

Laboratorio N°6

Estudiante: Victoria Jimenez Martinez

C.I. : 8610323

R.U. : 103334

REACT

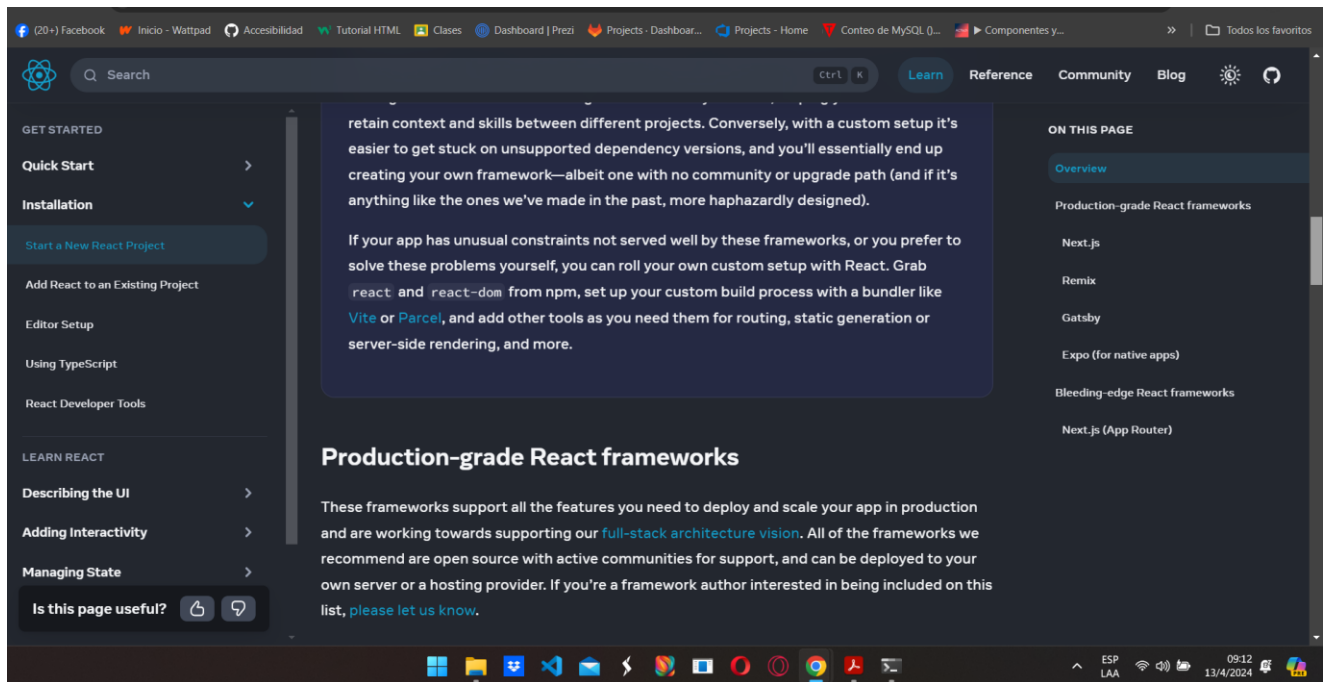
En el 2013 Facebook lanzo una biblioteca de JavaScript esta fue pensada para desarrollar UI dinámicas. No tengo ni idea lo que se comentaba sobre aquella biblioteca en aquel tiempo, si fue popular desde el inicio, o como todas las librerías empezaron a dudar de su utilidad. Esta biblioteca se llamaba “React”.

React al día de hoy, se encuentra en la versión 18 y próximo a lanzar la versión 19 la cual agregara funcionalidades interesantes.

Anteriormente se instalaba react mediante el siguiente comando: **npx create-react-app my-app**. Pero eso ya cambio, ahora hace uso de distintos frameworks los cuales puedes ver aquí: <https://react.dev/learn/start-a-new-react-project>

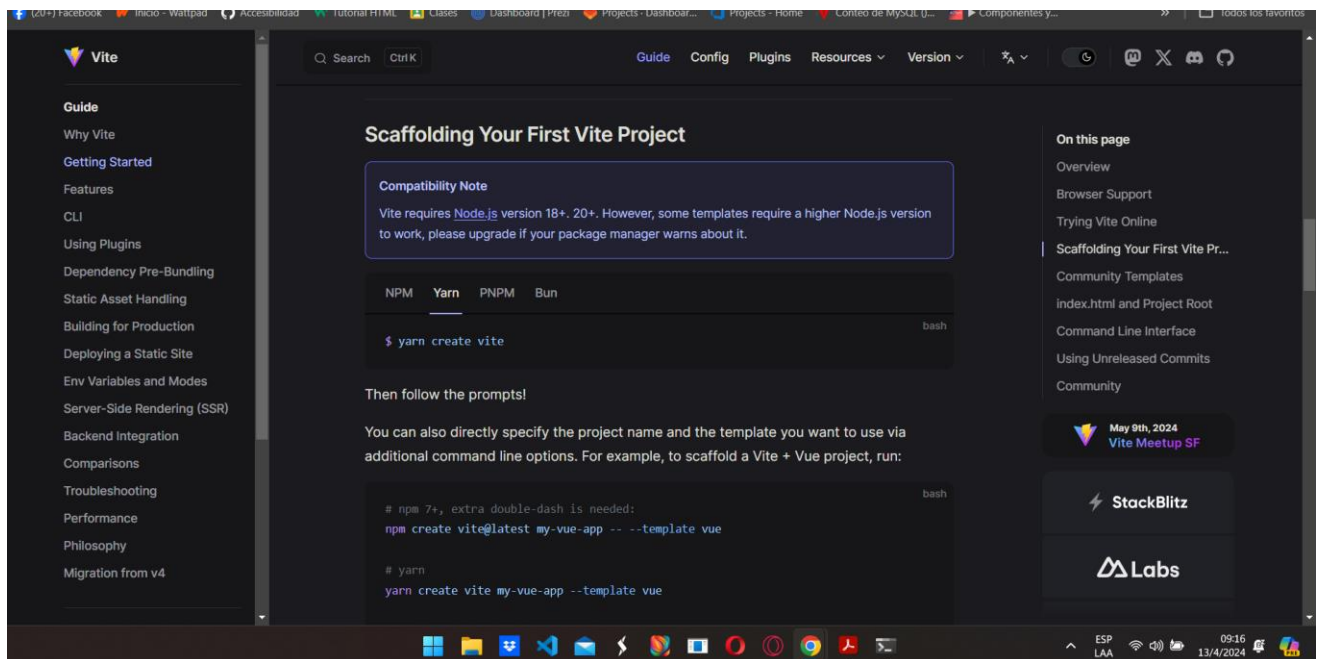
No necesariamente se necesita de los frameworks para hacer uso de React, bastara con configurar un entypoint para React, para poder usarlo adecuadamente. Pero eso es tema aparte. Realmente tampoco es una gran configuración que realizar, así que, si te interesa, puedes averiguarlo por tu cuenta.

En este caso usaremos Vite, tomate unos segundos para buscarlo en el enlace proporcionado arriba. ¿Lo encontraste? Talvez tu respuesta sea un no, eso se debe a que está muy oculto y es algo rebuscado, como la letra pequeña en los contratos.



Puedes ver más acerca de Vite en: <https://vitejs.dev/>

Bien comencemos creando el proyecto inicial, ejecuta el siguiente comando, veras que hay diferentes opciones, elige la que prefieras, en mi caso usare yarn:



Desde la terminal ve al directorio donde quieras ejecutar el comando. Y les pedirá lo siguiente:

```
C:\Users\Hp\Desktop\Seminario\LAB07>yarn create vite
yarn create v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Installed "create-vite@5.2.3" with binaries:
  - create-vite
  - cva
✓ Project name: ... LAB06
✓ Package name: ... lab06
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  Remix ↗

Seleccionaremos JavaScript

Trabajaremos con React una vez creado el proyecto nos dirá que pasos seguir para ejecutar el proyecto.
```

Trabajaremos con React una vez creado el proyecto nos dirá que pasos seguir para ejecutar el proyecto.

```
Scaffolding project in C:\Users\Hp\Desktop\Seminario\LAB07\LAB06...
Done. Now run:

  cd LAB06
  yarn
  yarn dev

Done in 201.56s.

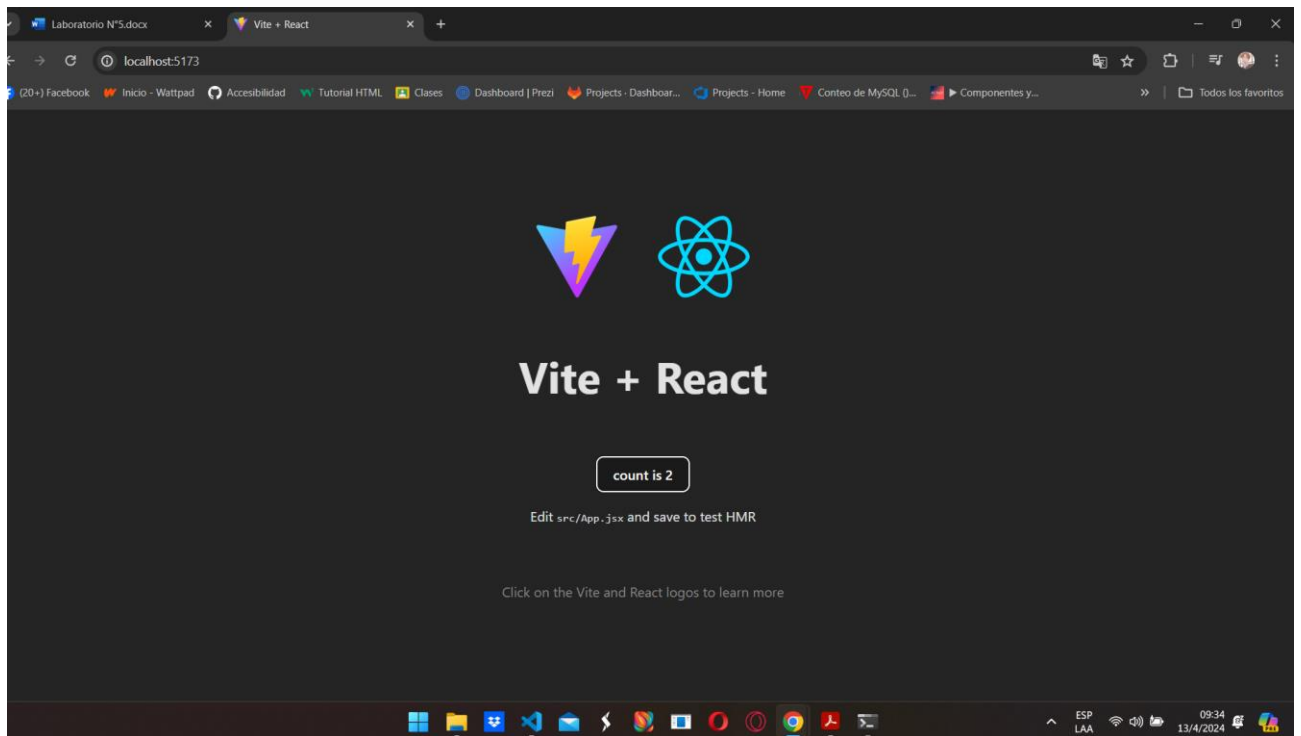
C:\Users\Hp\Desktop\Seminario\LAB07>|

Trabajaremos con React una vez creado el proyecto.
```

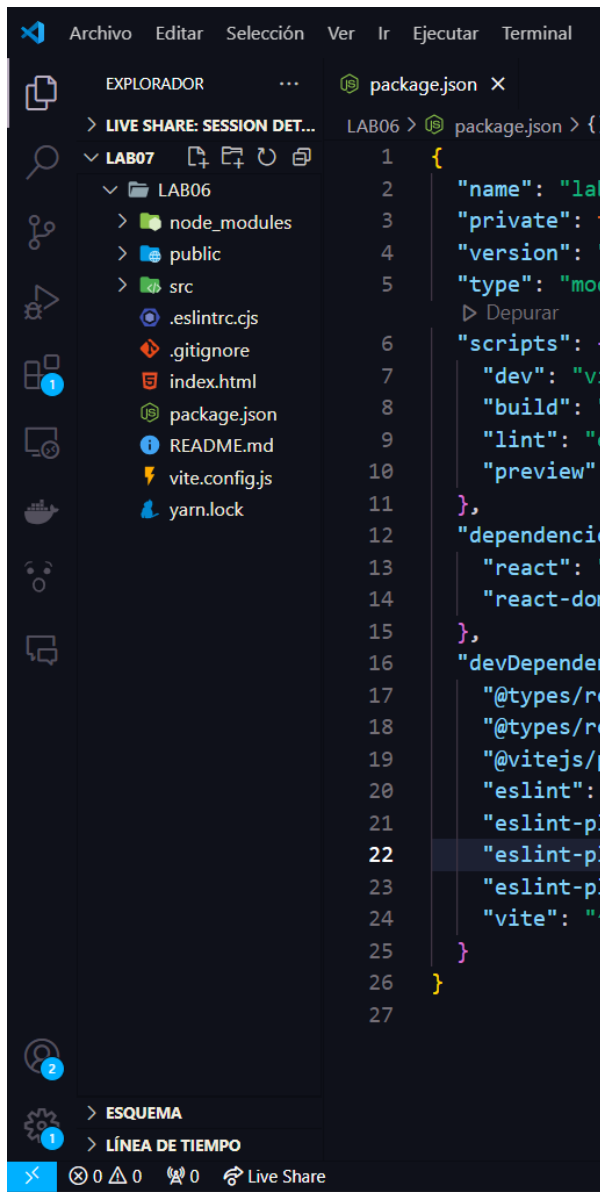
Psd: me equivoque de número de laboratorio. Si seguiste los pasos correctamente se mostrará lo siguiente:

```
VITE v5.2.8 ready in 5561 ms
Local: http://localhost:5173/
Network: use --host to expose
press h + enter to show help
| REACT
```

