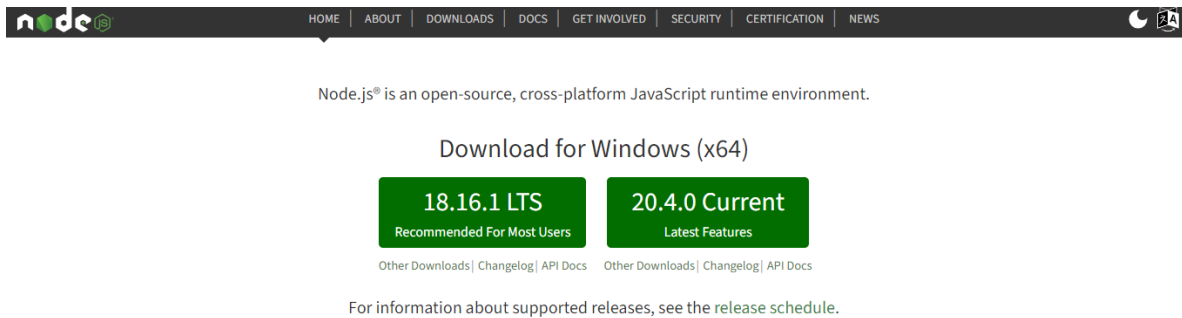
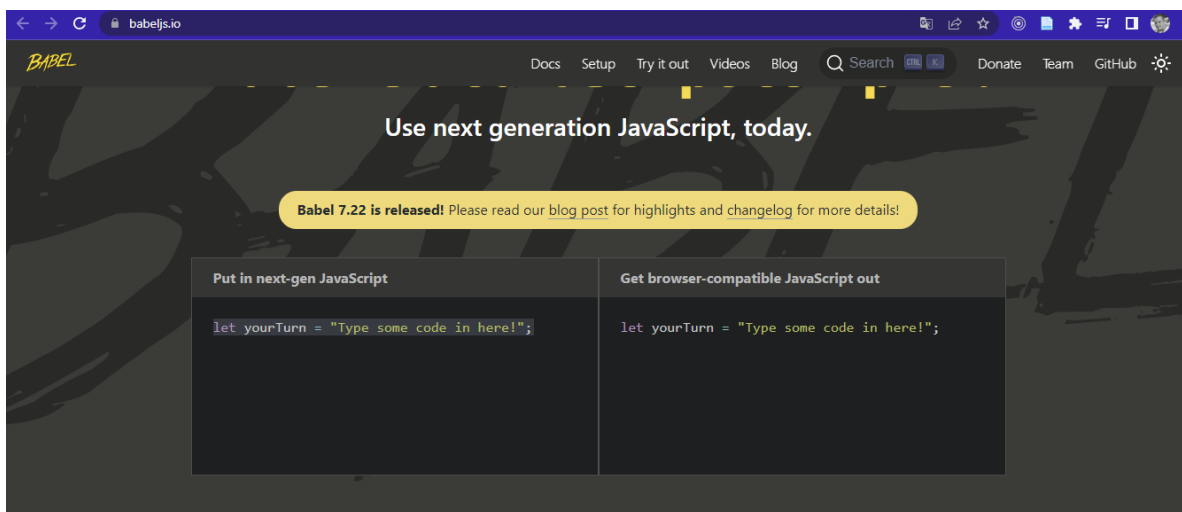


1. ACTIVIDAD 1 - BABELjs



Una vez instalado NODEJS, y siguiendo las indicaciones del instructor, ingresar al siguiente link:

<https://babeljs.io/>



1.1 Describe en las características principales de Babel.

Solución: Babel es una herramienta que nos permite transformar nuestro código JS de última generación (o con funcionalidades extras) a un código de JavaScript que cualquier navegador o versión de Node.js pueda entender.

- 1.2 Prueba el siguiente código en babeljs y en la consola del navegador, adjunta la captura de pantalla.

```
Put in next-gen JavaScript

const respApi = {
  personajes: ['goku', 'vegeta']
};

console.log(respApi.personajes.length)
```

```
> const respApi = {
  personajes: ['goku', 'vegeta']
}

console.log (respApi.personajes.length);
2
undefined
>
```

2. DESESTRUCTURACIÓN EN JAVASCRIPT

- 2.1 Para complementar los conceptos de la desestructuración, realice y conceptualice el siguiente material:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

- 2.2 Tomando como base el ejercicio anterior, construya el siguiente arreglo.

```
const persona = {
  nombre: 'Tony',
  edad: 45,
  clave: 'Iroman'
};
```

Solución:

```
> const persona = {
  nombre: 'tony',
  edad: '45',
  clave: 'jonathan becerra'
}

const nombre = [persona];

console.log (nombre);
```

```
▼ [(-)] 1
  ▶ 0: {nombre: 'tony', edad: '45', clave: 'jonathan becerra'}
      length: 1
  ▶ [[Prototype]]: Array(0)
```

undefined

2.3 En la consola del navegador utilizando la función “console.log ()” muestre los tres valores de nombre, edad y clave. Adjunte captura de pantalla. Reemplace el valor de “clave” por su primer nombre y apellido.

```
> const persona = {
  nombre: 'tony',
  edad: '45',
  clave: 'jonathan becerra'
}

const nombre = [persona];

console.log (nombre);
```

```
▼ [(-)] 1
  ▶ 0: {nombre: 'tony', edad: '45', clave: 'jonathan becerra'}
      length: 1
  ▶ [[Prototype]]: Array(0)
```

undefined

3. REACT – PRIMERA APLICACIÓN

<https://create-react-app.dev/>

<https://vitejs.dev/>

- ✓ Nuestra primera aplicación - Hola Mundo
- ✓ Exposiciones sobre los componentes

<https://drive.google.com/drive/folders/1xWMxG7mjxuwVDQi2r4dFrs1l-LXOVgWh?usp=sharing>

- ✓ Creación de componentes (Functional Components)
- ✓ Propiedades - Props
- ✓ Impresiones en el HTML
- ✓ PropTypes
- ✓ DefaultProps
- ✓ Introducción general a los Hooks
- ✓ useState

3.1 Crea en el directorio el punto de entrada a nuestra carpeta “react” utilizando “yarn create vite”, y el nombre de la aplicación: “01-counter-app-vite” (Con tecnología REACT).

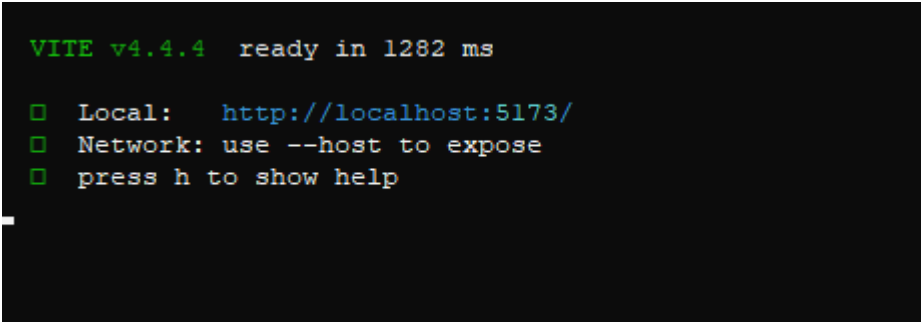
3.2 Instalar en el directorio las dependencias de yarn (yarn install).

