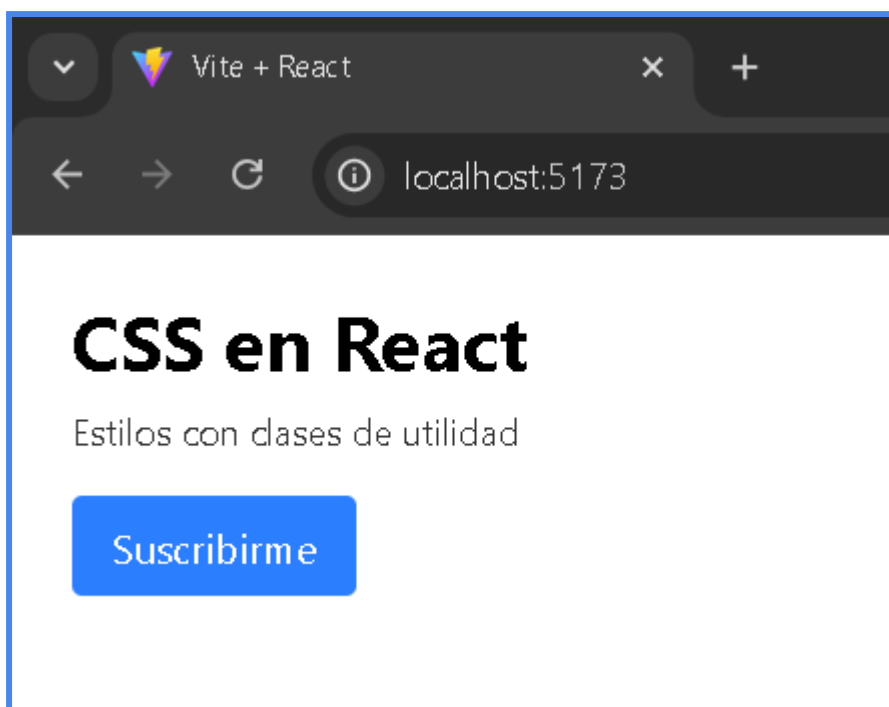
```
my-react-app > src > # style.css
1 /* src/index.css */
2 @import 'tailwindcss';
```

“Figura 11, css con tailwind importado, elaboración propia”



```
my-react-app > vite.config.js > default
1  import { defineConfig } from 'vite'
2  import react from '@vitejs/plugin-react'
3  import tailwindcss from '@tailwindcss/vite'
4  export default defineConfig({
5    plugins: [react(), tailwindcss()],
6  })
```

“Figura 12, archivo [vite.config.js](#) con los plugins, elaboración propia”

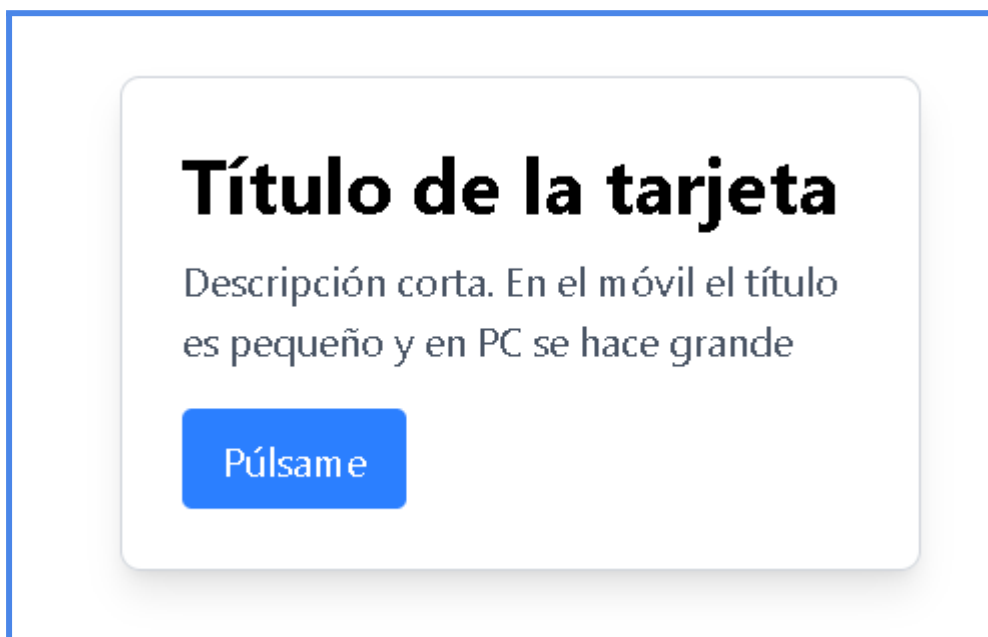


“Figura 13, muestra en pantalla, elaboración propia”