Si vas a esa dirección se mostrará lo siguiente:



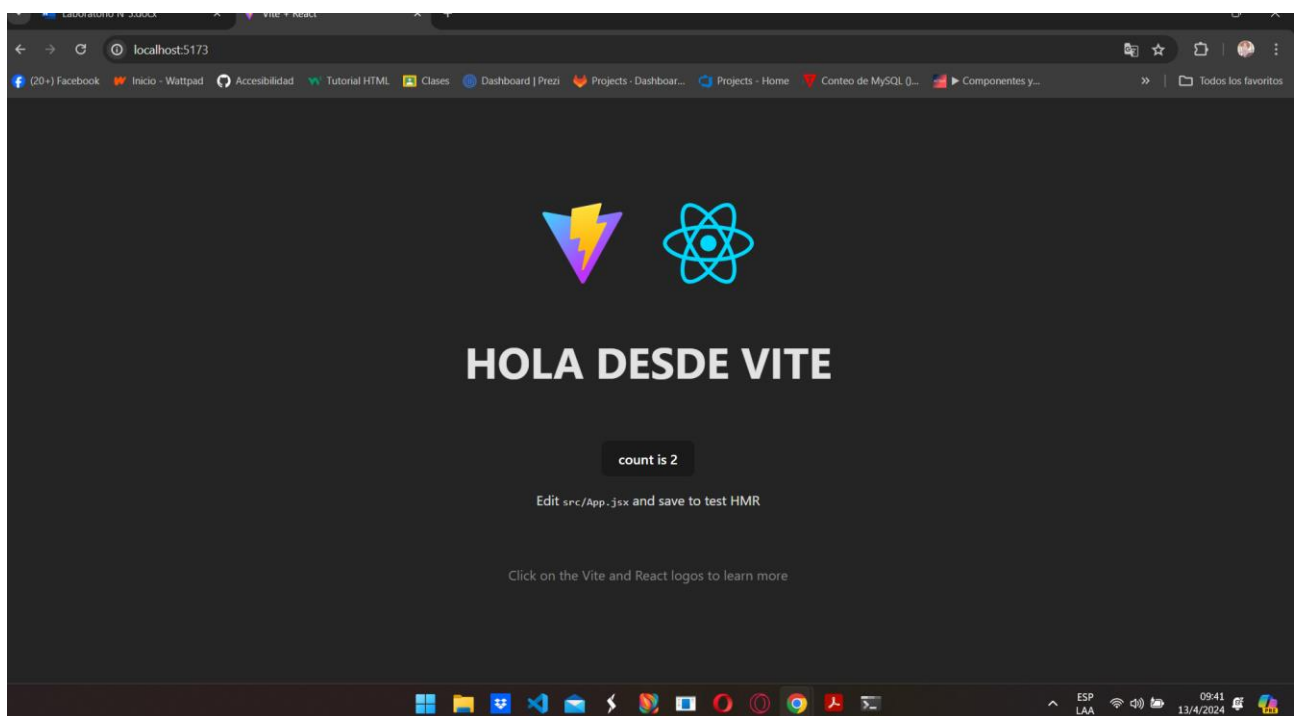
Talvez te preguntes por que iniciar un proyecto con Vite, incluso se podría decir que no se vea mas en clase o en lo que resta de la materia (posiblemente), pero siempre es bueno conocer nuevas tecnologías. Para tratar de justificar Vite un poco, tiene a su defensa que es fácil de usar y generalmente es usado para enseñar JS o Ts o tomar primeros acercamientos, hasta incluso como entorno de ejecución de Js. Si estas dispuesto a darle una oportunidad, sigamos con el lab, caso contrario también seguiremos con el lab. Bueno, veras si pulsas el Botón este incrementa en uno, y justamente hace uso de un hook para ello. Los hooks tienen que ver con el estado y ciclo de vida de la aplicación, el hook mas popular es el bien llamado “useState”. También es válido mencionar que podemos crear nuestros propios hooks, estos son llamados “Custom Hooks”. Una vez abierto el proyecto desde tu editor de código favorito veras lo siguiente:



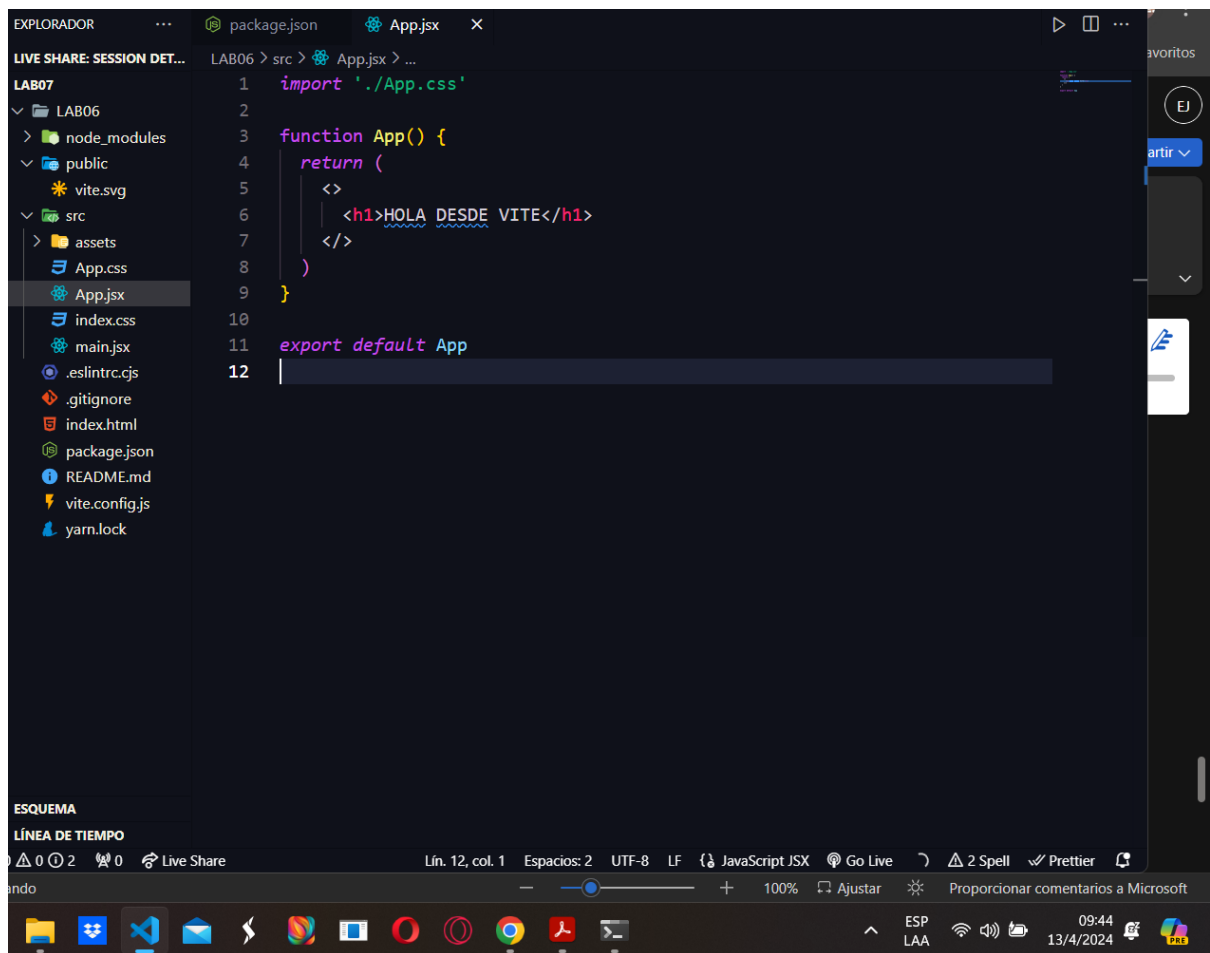
Esa es la estructura inicial que nos presenta Vite, para hacer uso de React. Tomate un tiempo para revisar todos los archivos, revísalos con mucho cuidado, bien podría ser la evaluación el explicar aquella estructura de archivos y carpetas. Aun ejecutándose el proyecto ve al archivo App.jsx.

```
... package.json App.jsx x
LAB06 > src > App.jsx > App
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7   const [count, setCount] = useState(0)
8
9   return (
10     <div>
11       <a href="https://vitejs.dev" target="_blank">
12         <img src={viteLogo} className="logo" alt="Vite logo" />
13       </a>
14       <a href="https://react.dev" target="_blank">
15         <img src={reactLogo} className="logo react" alt="React logo" />
16       </a>
17     </div>
18     <h1>HOLA DESDE VITE</h1>
19     <div className="card">
20       <button onClick={() => setCount((count) => count + 1)}>
21         count is {count}
22       </button>
23       <p>
24         Edit <code>src/App.jsx</code> and save to test HMR
25       </p>
26     </div>
27     <p className="read-the-docs">
28       Click on the Vite and React logos to learn more
29     </p>
30   )
31 }
32
```

Modifiquemos el h1 de la línea 19, coloca alguna frase y guarda el archivo. Y ve lo que paso en la página que tenías abierta.



Esta característica se llama HMR (Hot Module Replacement) esta actualiza partes específicas de la aplicación que fueron cambiadas y muestran el cambio hecho de manera inmediata. Eso es un concepto a grandes rasgos, puedes averiguar más si así lo deseas. Si deseas puedes eliminar todo y dejar solo el h1.



EXPLORADOR

LIVE SHARE: SESSION DET...

LAB07

- LAB06
 - node_modules
 - public
 - vite.svg
 - src
 - assets
 - App.css
 - App.jsx
 - index.css
 - main.jsx
 - .eslintrc.js
 - .gitignore
 - index.html
 - package.json
 - README.md
 - vite.config.js
 - yarn.lock

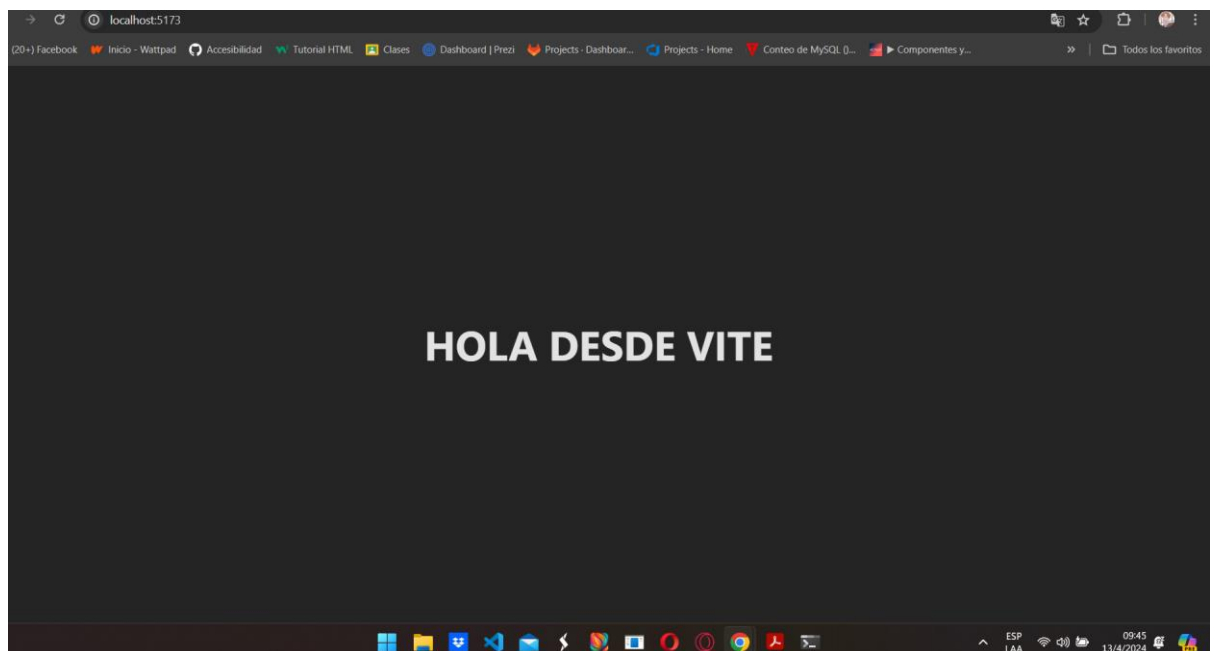
ESQUEMA

LÍNEA DE TIEMPO

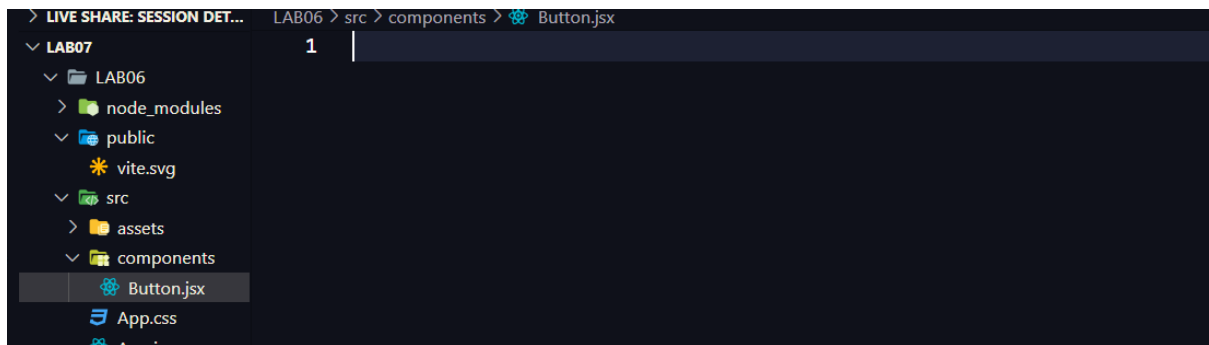
```
1 import './App.css'
2
3 function App() {
4   return (
5     <>
6     <h1>HOLA DESDE VITE</h1>
7     </>
8   )
9 }
10
11 export default App
12
```