3.3 Crear el proyecto “01-counter-app-npx”

3.4 Primera aplicación Hola mundo:

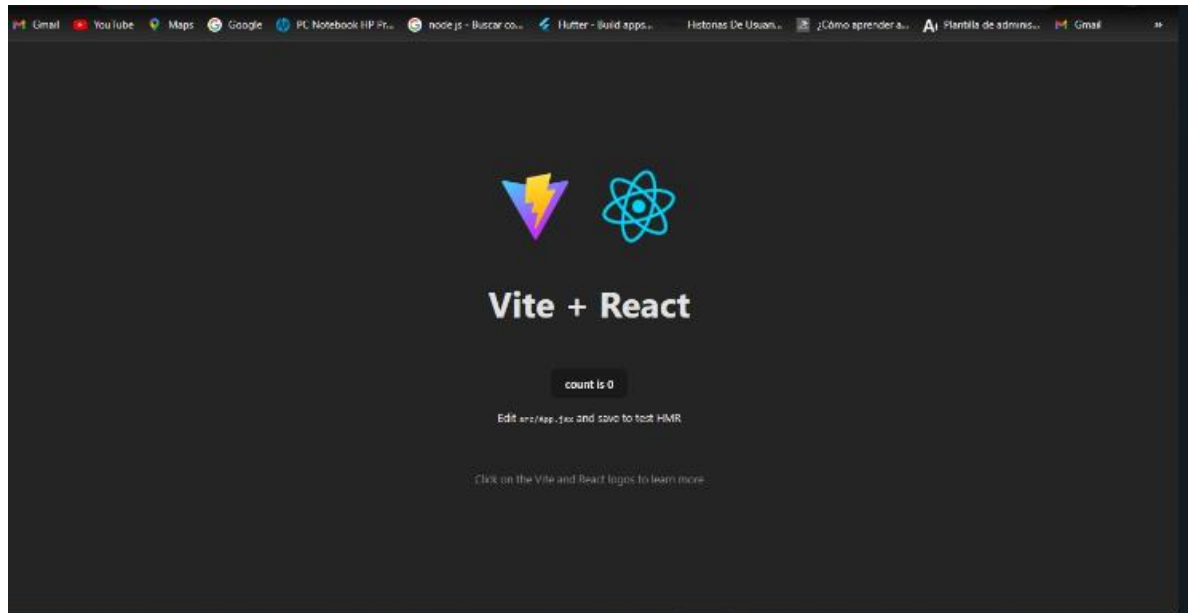
A. En el directorio **01-counter-app-vite** ejecutar el comando “yarn dev” (adjunta captura de pantalla)

B. Abrir la dirección “Local” en el navegador

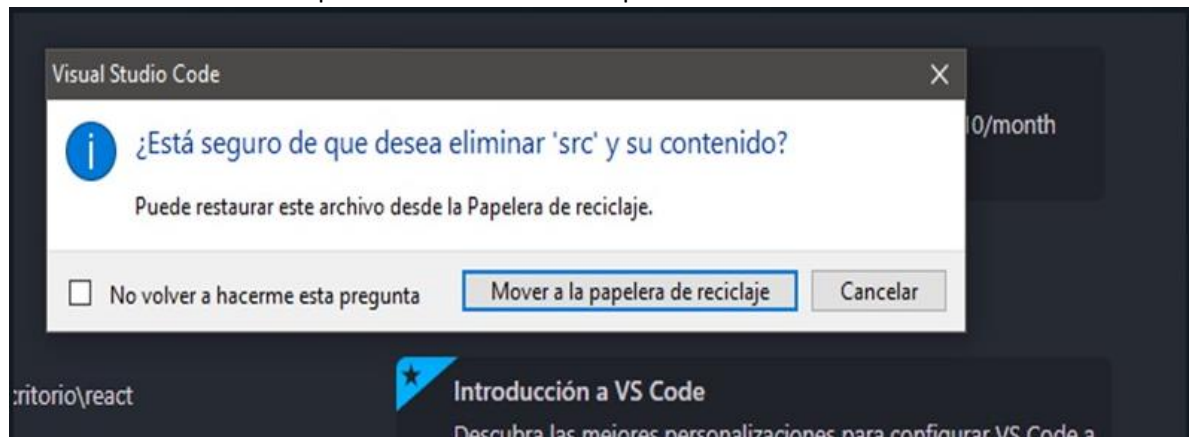


```
VITE v4.4.4 ready in 1282 ms

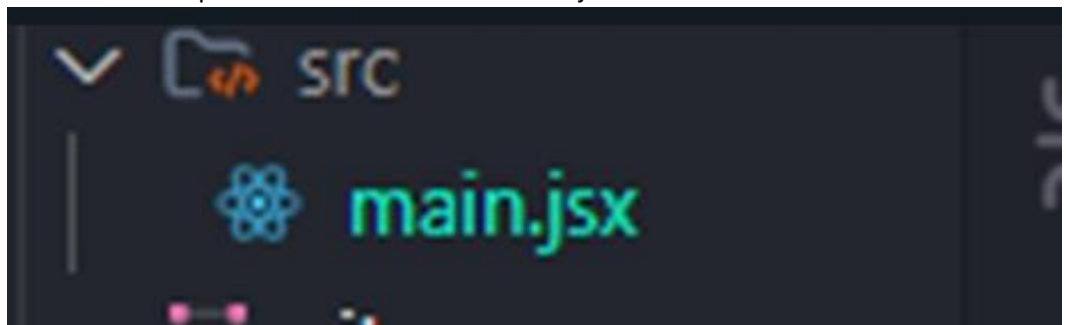
❏ Local:   http://localhost:5173/
❏ Network: use --host to expose
❏ press h to show help
```



- C. Borrar todo el contenido que se encuentra en la carpeta **SRC**



- D. Dentro de la carpeta **SRC** crear el archivo "main.jsx".



- E. Utilizando el comando `imr`:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3
4 function App() {
5
6   return (<h1>Hola Mundo</h1>);
7 }
8
9 ReactDOM.createRoot(document.getElementById('root')).render(
10   <React.StrictMode>
11     <App />
12   </React.StrictMode>
13 );
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