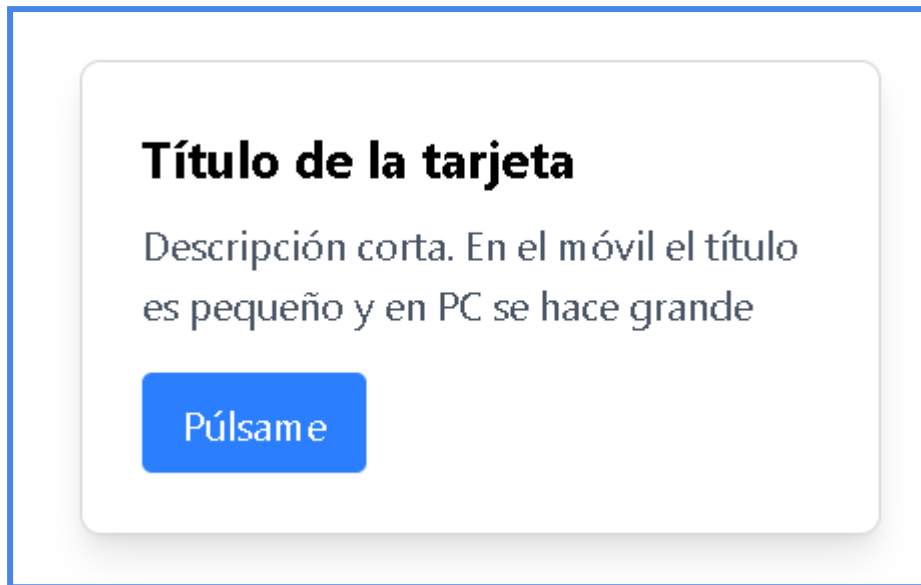
Miniretillo (seguimos sin ayuda)

```
Help  ← →  nivel6_react
# style.css  main.jsx  App.jsx  vite.config.js  App.module.css
my-react-app > src > App.jsx > App
1  function App() {
2    return (
3      <div className="p-10 flex justify-center">
4
5        <div className="border border-gray-300 rounded-lg shadow-lg p-6 max-w-xs">
6
7          <h2 className="text-xl md:text-3xl font-bold mb-2">
8            Título de la tarjeta
9          </h2>
10
11          <p className="text-gray-600 mb-4">
12            Descripción corta. En el móvil el título es pequeño y en PC se hace grande
13          </p>
14
15          <button className="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-700 transition">
16            Púlsame
17          </button>
18
19        </div>
20
21      </div>
22    )
23  }
24
25  export default App
```

“Figura 14, div con un título, un texto y un botón, elaboración propia”



“Figura 15, muestra en pantalla de PC, elaboración propia”



"Figura 16, muestra en pantalla de móvil, elaboración propia"

Preguntas extra para el PDF

1. ¿Qué diferencia hay entre `onClick` y `onSubmit`?

- El `onClick` detecta cuando le doy click con el ratón a cualquier cosa, mientras que `onSubmit` se puede hacer click tanto con el ratón o con la tecla enter.

2. ¿Por qué usamos `e.preventDefault()` en un formulario?

- Porque si no lo uso la página empieza a parpadear porque está cargando la página entera, si lo uso se carga de forma más suave y que no se puede ver

3. ¿Qué es una "entrada controlada" y por qué usamos `value` + `onChange`?

- Es cuando el que controla el input es React y no el navegador
- uso `value` para hacer que el input muestre lo que dice el estado si o si, y uso `onChange` para actualizar el estado mientras que escribo

4. En tu mini-reto, que estado(s) manejas y que evento(s) los actualizan?

- Manejo 3 estados:
- booleano para el párrafo que se muestra y se oculta
- número para el contador que suma y resta numeros
- texto para mostrar el modo en el que estoy

- Los actualizo con onClick para los botones y el contador y onChange para poder capturar lo que pongo en el input.