Lin. 12, col. 1 Espacios: 2 UTF-8 LF JavaScript JSX Go Live 2 Spell Prettier

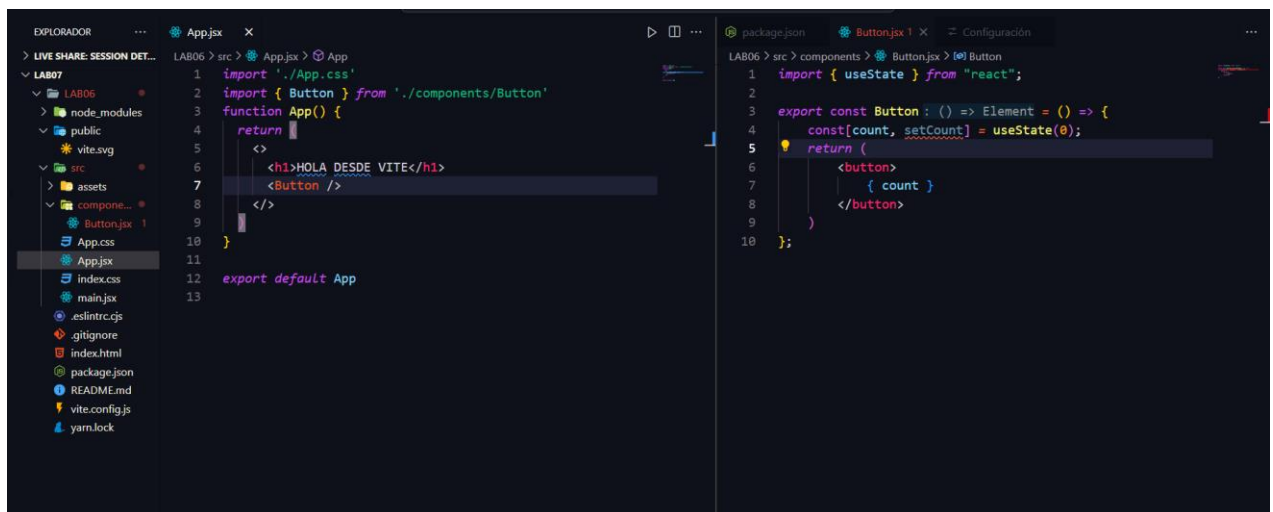
09:44 13/4/2024



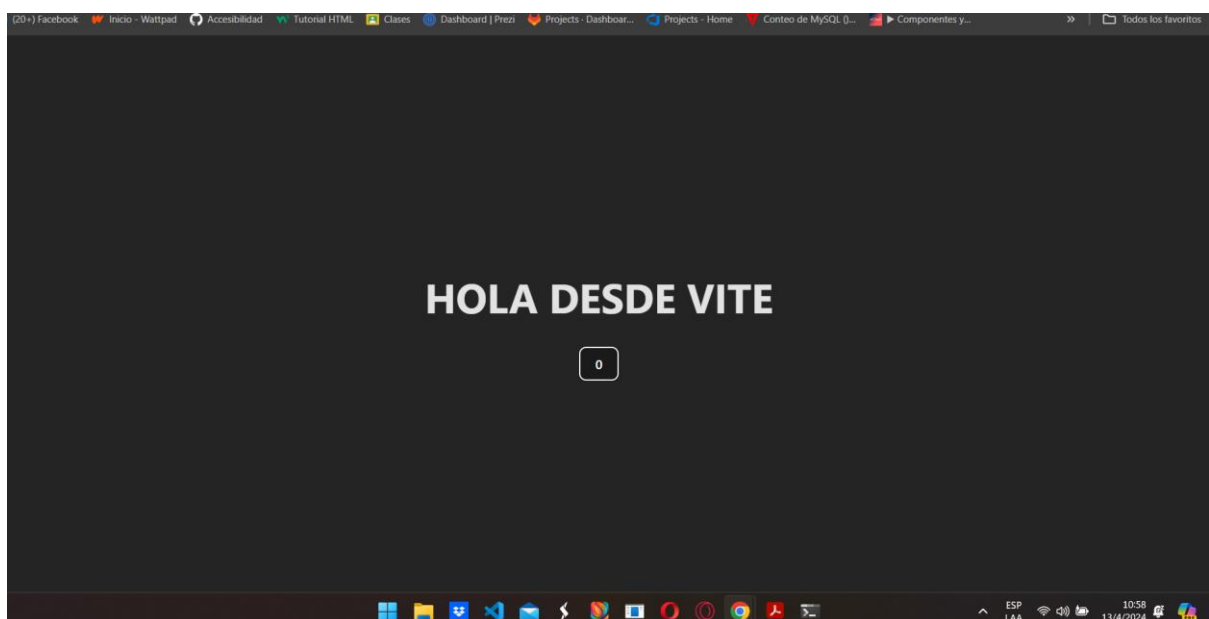
Creemos una nueva carpeta llamada components dentro de src. Dentro de la carpeta crea un archivo llamado Button.jsx.



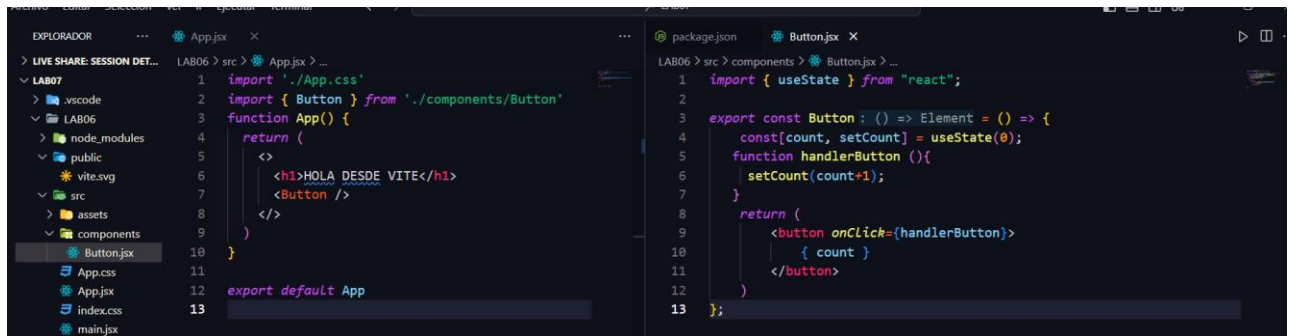
Talvez olvide mencionarlo, pero todos los componentes creados haciendo uso de React deben tener la extensión .jsx si se trabaja con JavaScript y .tsx si se trabaja con TypeScript. Dentro de Button hagamos lo siguiente:



Sin importar cuantas veces hagas click en el botón este no hará nada. Agreguémosle funcionalidad para replicar el comportamiento anterior en el que incrementaba su valor en 1. Vamos a Button.jsx y hagamos lo siguiente:



Sin importar cuantas veces hagas click en el botón este no hará nada. Agreguémosle funcionalidad para replicar el comportamiento anterior en el que incrementaba su valor en 1. Vamos a Button.jsx y hagamos lo siguiente:

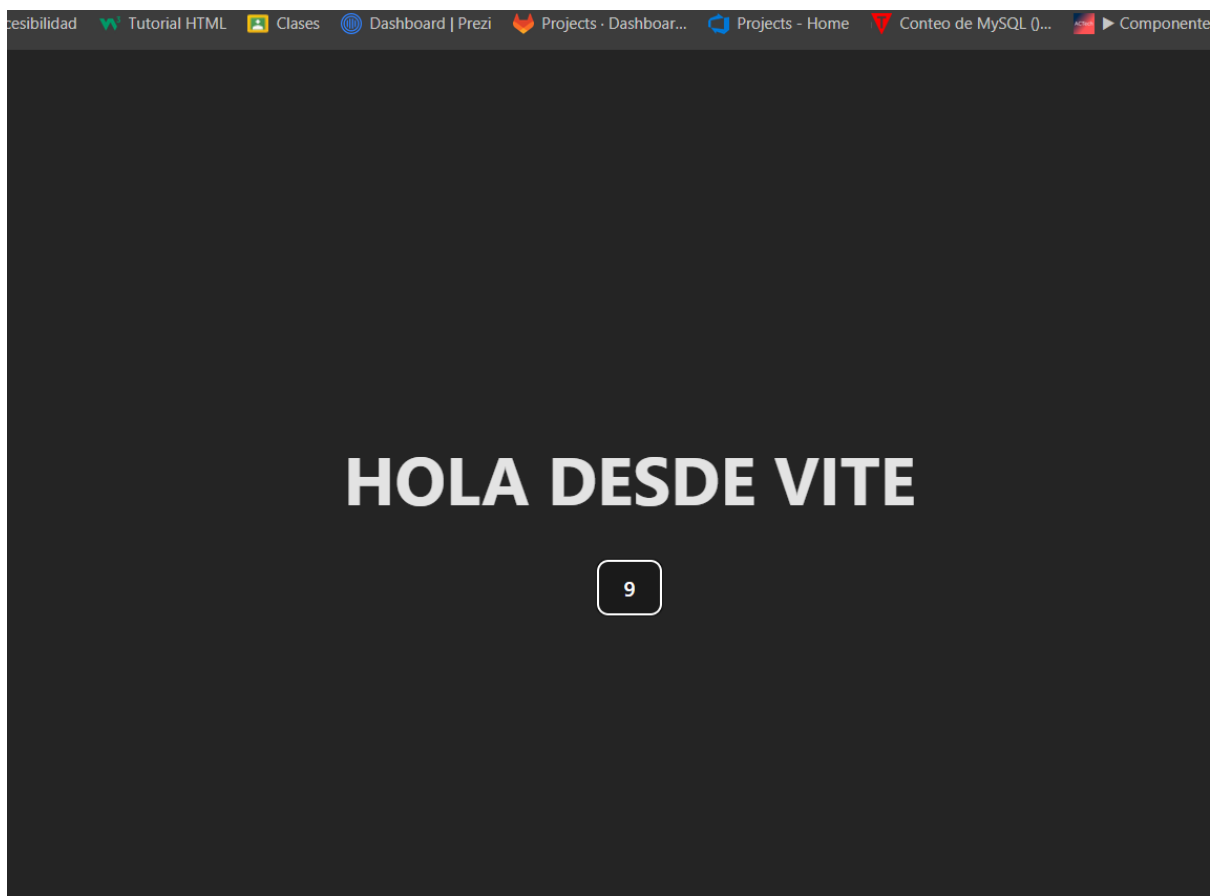


```
EXPLORADOR
... App.jsx x
> LIVE SHARE: SESSION DET... LAB06 > src > App.jsx > ...
  > LAB07
  > .vscode
  > LAB06
  > node_modules
  > public
  > vite.svg
  > src
  > assets
  > components
    > Button.jsx
    > App.css
    > App.jsx
    > index.css
    > main.jsx

1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4   return (
5     <>
6       <h1>HOLA DESDE VITE</h1>
7       <Button />
8     </>
9   )
10 }
11
12 export default App
13
```

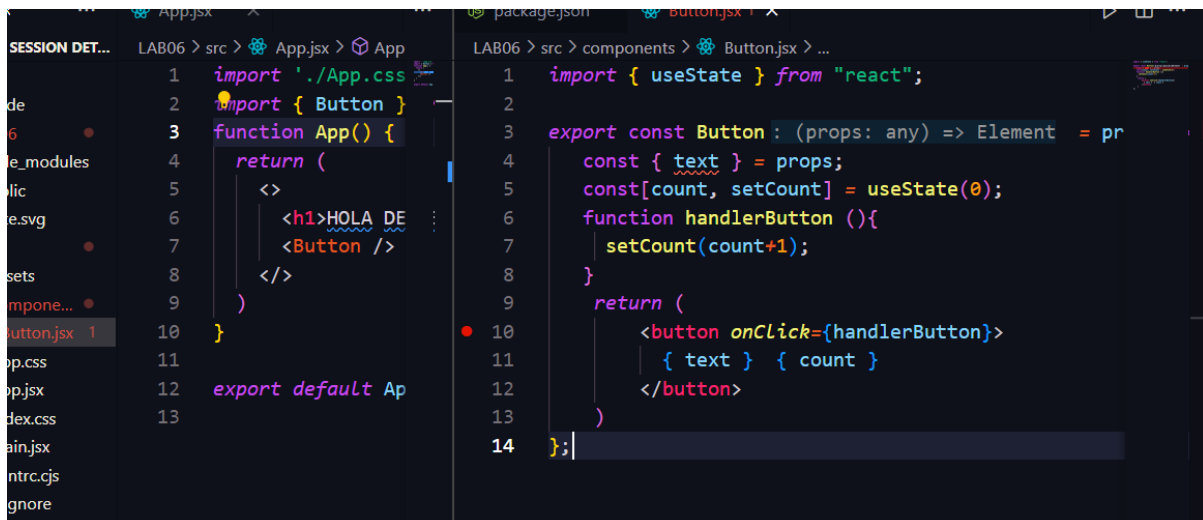
```
package.json Button.jsx x
LAB06 > src > components > Button.jsx > ...
1 import { useState } from "react";
2
3 export const Button : () => Element = () => {
4   const [count, setCount] = useState(0);
5   function handlerButton () {
6     setCount(count+1);
7   }
8   return (
9     <button onClick={handlerButton}>
10       { count }
11     </button>
12   )
13};
```

Ahora al pulsar el Botón este aumentara en una unidad al presionarlo.