VITE v4.4.4 ready in 1454 ms

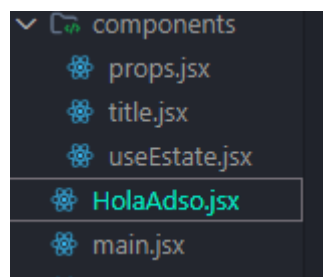
→ Local: <http://localhost:5174/>

→ Network: use --host to expose

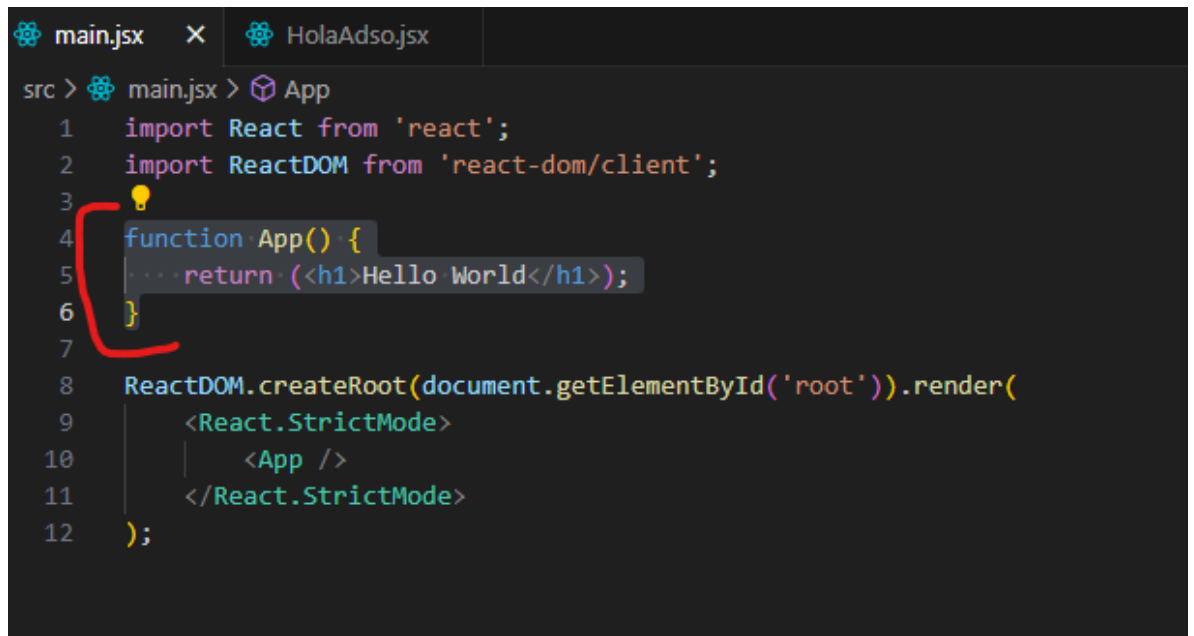
→ press h to show help

4. CREAR PRIMER COMPONENTE - FUNTIONAL COMPONENT

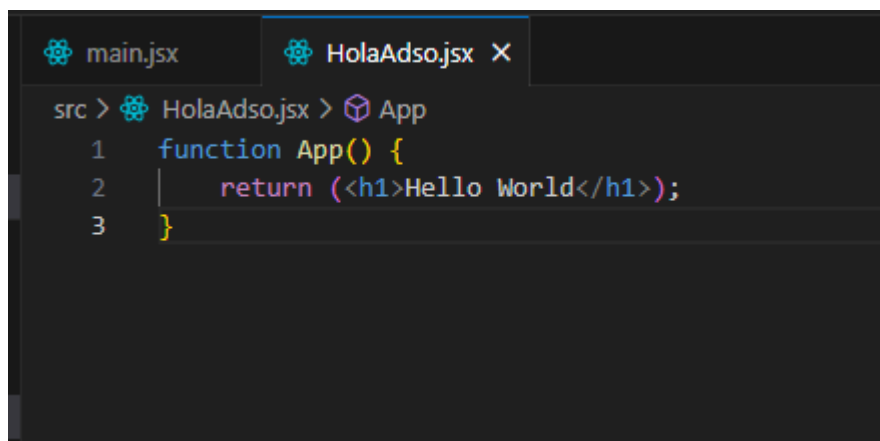
4.1 Dentro de la carpeta source (**SRC**), cree el archivo “HolaAdso.jsx”



4.2 Cortar del archivo main.jsx las líneas de código 4, 5 y 6 que se muestran en la siguiente imagen y pegarlas en el nuevo archivo “HolaAdso.jsx”

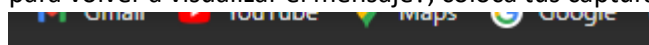


```
main.jsx x HolaAdso.jsx
src > main.jsx > App
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3
4 function App() {
5   ...return (<h1>Hello World</h1>);
6 }
7
8 ReactDOM.createRoot(document.getElementById('root')).render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
```



```
main.jsx x HolaAdso.jsx x
src > HolaAdso.jsx > App
1 function App() {
2   return (<h1>Hello World</h1>);
3 }
```

4.3 En este momento el mensaje de “Hello World” ya no se visualiza en el navegador, ¿qué líneas de código nuevas, debemos agregar en los dos archivos “main.jsx” y “HolaAdso” para volver a visualizar el mensaje?, coloca tus capturas de pantalla.



Hola Adso

- 4.4 Crear un componente en un **ARCHIVO NUEVO**, el cual debe retornar su nombre y apellido en el navegador, coloca tus capturas de pantalla.

```
import React from "react";

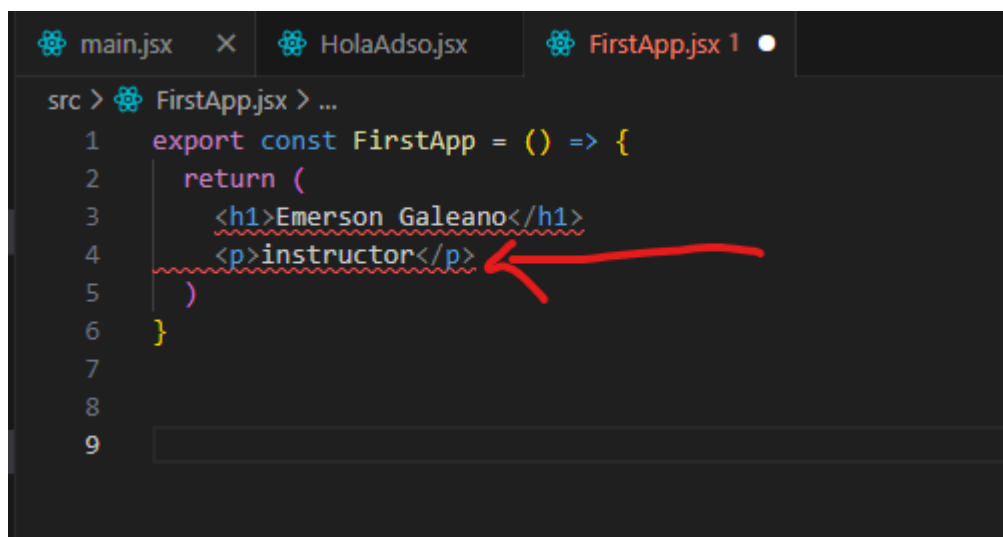
export default class Title extends React.Component{

  render() {
    return (
      <>
        <h1>Jonathan Andres Becerra Sanchez</h1>
        <p>Aprendiz</p>
      </>
    );
  }
}
```

Jonathan Andres Becerra Sanchez

Aprendiz

- 4.5 Al querer colocar un texto en la etiqueta <p>, de la forma en que se muestra en la siguiente imagen, se genera un error, tanto en el editor como en el navegador.



The screenshot shows a code editor with three tabs: main.jsx, HolaAdso.jsx, and FirstApp.jsx 1. The active tab is FirstApp.jsx 1, showing the following code:

```
src > FirstApp.jsx > ...
1  export const FirstApp = () => {
2    return (
3      <h1>Emerson Galeano</h1>
4      <p>instructor</p>
5    )
6  }
7
8
9
```

A red arrow points to the closing tag </p> on line 4, which is underlined with a red wavy line, indicating a syntax error. The error is due to the missing closing tag for the <p> element.

- ¿Por qué se da este error?
Solución: por que React se maneja por componentes y si no esta agrupado puede romper o fracturar en su sintaxis a React
- ¿Cómo lo corriges? (coloca captura de pantalla de la solución)
Solución: colocando frafment o con solo los signos de mayor y menor que vacíos

```
import React from "react";
import { Fragment } from "react";

export default class Title extends React.Component{

  render() {
    return (
      <Fragment>
        <h1>Jonathan Andres Becerra Sanchez</h1>
        <p>Aprendiz</p>
      </Fragment>
    );
  }
}
```

```
import React from "react";
// import { Fragment } from "react";

export default class Title extends React.Component{

  render() {
    return (
      <>
        <h1>Jonathan Andres Becerra Sanchez</h1>
        <p>Aprendiz</p>
      </>
    );
  }
}
```

4.6 Realizar tutorial react 3 en línea, colocar capturas de pantalla del desarrollo y código

Solución:

<https://es.react.dev/learn/tutorial-tic-tac-toe>

Siguiente jugador: X



1. Ir al inicio del juego

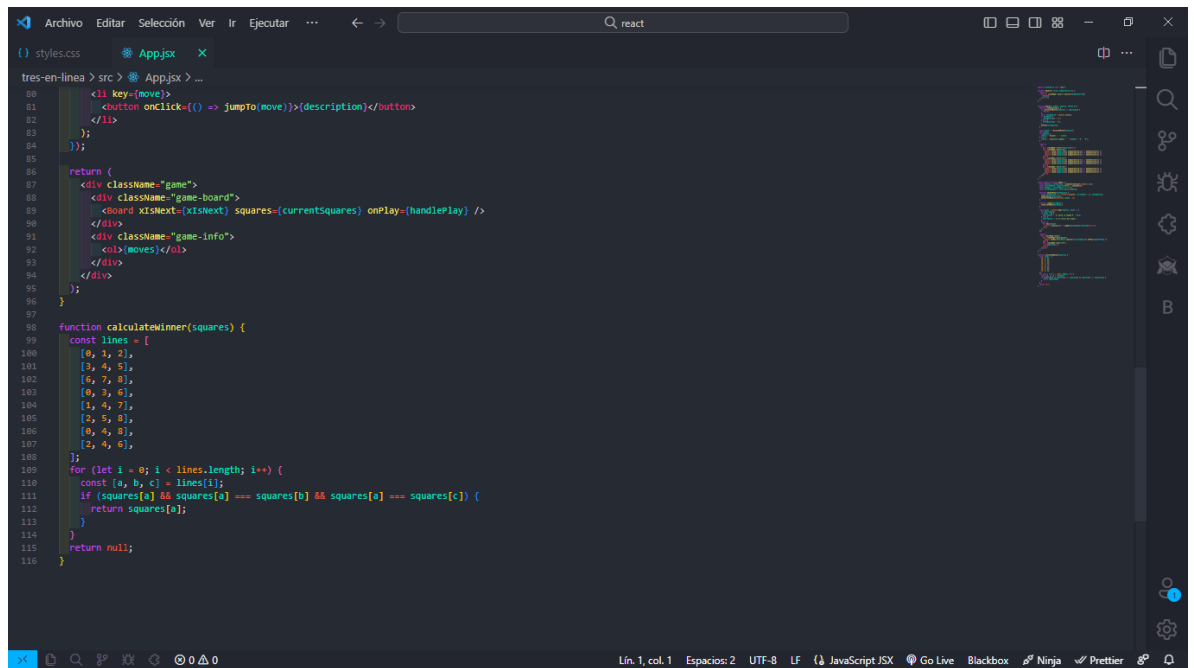
```
Archivo Editar Selección Ver Ir Ejecutar ... ← → react
```

```
( ) styles.css App.jsx X
```

```
tres-en-linea > src > App.jsx > ...
```

```
3 function Square({ value, onClick }) {
4   return (
5     <button className="square" onClick={onClick}>
6       {value}
7     </button>
8   );
9 }
10
11 function Board({ xIsNext, squares, onPlay }) {
12   function handleClick(i) {
13     if (calculateWinner(squares) || squares[i]) {
14       return;
15     }
16     const nextSquares = squares.slice();
17     if (xIsNext) {
18       nextSquares[i] = 'X';
19     } else {
20       nextSquares[i] = 'O';
21     }
22     onPlay(nextSquares);
23   }
24
25   const winner = calculateWinner(squares);
26   let status;
27   if (winner) {
28     status = 'Ganador: ' + winner;
29   } else {
30     status = 'Siguiente jugador: ' + (xIsNext ? 'X' : 'O');
31   }
32
33   return (
34     <div>
35       <div className="status">{status}</div>
36       <div className="board-row">
37         <Square value={squares[0]} onClick={() => handleClick(0)} />
38         <Square value={squares[1]} onClick={() => handleClick(1)} />
39         <Square value={squares[2]} onClick={() => handleClick(2)} />
40       </div>
41       <div className="board-row">
42         <Square value={squares[3]} onClick={() => handleClick(3)} />
43         <Square value={squares[4]} onClick={() => handleClick(4)} />
44         <Square value={squares[5]} onClick={() => handleClick(5)} />
45       </div>
46     </div>
47   );
48 }
49
50 export default function Game() {
51   const [history, setHistory] = useState([Array(9).fill(null)]);
52   const [currentMove, setCurrentMove] = useState(0);
53   const xIsNext = currentMove % 2 === 0;
54   const currentSquares = history[currentMove];
55
56   function handlePlay(nextSquares) {
57     const nextHistory = [...history.slice(0, currentMove + 1), nextSquares];
58     setHistory(nextHistory);
59     setCurrentMove(nextHistory.length - 1);
60   }
61
62   function jumpTo(nextMove) {
63     setCurrentMove(nextMove);
64   }
65
66   const moves = history.map((squares, move) => {
67     let description;
68     if (move > 0) {
69       description = 'Ir hacia la jugada #' + move;
70     } else {
71       description = 'Ir al inicio del juego';
72     }
73     return (
74       <li key={move}>
75         <button onClick={() => jumpTo(move)}>{description}</button>
76       </li>
77     );
78   });
79
80   return (
81     <div className="game">
82       <div className="game-board">
83         <Board xIsNext={xIsNext} squares={currentSquares} onPlay={handlePlay} />
84       </div>
85     </div>
86   );
87 }
```

```
Lin. 1, col. 1 Espacios: 2 UTF-8 LF JavaScript JSX Go Live Blackbox Ninja Prettier
```



```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  ...  ←  →  react
() styles.css  App.jsx  x
tres-en-linea > src > App.jsx > ...
80   <li key={move}>
81     <button onClick={() => jumpTo(move)}><description></button>
82   </li>
83 </>
84 </>
85
86   return (
87     <div className="game">
88       <div className="game-board">
89         <Board xisNext={xisNext} squares={currentSquares} onPlay={handlePlay} />
90       </div>
91       <div className="game-info">
92         <ol><moves></ol>
93       </div>
94     </div>
95   );
96 }
97
98 function calculateWinner(squares) {
99   const lines = [
100     [0, 1, 2],
101     [3, 4, 5],
102     [6, 7, 8],
103     [0, 3, 6],
104     [1, 4, 7],
105     [2, 5, 8],
106     [0, 4, 8],
107     [2, 4, 6],
108   ];
109   for (let i = 0; i < lines.length; i++) {
110     const [a, b, c] = lines[i];
111     if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {
112       return squares[a];
113     }
114   }
115   return null;
116 }
```

4.7 IMPRIMIR VARIABLES

4.7.1 Crea una variable por fuera del funtional component, cuyos valores sean:

- ✓ Un string
- ✓ Un numero
- ✓ Un arreglo
- ✓ Un objeto

Has que su valor se muestre en el navegador. Coloca las capturas de pantalla.
¿Todos los valores se pueden visualizar? ¿Cuál no? ¿Por qué?