Props

Para acabar con esta introducción hablemos un poco de los props. Vamos a Button.jsx y hagamos lo siguiente:



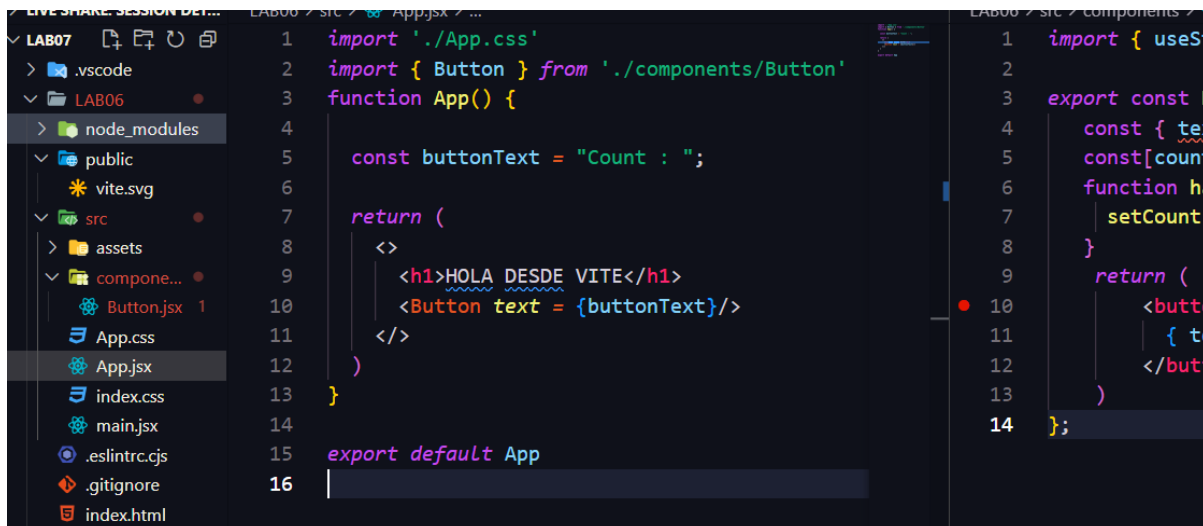
The screenshot shows two files in a VS Code editor. On the left is `App.jsx` with the following code:

```
1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4   return (
5     <>
6       <h1>HOLA DESDE VITE</h1>
7       <Button />
8     </>
9   )
10 }
11
12 export default App
```

On the right is `Button.jsx` with the following code:

```
1 import { useState } from "react";
2
3 export const Button = (props: any) => Element = props => {
4   const { text } = props;
5   const [count, setCount] = useState(0);
6   function handlerButton () {
7     setCount(count+1);
8   }
9   return (
10     <button onClick={handlerButton}>
11       { text } { count }
12     </button>
13   )
14 };
```

Y en la clase `App.jsx` realicemos los siguientes cambios:

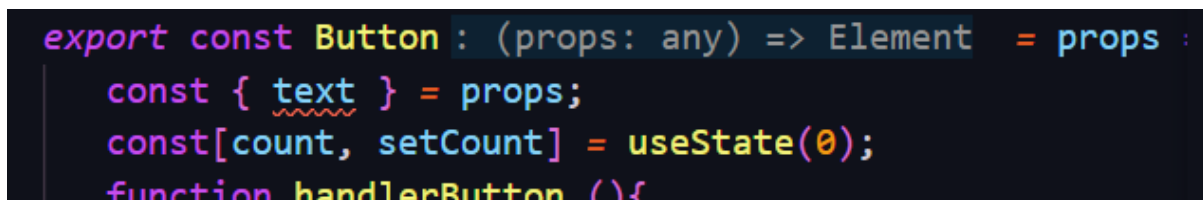


The screenshot shows the same two files after modifications. The `App.jsx` file now has:

```
1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4   const buttonText = "Count : ";
5
6   return (
7     <>
8       <h1>HOLA DESDE VITE</h1>
9       <Button text = {buttonText}/>
10     </>
11   )
12 }
13
14 export default App
```

The `Button.jsx` file remains the same as in the previous image.

Ahora expliquemos un poco acerca de lo que se hizo, los props en react son propiedades de un componente, como en este caso creamos una propiedad `text`, pero no se limita a eso, las props pueden llamarse como uno desee y pueden ser ninguna, una, dos, tres, ... etc. Esto nos sirve para pasarse al componente diferentes propiedades, como funciones, estados, etc. En la siguiente imagen se observa algo curioso:



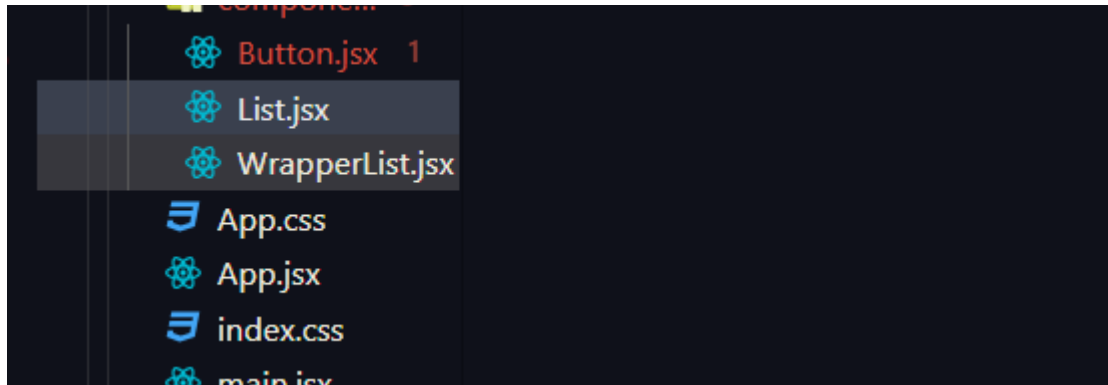
This is a close-up of the `Button.jsx` file, specifically the first three lines of the component definition:

```
export const Button = (props: any) => Element = props => {
  const { text } = props;
  const [count, setCount] = useState(0);
  function handlerButton () {
```

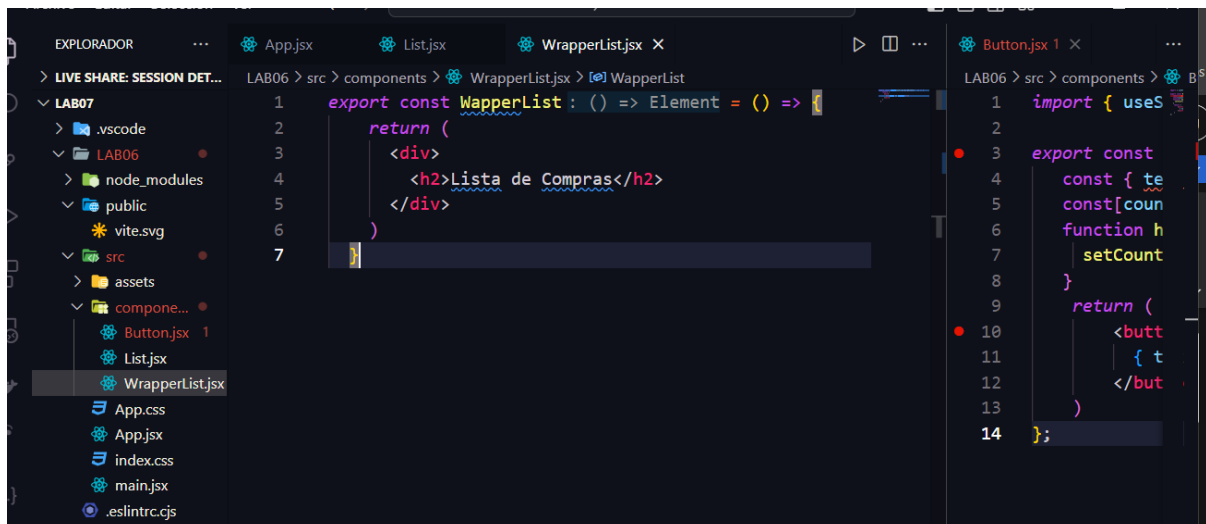
Lo que ves ahí se llama desestructuración y sirve para justamente eso, desestructurar un objeto. En este caso `props` contiene la prop `"text"` así que la extraemos de esa manera.

Childrens

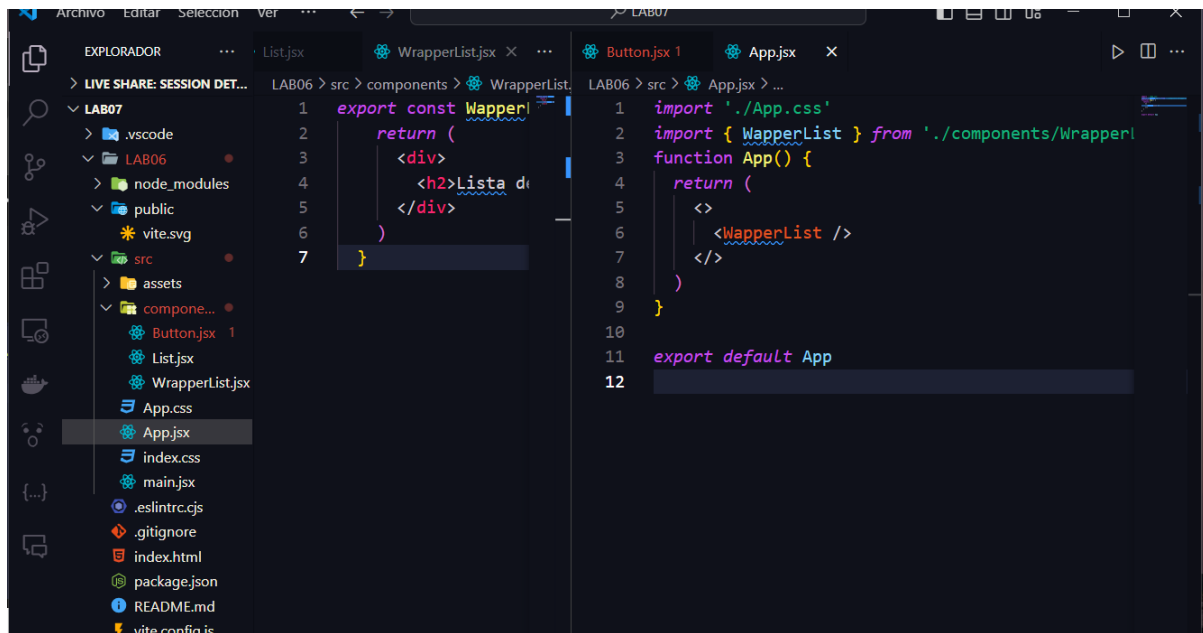
Ahora hablemos sobre los Childrens y como funcionan en React. Crea los 2 comps dentro de components



El código de WrapperLis.jsx es el siguiente:

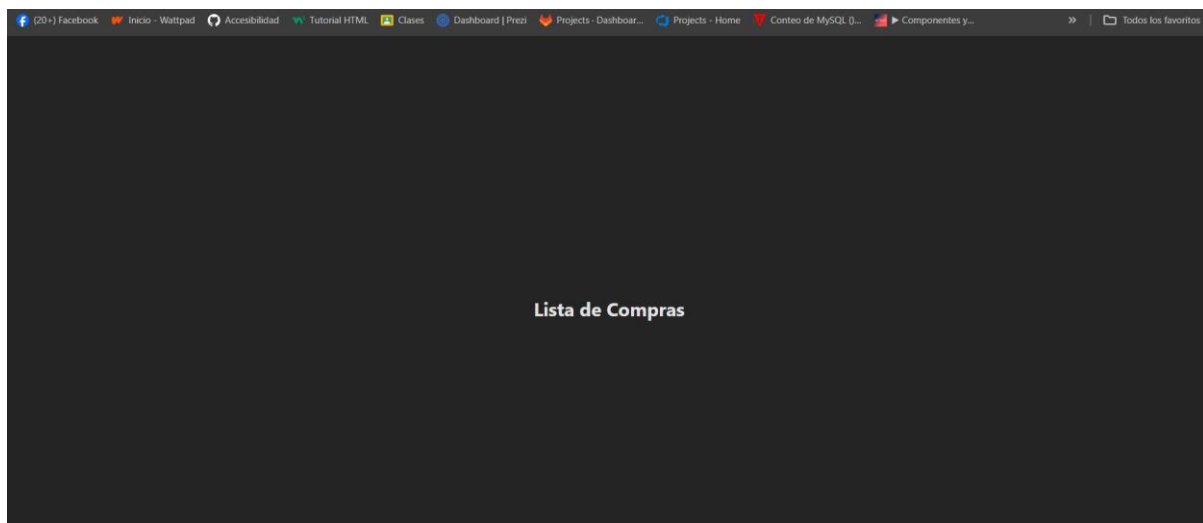


Al momento de llamarlo en App.jsx debera mostrar lo siguiente:

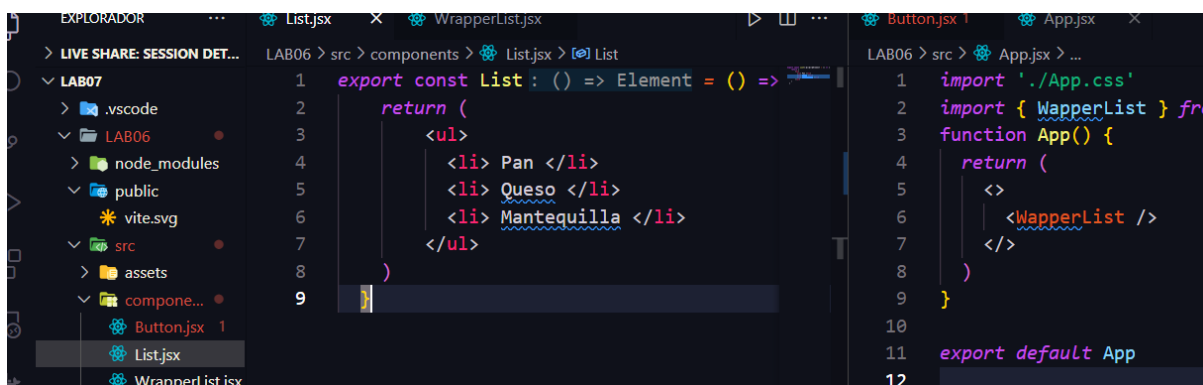


```
LAB06 > src > components > WrapperList.jsx
1 export const WrapperList = () => {
2   return (
3     <div>
4       <h2>Lista de Compras</h2>
5     </div>
6   )
7 }

LAB06 > src > App.jsx
1 import './App.css'
2 import { WrapperList } from './components/WrapperList'
3 function App() {
4   return (
5     <>
6       <WrapperList />
7     </>
8   )
9 }
10
11 export default App
12
```



Ahora en List.jsx teniendo el siguiente codigo:



```
LAB06 > src > components > List.jsx
1 export const List : () => Element = () => {
2   return (
3     <ul>
4       <li>Pan</li>
5       <li>Queso</li>
6       <li>Mantequilla</li>
7     </ul>
8   )
9 }

LAB06 > src > App.jsx
1 import './App.css'
2 import { WrapperList } from './components/WrapperList'
3 function App() {
4   return (
5     <>
6       <WrapperList />
7     </>
8   )
9 }
10
11 export default App
12
```

hacer esto:

```

function App() {
  return (
    <>
      <WrapperList />
    </>
  )
}

export default App

```

Hagamos lo siguiente:

```

return (
  <ul>
    <li>
    <li>
    <li>
  </ul>
)
}

2 import { WrapperList } from './components/WrapperList';
3 function App() {
4   return (
5     <>
6     <WrapperList>
7
8     </WrapperList>
9     </>
10  )
11 }
12
13 export default App
14

```

Y envolvamos a list dentro de wrapperList

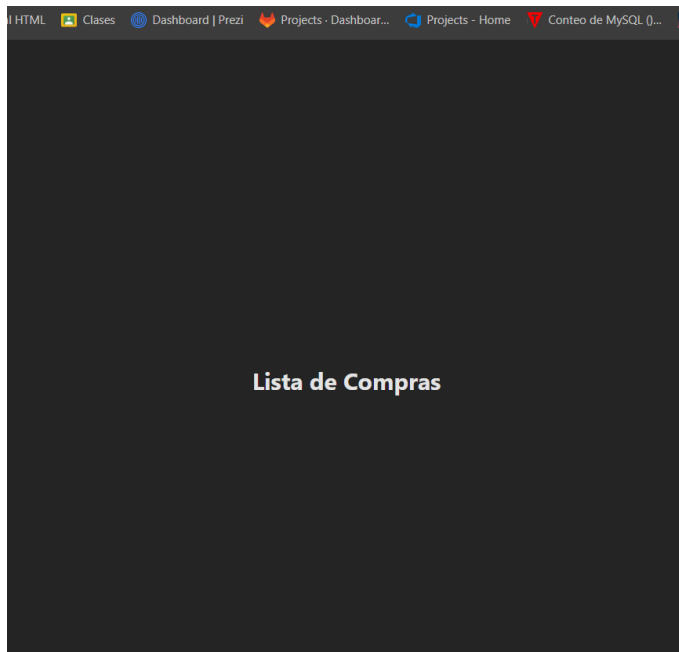
```

LAB06 > src > components > List
1 export const
2   return (
3     <ul>
4       <li>
5       <li>
6       <li>
7     </ul>
8   )
9 }

LAB06 > src > App.jsx > App
1 import './App.css'
2 import { WrapperList } from './components/WrapperList';
3 import { List } from './components/List';
4
5 function App() {
6   return (
7     <>
8       <WrapperList>
9       <List />
10     </WrapperList>
11   </>
12 )
13 }
14
15 export default App
16

```

Si guardamos el código, no muestra ningún cambio:



Eso se debe a que aun no agregamos el Children en WrapperList.

En WrapperList, ahora agrega lo siguiente:



Con eso debería mostrar lo siguiente:

Lista de Compras

- Pan
- Queso
- Mantequilla

Es necesario aclarar lo siguiente, si bien el manejo de `childrens` es muy útil, hay que saber donde utilizarlos, pues una mala implementación de este podría ser un fallo terrible al momento de desarrollar un proyecto o bien que el código aumente la complejidad cognitiva, y hasta a veces una solución simple seria mejor que una compleja con uso de `childrens`. Eso es todo en cuanto un pequeño vistazo a lo que puede realizar React. Ahora es tu turno de aplicar todo lo aprendido en la parte de evaluación.

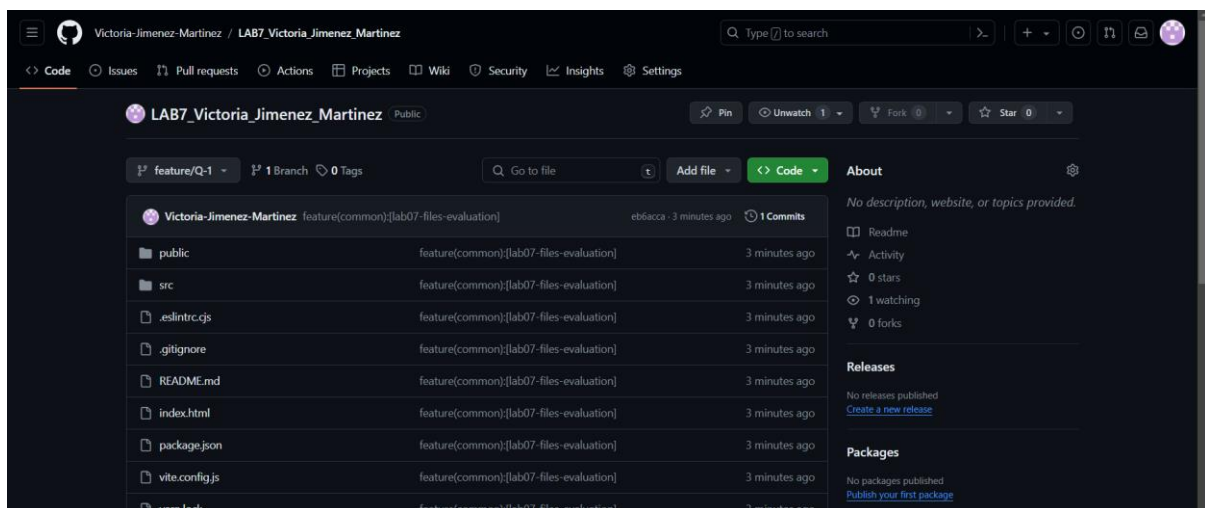
Evaluación

PREGUNTA 1

1.- Cree un nuevo repo “LAB-05” y suba todo el código realizado hasta el momento.

```
create mode 100644 src/index.css
create mode 100644 src/main.jsx
create mode 100644 vite.config.js
create mode 100644 yarn.lock

C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git push -u origin feat
ure/Q-1
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 8 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (22/22), 48.92 KiB | 6.99 MiB/s, done.
Total 22 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jim
enez_Martinez.git
* [new branch]      feature/Q-1 -> feature/Q-1
branch 'feature/Q-1' set up to track 'origin/feature/Q-1'.
```



PREGUNTA 2

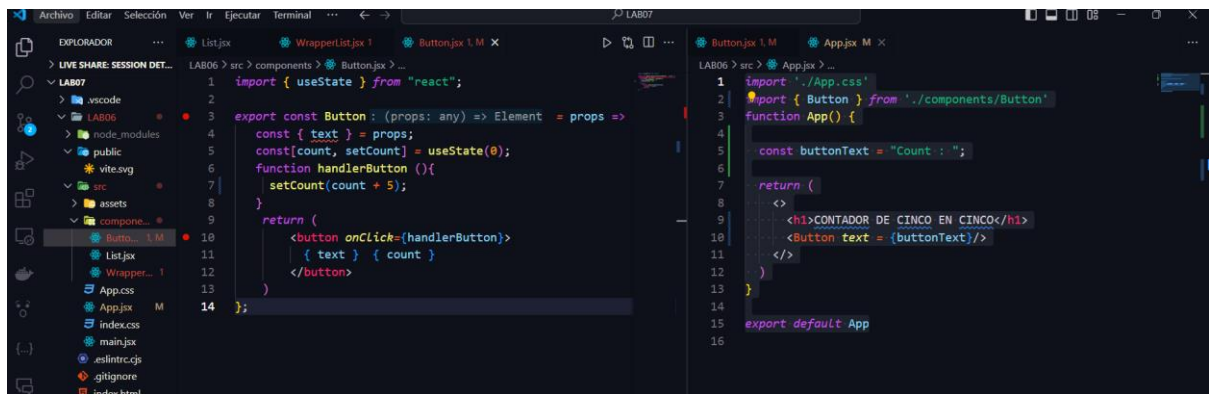
2.- Cree una nueva rama “feature/Q-2” y cambie a la nueva rama, una vez hecho eso haga que el contador incremente de 5 en 5. Suba el código, para el commit, dependerá de su criterio.

```
ite] hmr update ./src/components/WrapperList.jsx

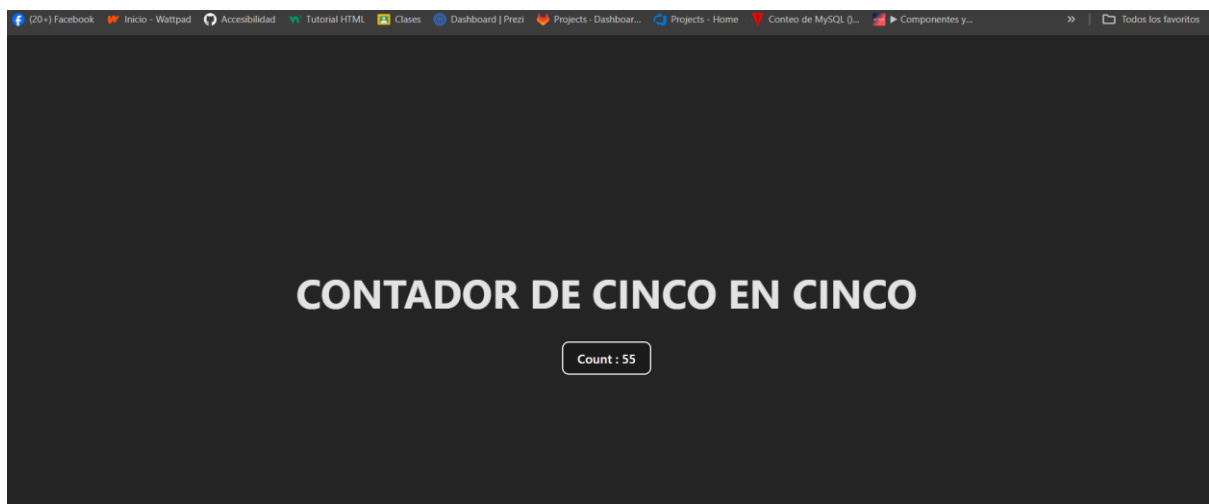
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git checkout -b feature
/Q-2
Switched to a new branch 'feature/Q-2'

C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git branch List;
feature/Q-1 import { List } from './components/List';
* feature/Q-2
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>

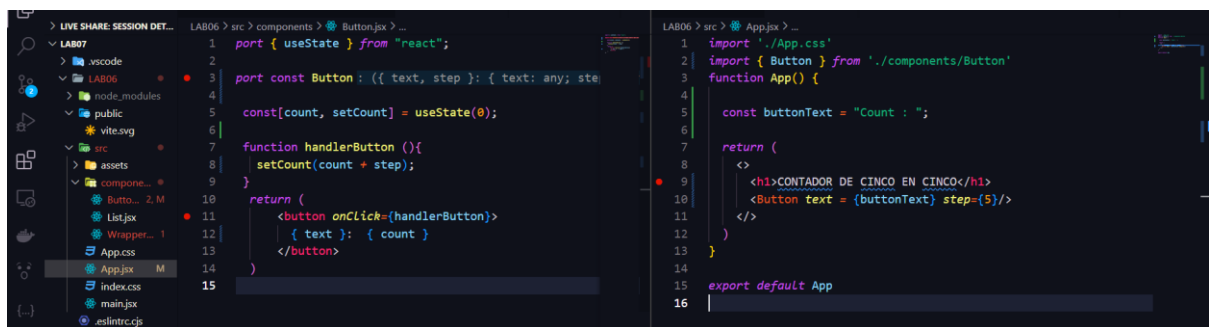
1  export const WrapperList = (props: any) => Element = props
2
3  const { children } = props
4
5  return (
6    <div>
7      <h2>Lista de Compras</h2>
8      <div>
9        { children }
10      </div>
11    </div>
12  )
```