hola mundo soy un string

1

12345

{"dato1":"dato1","dato2":"dato2","dato3":"dato3"}

```
const string = 'hola mundo soy un string';
const number = 1;
const arreglo = [1, 2, 3, 4, 5];
const objeto = {
  dato1: 'dato1',
  dato2: 'dato2',
  dato3: 'dato3'
};
return (
  <>
    <h2>{string}</h2>
    <h2>{number}</h2>
    <h2>{arreglo}</h2>
    <h2>{JSON.stringify(objeto)}</h2>
  </>
);
```

4.7.2 Sí obligatoriamente necesitamos imprimir el objeto (visualizarlo en el navegador)
¿Cuál sería la solución? (tip, intenta utilizar “JSON.stringify()”)


4.7.3 Crea una función llamada getResult y has que imprima el valor retornado

```
const getResult = () => {
  return 3 + 4;
}
```

8

```
const getResult = () => {
  return 4 + 4
}
return (
  <>
  <h2>{getResult()}</h2>
  </>
);
```

4.7.4 ¿Qué sucede si la función es asíncrona?

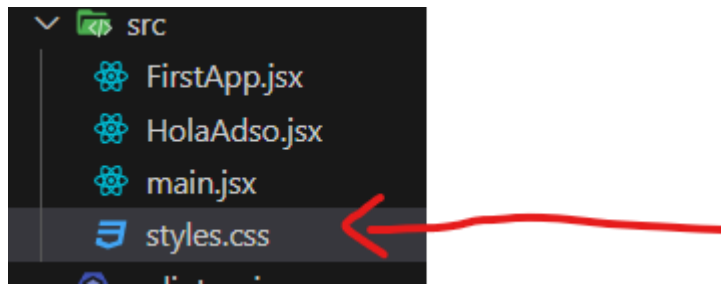


```
const getResult = async() => {
  return 4 + 4;
}
```

The image shows a code editor with the above code. Below the code editor, there is a screenshot of a web browser displaying a blank white page, indicating a runtime error. To the right of the browser, there is a screenshot of the React DevTools error boundary component stack, showing the following error:

```
Uncaught Error: Objects are not valid as a React child (found: [object Promise]). If you want to render a collection of children, use an array instead.
    at throwOnInvalidObjectType (react-dom.development.js:14887:9)
    at reconcileChildFibers (react-dom.development.js:15288:7)
    at reconcileChildren (react-dom.development.js:15174:28)
    at updateHostComponent (react-dom.development.js:14924:11)
    at beginWork (react-dom.development.js:21318:14)
    at HTMLUnknownElement.callCallback (react-dom.development.js:4164:14)
    at Object.invokeGuardedCallbackDev (react-dom.development.js:4171:16)
    at invokeGuardedCallback (react-dom.development.js:4227:13)
    at beginWork$1 (react-dom.development.js:21431:7)
    at performUnitOfWork (react-dom.development.js:20557:12)
    at workLoop (react-dom.development.js:20466:12)
    at renderRootSync (react-dom.development.js:21434:7)
    at recoverFromConcurrentError (react-dom.development.js:20458:20)
```

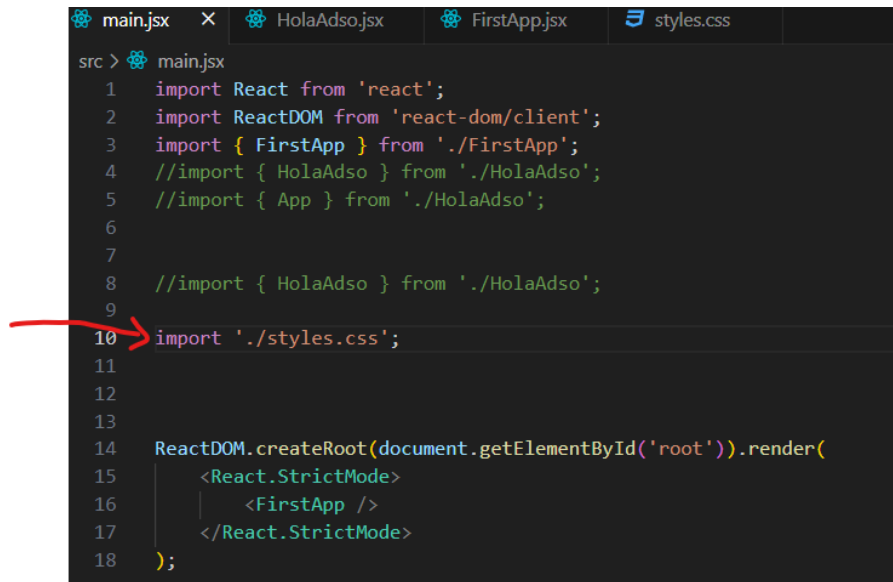
5. Dentro de la carpeta SRC crear el archivo “styles.css”



5.1 Crear los siguientes estilos básicos dentro de “styles.css”

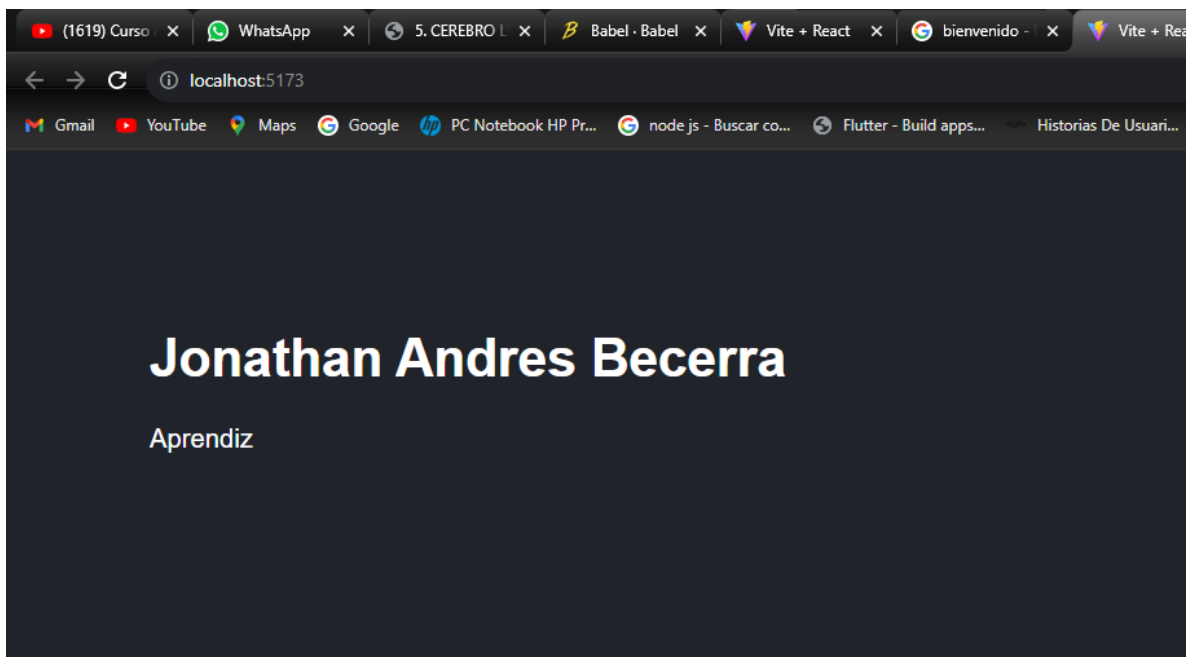
```
src > styles.css > html
1  /* CTRL + SHIFT + P */
2
3  /* Seleccionar en el cuadro de búsqueda "Sort Lines Ascending" */
4
5  html,
6  body {
7      background-color: #21232A;
8      color: white;
9      font-family: Helvetica, Arial, sans-serif;
10     font-size: 1.3 rem;
11     padding: 70px;
12
13 }
```