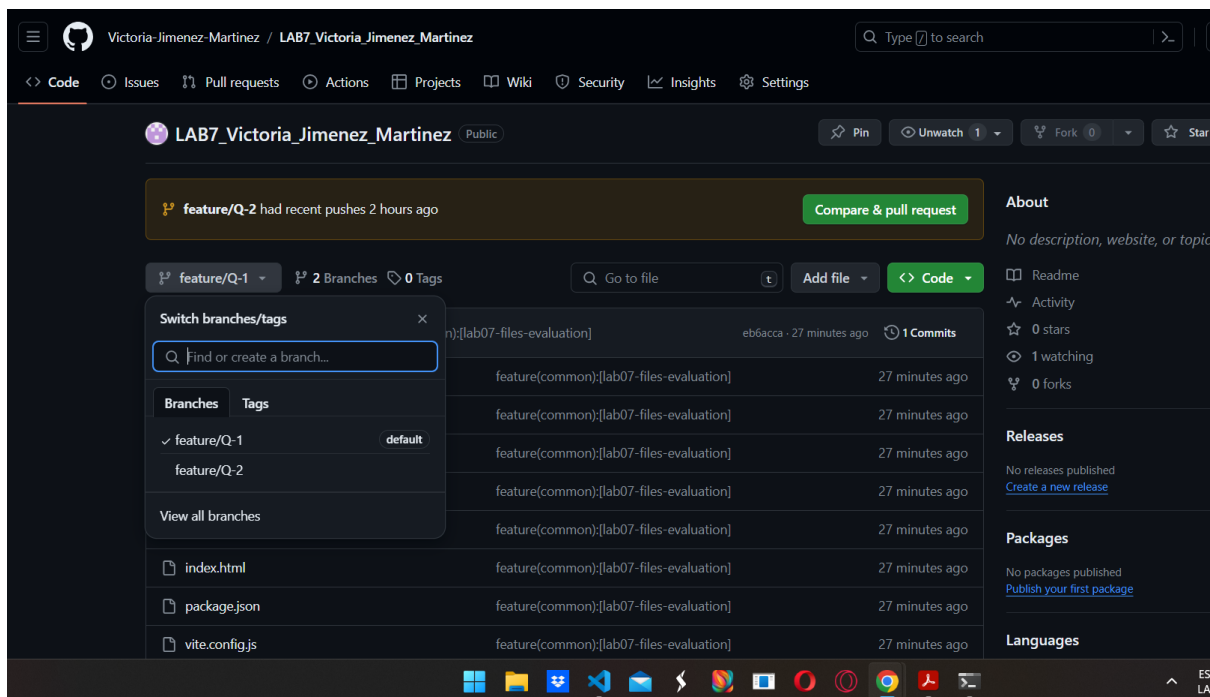
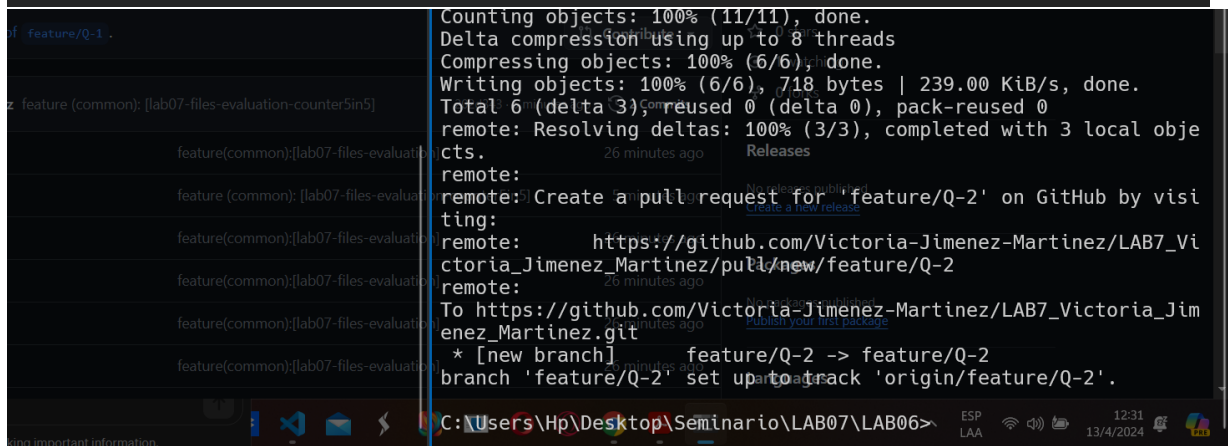
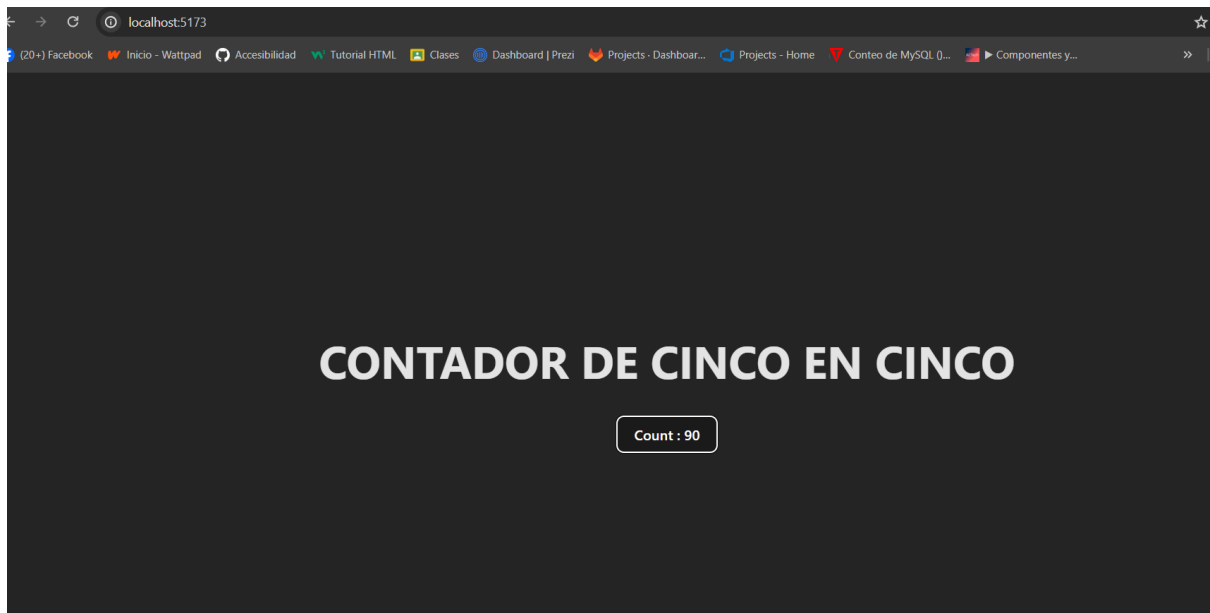
```
LAB07 > src > components > Button.js > ...
1 import { useState } from "react";
2
3 export const Button: (props: any) => Element = props =>
4
5   const { text } = props;
6   const [count, setCount] = useState(0);
7   function handlerButton () {
8     setCount(count + 5);
9   }
10
11   return (
12     <button onClick={handlerButton}>
13       { text } { count }
14     </button>
15   );
16
17
LAB06 > src > App.js > ...
1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4
5   const buttonText = "Count : ";
6
7   return (
8     <>
9       <h1>CONTADOR DE CINCO EN CINCO</h1>
10       <Button text={buttonText}/>
11     </>
12   )
13 }
14
15 export default App
16
```



OTRO METODO

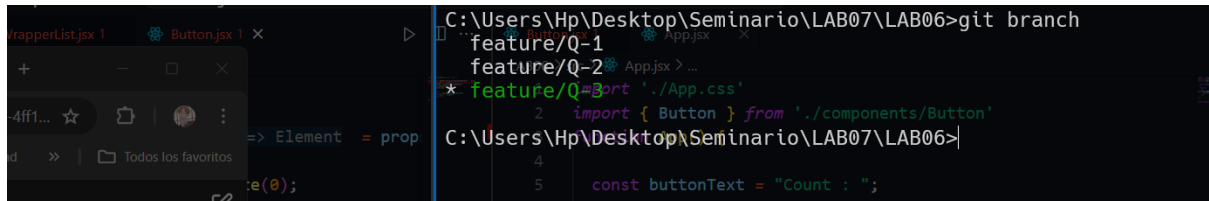


```
LAB06 > src > components > Button.js > ...
1 import { useState } from "react";
2
3 export const Button: ({ text, step }: { text: any; step: any }) => Element = ({ text, step }) =>
4
5   const [count, setCount] = useState(0);
6   function handlerButton () {
7     setCount(count + step);
8   }
9
10   return (
11     <button onClick={handlerButton}>
12       { text } { count }
13     </button>
14   );
15
16
LAB06 > src > App.js > ...
1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4
5   const buttonText = "Count : ";
6
7   return (
8     <>
9       <h1>CONTADOR DE CINCO EN CINCO</h1>
10       <Button text={buttonText} step={5}/>
11     </>
12   )
13 }
14
15 export default App
16
```



PREGUNTA 3

3.- Cree una nueva rama “feature/Q-3” y cambie a esa rama, ahora modifique el Botón para que cada vez que se haga click este muestre una alerta con un valor aleatorio entre 1 y 100. Se deberá hacer uso de **alert()** para tal efecto, si no estas familiarizado con su uso, te recomiendo investigar un poco. Suba el código.