5.2 En el archivo “main.jsx” importe la hoja de estilos “styles.css”



```
src > main.jsx
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { FirstApp } from './FirstApp';
4  //import { HolaAdso } from './HolaAdso';
5  //import { App } from './HolaAdso';
6
7
8  //import { HolaAdso } from './HolaAdso';
9
10 import './styles.css';
11
12
13
14 ReactDOM.createRoot(document.getElementById('root')).render(
15   <React.StrictMode>
16     <FirstApp />
17   </React.StrictMode>
18 );
```

Coloca captura de pantalla, donde se debe observar tu nombre de la siguiente forma:



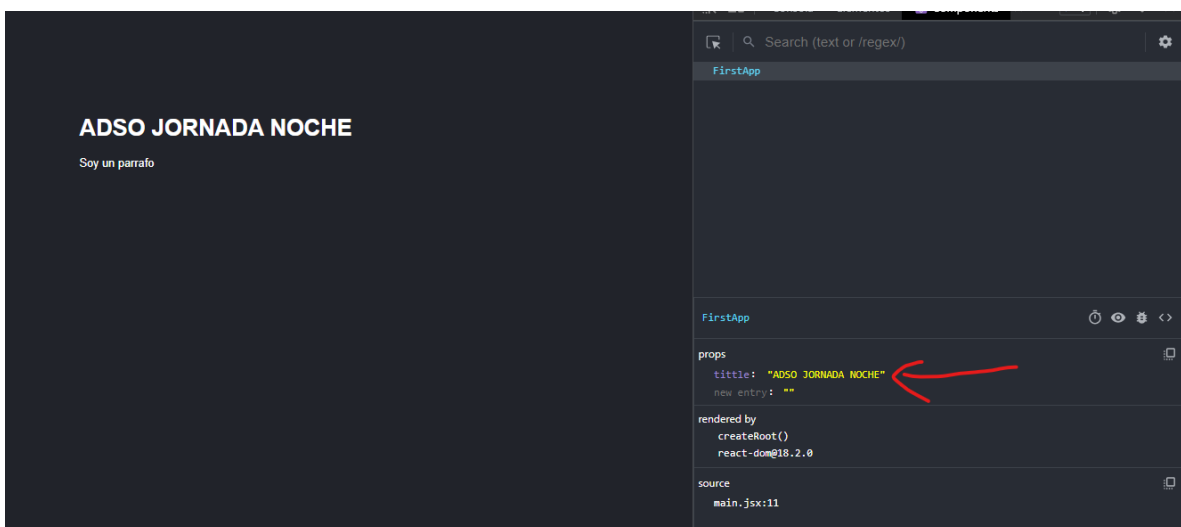
6. PROPS

- props significa propiedades.
- React Props son como argumentos de función en JavaScript y atributos en HTML. Para enviar accesorios a un componente

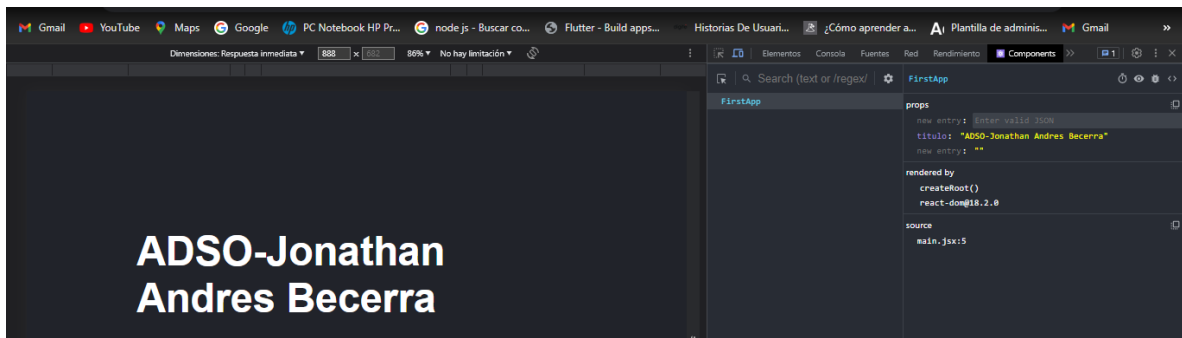
- Analiza el siguiente prop

```
export const FirstApp = ( {tittle = "El titulo esta vacio"} ) => {  
  return (  
    <>  
      <h1>{tittle}</h1>  
      <p>Soy un parrafo</p>  
    </>  
  )  
}
```

- ¿Qué nombre tiene el prop?
- Cambia su valor desde el inspector de código-componentes



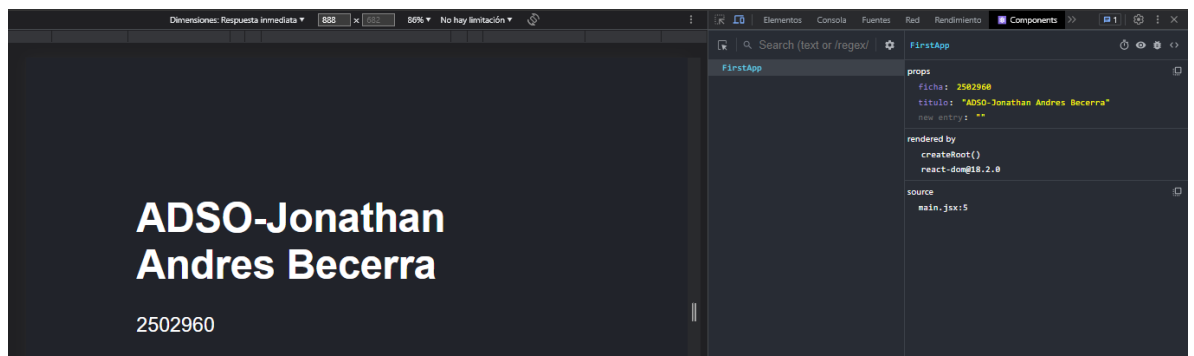
Solución:



- Sí se desea adicionar 1 al numero de la ficha, ¿En que archivo se debe enviar los valores del prop?

```
export const FirstApp = ( {tittle} ) => {  
  return (  
    <>  
      <h1>{tittle}</h1>  
      <p>{ ficha+1 }</p>  
    </>  
  )  
}
```

Solución:



```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <FirstApp titulo='ADSO-Jonathan Andres Becerra' ficha={2502959+1}/>
  </React.StrictMode>
);
```

```
export const FirstApp = ( { titulo,ficha } ) => {
  return (
    <>
      <h1>{titulo}</h1>
      <p>{ficha}</p>
    </>
  )
}
```

6.1 PROPTYPES

- Borra el Prop "title" del archivo main.jsx

```
✓ ReactDOM.createRoot(document.getElementById('root')).render(
✓ |   <React.StrictMode>
  |     <FirstApp ficha={123}/>
  |   </React.StrictMode>
  | );
```

- Al colocar un IF para colocar un mensaje de error en caso de no recibir el "title", nos genera un error en el componente ¿Por qué?

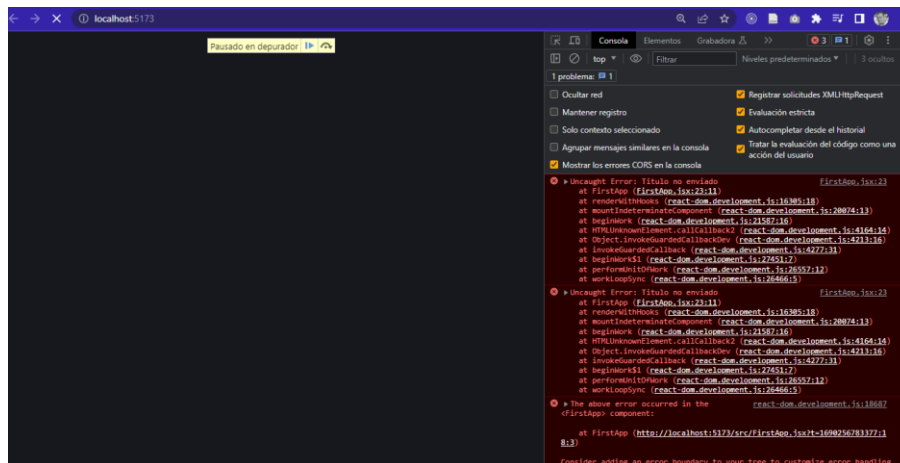
Solución: por que no se esta viendo el prop tittle en el main por eso aparece el internet

```

export const FirstApp = ( {tittle, ficha} ) => {
  if(!tittle){
    throw new Error ('Titulo no enviado');
  }

  return (
    <>
    <h1>{tittle}</h1>
    <p>{ficha+1}</p>
    </>
  )
}

```



Configuración de Proptypes

- ✓ Realizar la importación de Prop-types en el archivo FirstApp.jsx

```

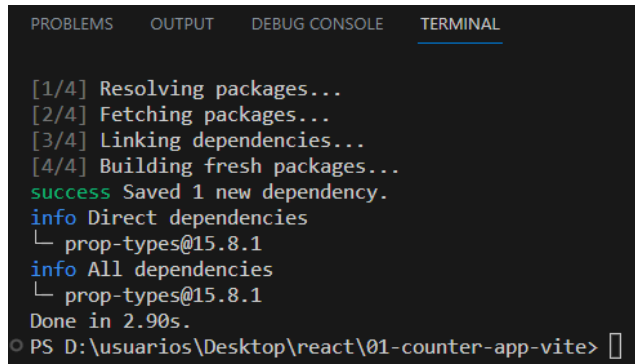
FirstApp.jsx x main.jsx
src > FirstApp.jsx > ...
19
20 import PropTypes from 'prop-types';
21
22 export const FirstApp = ( {tittle, ficha} ) => {
23

```

- ✓ Ejecutar por consola CMD, el siguiente comando (si utilizó CRA no es necesario), no usar la RED del SENA, pues da error:

```
PS D:\usuarios\Desktop\react\01-counter-app-vite> yarn add prop-types
```

No debe presentar errores.

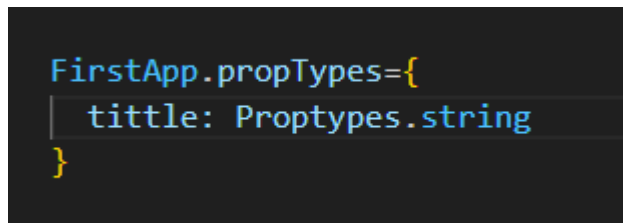


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved 1 new dependency.
info Direct dependencies
└─ prop-types@15.8.1
info All dependencies
└─ prop-types@15.8.1
Done in 2.90s.
PS D:\usuarios\Desktop\react\01-counter-app-vite>
```

(Si utiliza npm el comando sería: “npm install prop-types”)

6.2 Definir las PropTypes

Podemos definir un PropTypes, para hacer “obligatorio” el envío de un determinado tipo de Prop.