The screenshot shows a code editor with a terminal window on the right and a code file on the left. The terminal window displays the following commands and output:

```
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git branch
feature/Q-1
feature/Q-2
* feature/Q-3
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>
```

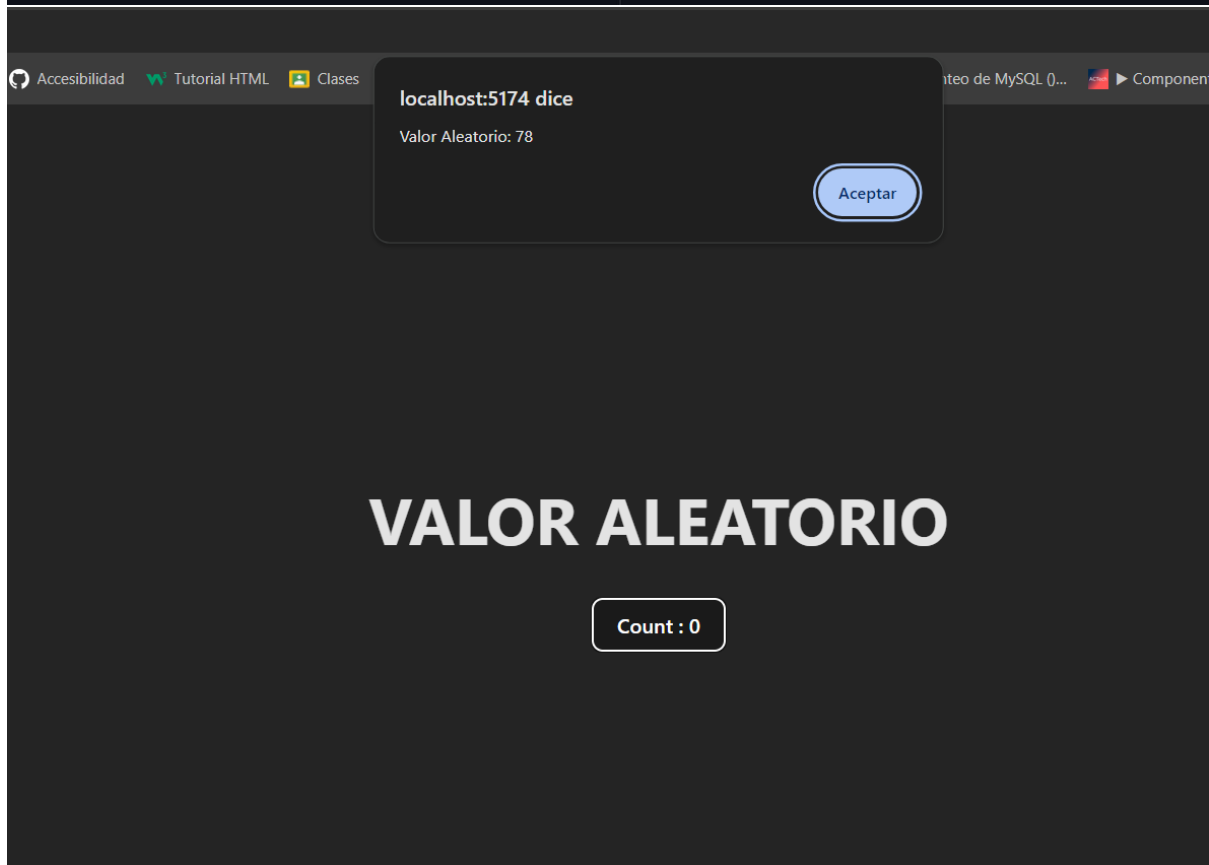
The code file on the left shows the following code:

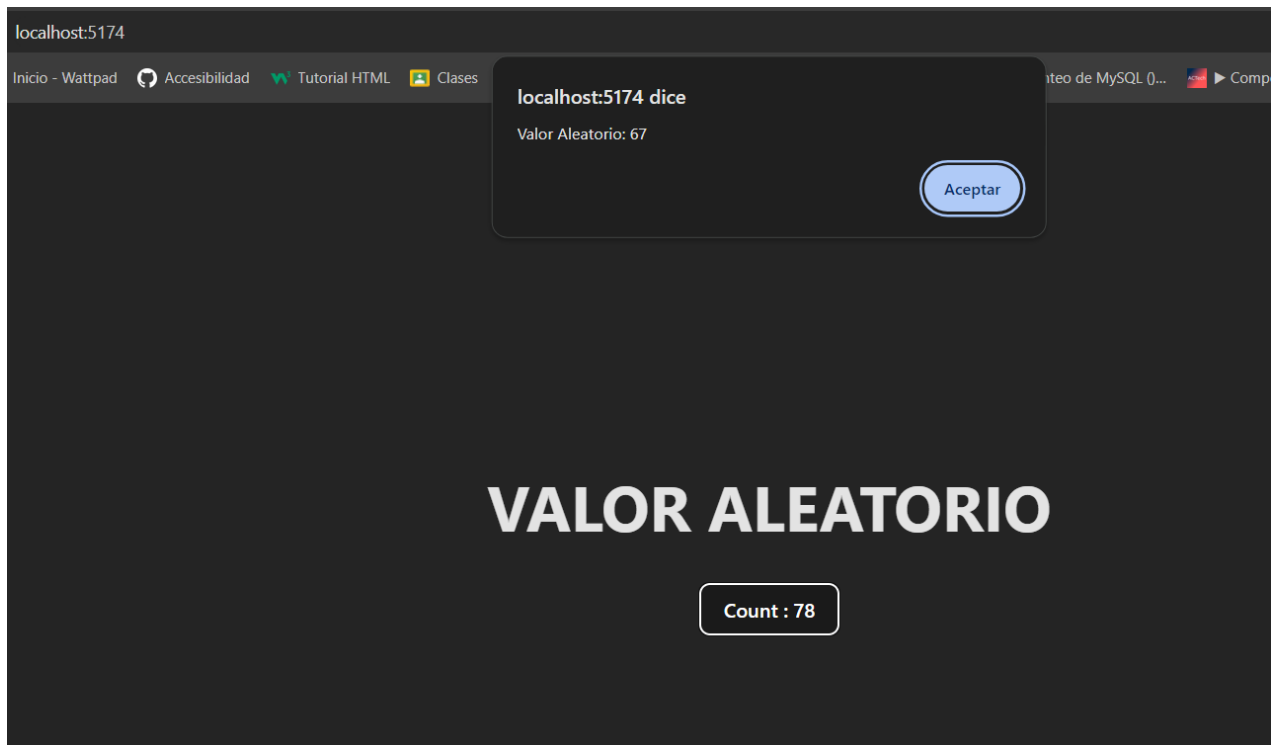
```
import './App.css'
import { Button } from './components/Button'

const buttonText = "Count : ";
```

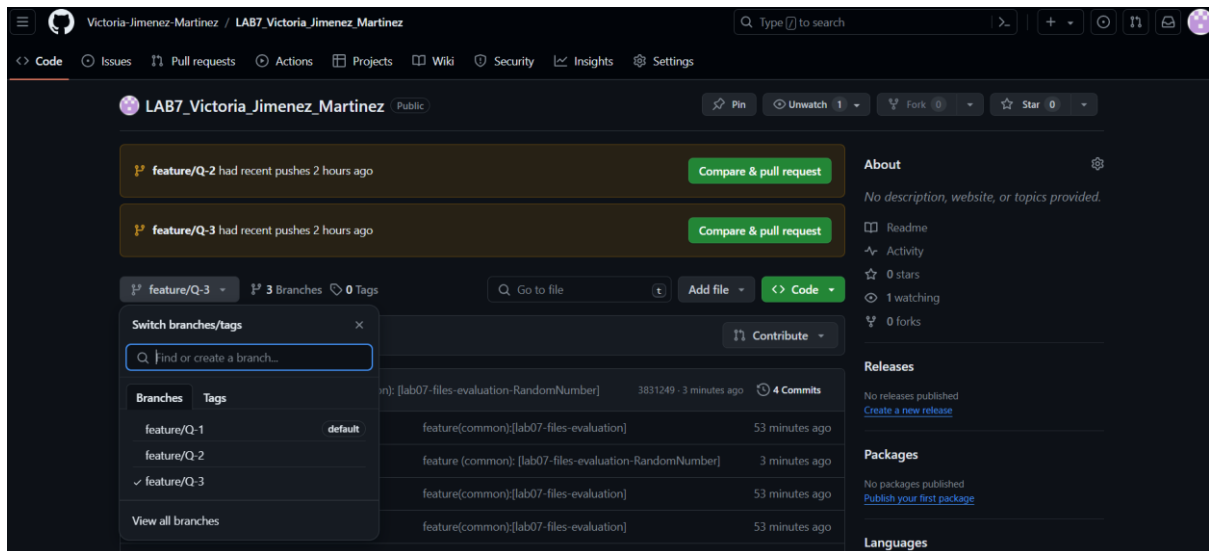
```
LAB06 > src > components > Button.jsx > ...
1 import { useState } from "react";
2
3 export const Button : (props: any) => Element = {
4   const {text} = props;
5   const [count, setCount] = useState(0);
6   function handlerButton () {
7
8     const randomNumber : number = Math.floor(Math.random() * 100);
9     alert('Valor Aleatorio: ${randomNumber}');
10
11     setCount(randomNumber)
12
13   }
14   return (
15     <button onClick={handlerButton}>
16       {text} {count}
17     </button>
18   );
19 }
```

```
LAB06 > src > App.jsx > ...
1 import './App.css'
2 import { Button } from './components/Button'
3 function App() {
4
5   const buttonText = "Count : ";
6
7   return (
8     <>
9       <h1>VALOR ALEATORIO</h1>
10       <Button text = {buttonText} />
11     </>
12   )
13 }
14
15 export default App
16
```





```
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git push origin feature/Q-3
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 839 bytes | 839.00 KiB/s, done.
Total 7 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
remote:
remote: Create a pull request for 'feature/Q-3' on GitHub by visiting:
remote:   https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez-Martinez/pull/new/feature/Q-3
remote:
To https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez-Martinez.git
 * [new branch]      feature/Q-3 -> feature/Q-3
```



PREGUNTA 4

4.- Cree una nueva rama “feature/Q-4” y cambie a esa rama, con todo lo aprendido, cree un nuevo componente, llame a este componente RandomComponent.jsx y deje volar su imaginación, bien puede devolver un simple párrafo, alguna imagen, un botón, o puede crear una opción de resetear a 0, o decrementar el valor, etc. Suba el código resultante y no olvide llamar al componente en App.jsx, para su respectiva visualización.

```

C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git checkout -b feature/Q-4
Switched to a new branch 'feature/Q-4'

C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git branch
* feature/Q-4
feature/Q-1
feature/Q-2
feature/Q-3
feature/Q-4

C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>
  
```

```

LAB06 > src > components > Button.jsx > Button
1 import { useState } from "react";
2
3 export const Button = (props: any) => Element = props => {
4   const {text} = props;
5   const [count, setCount] = useState(0);
6   const handleIncrement = () => void = () => {
7     const randomNumber : number = Math.floor(Math.random() *
8     setCount(count + randomNumber);
9   };
10
11   const handleDecrement = () => void = () => {
12     const randomNumber : number = Math.floor(Math.random() *
13     setCount(count - randomNumber);
14   };
15   return (
16     <div>
17       <button onClick={handleIncrement}>Incrementar</button>
18       <button onClick={handleDecrement}>Decrementar</button>
19       <h2>{text} {count}</h2>
20     </div>
21   );
22 }
  
```

Incrementar y Decrementar con Números Random

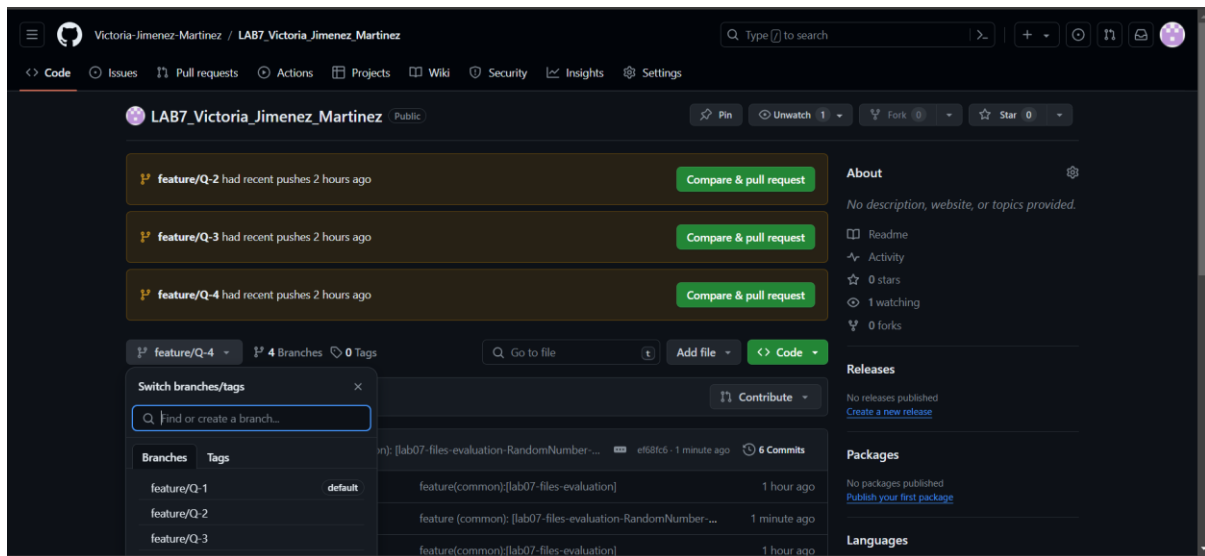
Incrementar

Decrementar

Count : 377

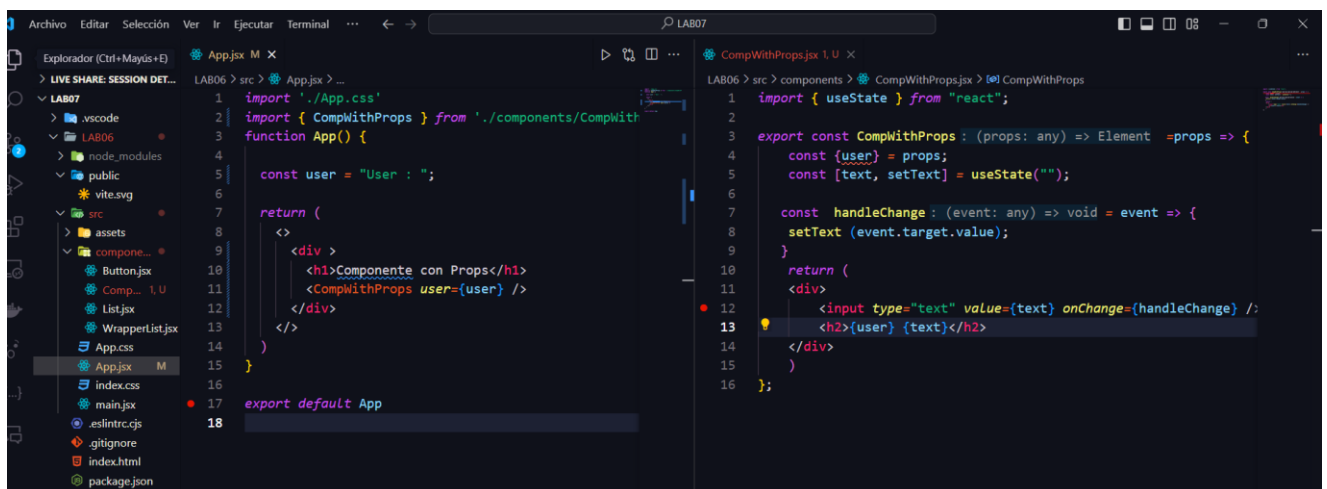
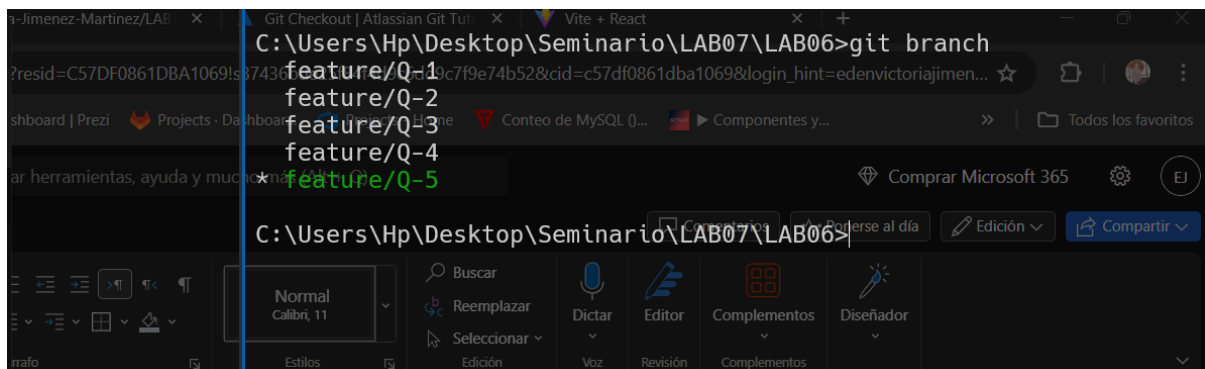
```
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git add .
warning: LF will be replaced by CRLF in src/App.jsx.
The file will have its original line endings in your working directory
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git commit -m "feature
(common): [lab07-files-evaluation-RandomNumber-DecrementIncrement
]"
[feature/Q-4 ef68fc6] feature (common): [lab07-files-evaluation-R
andomNumber-DecrementIncrement]
2 files changed, 23 insertions(+), 18 deletions(-)
rewrite src/components/Button.jsx (63%)

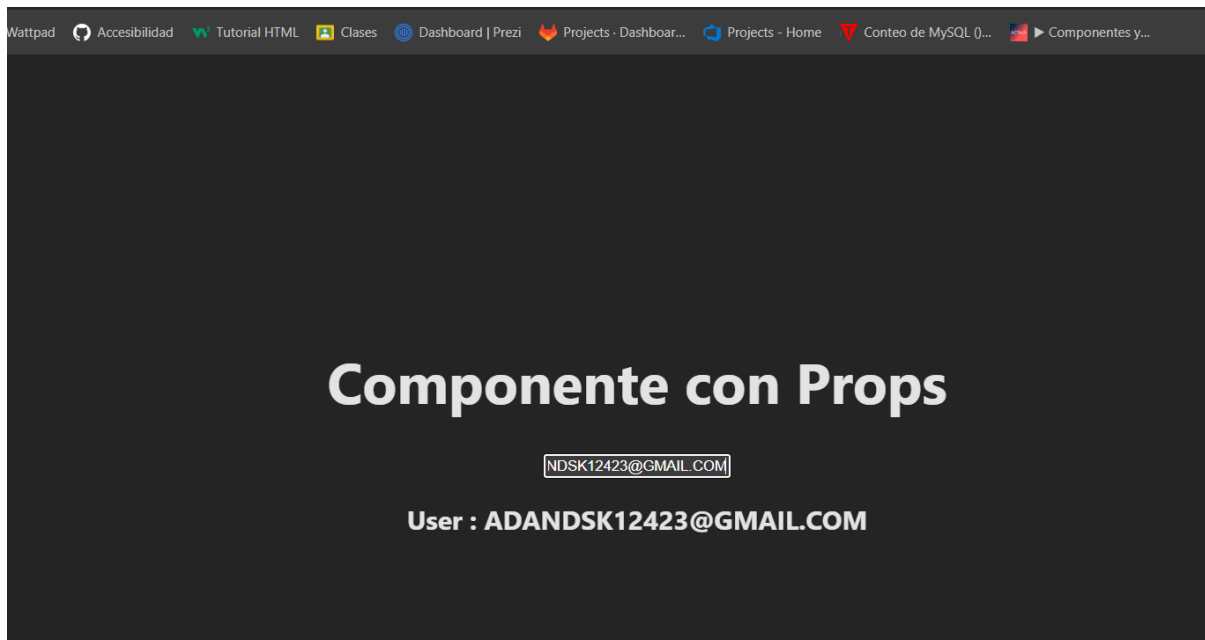
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git push origin feature
/Q-4
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 1.23 KiB | 419.00 KiB/s, done.
Total 11 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 4 local objects.
remote:
remote: Create a pull request for 'feature/Q-4' on GitHub by visiting:
remote:   https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez_Martinez/pull/new/feature/Q-4
remote:
To https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez_Martinez.git
* [new branch] feature/Q-4 -> feature/Q-4
```



PREGUNTA 5

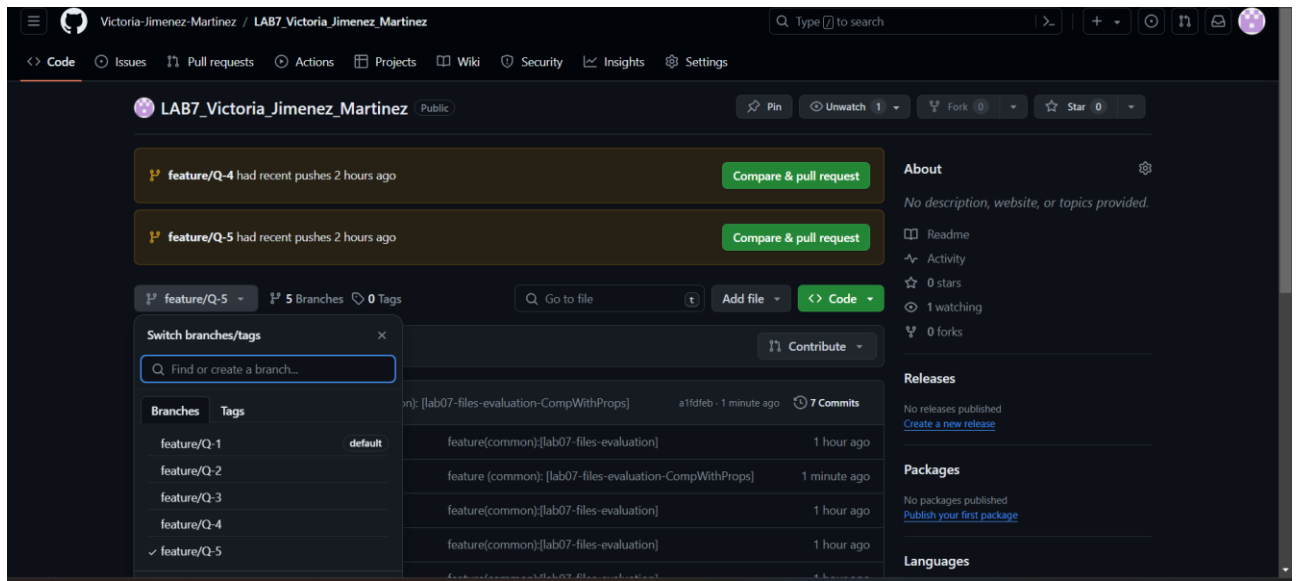
5.- Cree una nueva rama “feature/Q-5” y cambie a esa rama, juguemos un poco con las props, cree un componente llamado “CompWithProps.jsx” ese componente hará lo siguiente: tendrá 2 partes, una que es donde el texto es escrito, y otra donde vaya actualizando lo que se vaya escribiendo.



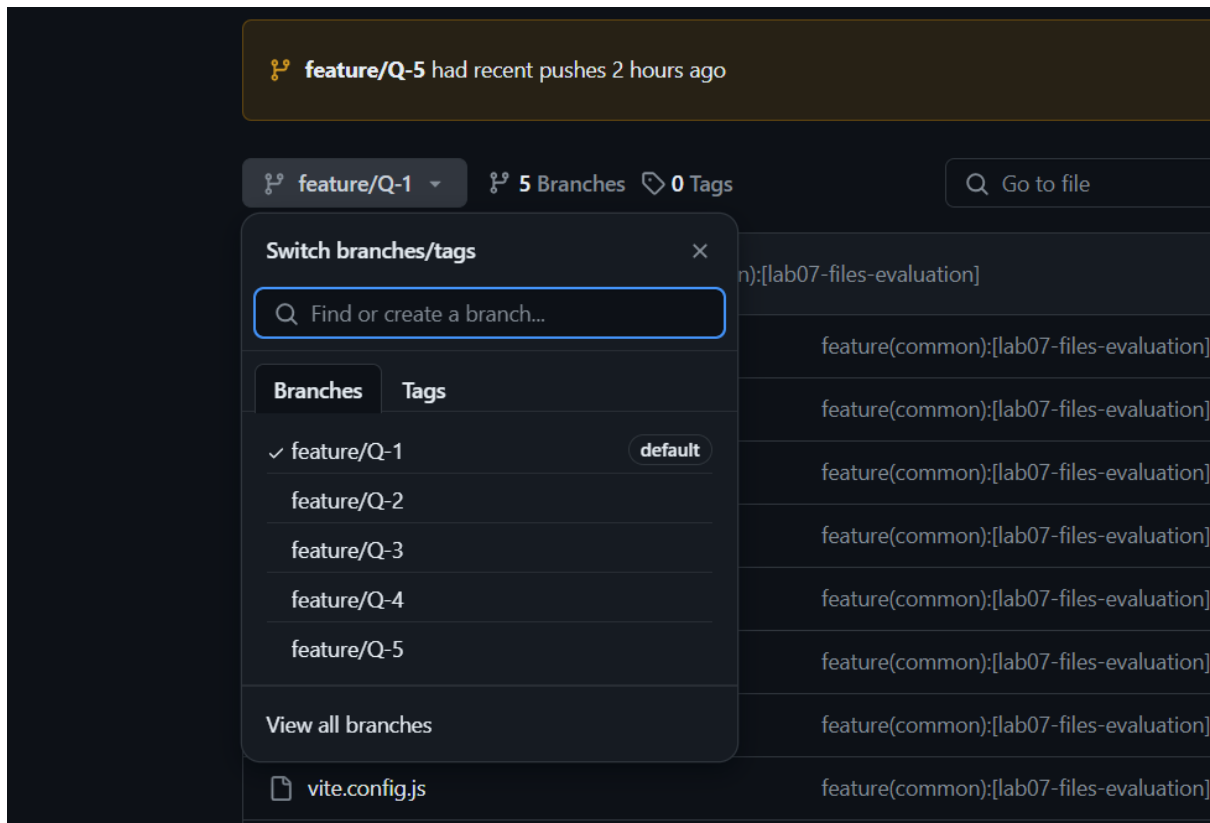


Especificando un poco mas donde va el texto es un input, mientras que la parte de abajo puede ser un p, h1, h2, ... etc. El punto es que apliques todo lo aprendido, además de revisar la documentación de como funciona un input, y como extraer su valor para mostrarlo o manipularlo. No olvides subir el código. Hasta el momento se crearon un total de 4 ramas sin contar la principal, deberás entregar un enlace de tu proyecto. Espero y te haya parecido interesante todo lo visto aquí, recuerda bien el como realizaste los componentes, y demás. Espero hayas realizado todo lo pedido, nos vemos en el siguiente lab.

```
2 files changed, 22 insertions(+), 4 deletions(-)
create mode 100644 src/components/CompWithProps.jsx
14 </div>
C:\Users\Hp\Desktop\Seminario\LAB07\LAB06>git push origin feature/Q-5
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 940 bytes | 470.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/Q-5' on GitHub by visiting:
remote:   https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez-Martinez/pull/new/feature/Q-5
remote:
To https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez-Martinez.git
* [new branch] feature/Q-5 -> feature/Q-5
```



RAMAS CREADAS EN GITHUB:



ENLACE EL REPOSITORIO:

https://github.com/Victoria-Jimenez-Martinez/LAB7_Victoria_Jimenez_Martinez.git