```
FirstApp.propTypes={
  title: PropTypes.string
}
```

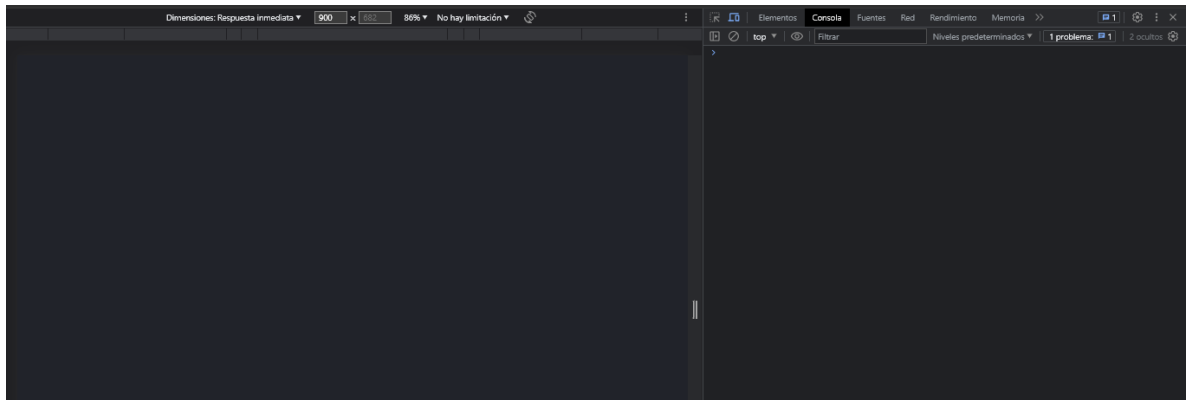
Escribe 4 tipos de PropTypes que encontramos en React.

Solución:

- String
- Objeto
- Numérico
- Array

6.3 ¿Qué sucede si no se envía el Prop que hemos definido?, coloca captura de pantalla y da una explicación de lo ocurrido.

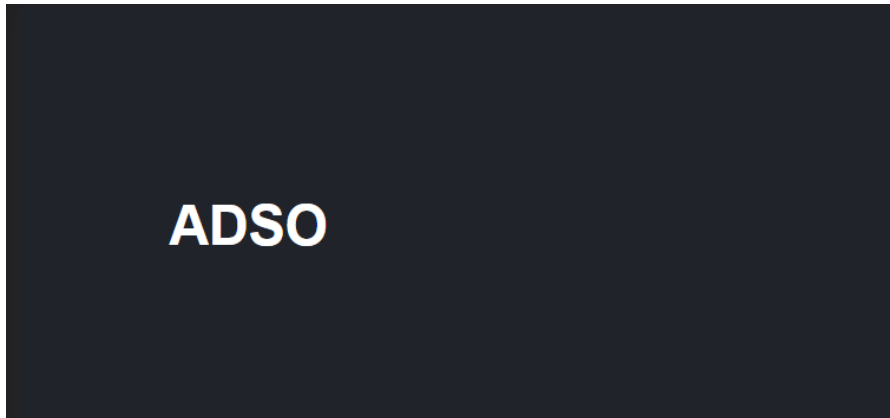
Solución:



No lo muestra ya que no está definido el prop title

6.4 Envía en el Prop “title” el valor de string “ADSO”, ¿Qué sucede?

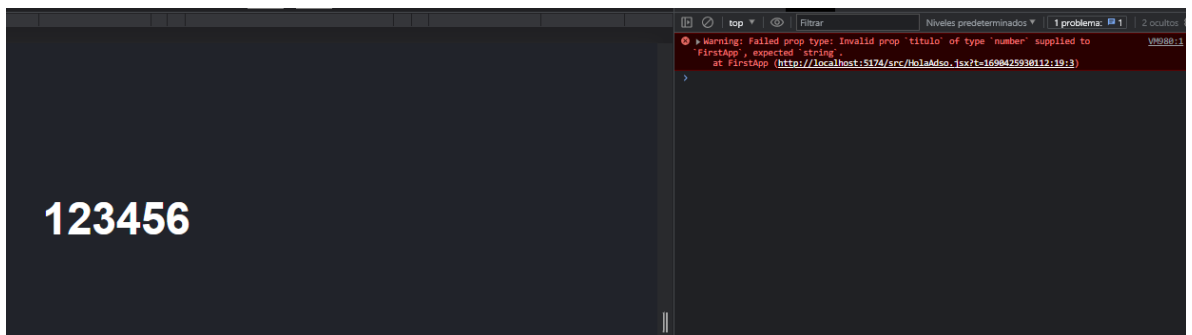
Solución:



Muestra el mensaje ya que esta definido y tiene algo por dentro el prop title

6.5 Ahora reemplaza el valor del Prop “title” por un valor de tipo numérico {123}, ¿Qué sucede? (Da tu explicación y coloca captura de pantalla)

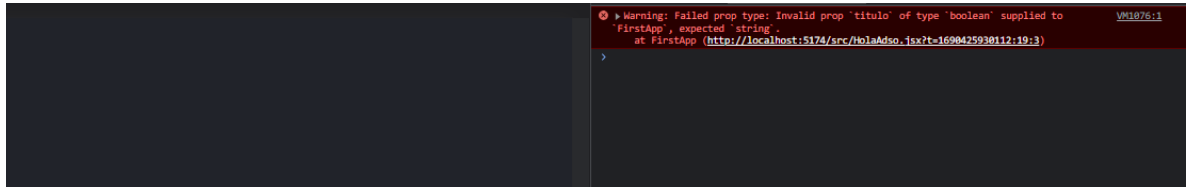
Solución:



Muestra el numero en pantalla pero en la consola muestra un error ya que el prop espera es un string o sea una cadena de texto y no un dato numérico

6.6 Reemplaza el valor del Prop “title” por un valor de tipo booleano, ¿Qué sucede? (Da tu explicación y coloca captura de pantalla).

Solución:



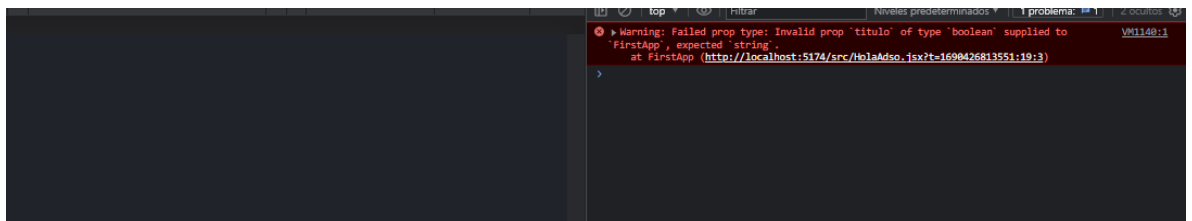
muestra un error ya que le mandamos un boolean y el prop espera es un string ósea una cadena de texto

6.7 Implementa el siguiente código:

```
FirstApp.propTypes={
  tittle: Proptypes.string.isRequired
}
```

¿Cuál sería su funcionalidad?, ¿Qué sucede sí el prop va vacío?, adjunta captura de pantalla con el error

Solución:



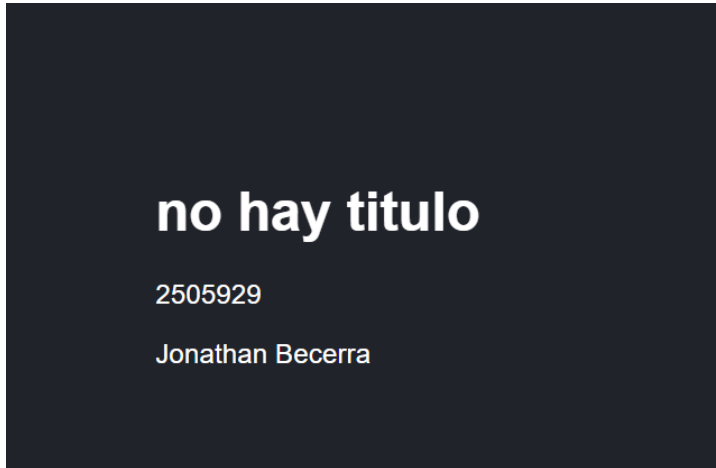
Da un error de que no se encuentra definido el value tittle pero es requerido por el prop firstapp

7. DefaultProps

7.1 Implementa el siguiente código, reemplazando el valor de “name” por tu respectivo nombre

```
FirstApp.defaultProps = {
  tittle: "no hay título",
  ficha: 1234,
  nombre: "Emerson"
}
```

Explica el funcionamiento de “defaultProps” y coloca captura de pantalla con los resultados.



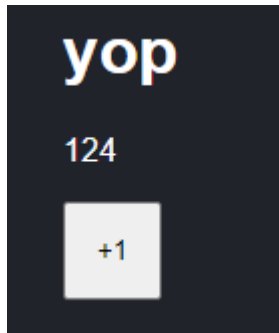
El defaultProps envía un prop por defecto si no hay ningún prop llamado en el main

8. EJERCICIO

1. Crea un nuevo componente en la carpeta SRC, nombrado ContadorApp
2. El ContadorApp debe de ser un **Functional Component**
3. El contenido del **ContadorApp** debe de ser:
 4. `<h2>ContadorApp</h2>`
 5. `<h3> { value } </h3>`
 6. Donde "value" es una propiedad enviada desde el padre hacia el componente **ContadorApp (Debe ser numérica validada con PropTypes)**
 7. Modificar en el index.js ó main.jsx el componente de `<FirstApp />` por el componente `<CountadorApp />` (no se olviden del value que debe de ser de tipo numérico)
8. No debe tener errores ni warnings

9. EVENTO CLIC (Eventos en general)

9.1 Crea un botón y agrégale estilos



9.2 Implementa el siguiente código para dar una funcionalidad al botón creado

```
import PropTypes from 'prop-types';

export const CounterApp = ({ value }) => {

  function handleAdd(event, newValue) {
    // console.log(event)
    console.log(newValue);
  }

  return (
    <>
      <h1>CounterApp</h1>
      <h2> { value } </h2>

      <button onClick={ (event) => handleAdd(event, 'hola') }>
        +1
      </button>
    </>
  );
};

CounterApp.propTypes = {
```

9.3 Si modificamos el prop “value” ¿por qué no se modifica en el DOM?

Solución: porque ya predefinimos un valor inicial que esta en el main y así cuando renderice va a mostrar el valor predefinido en el main


```
export const FirstApp = ( {tittle, ficha, nombre, value} )

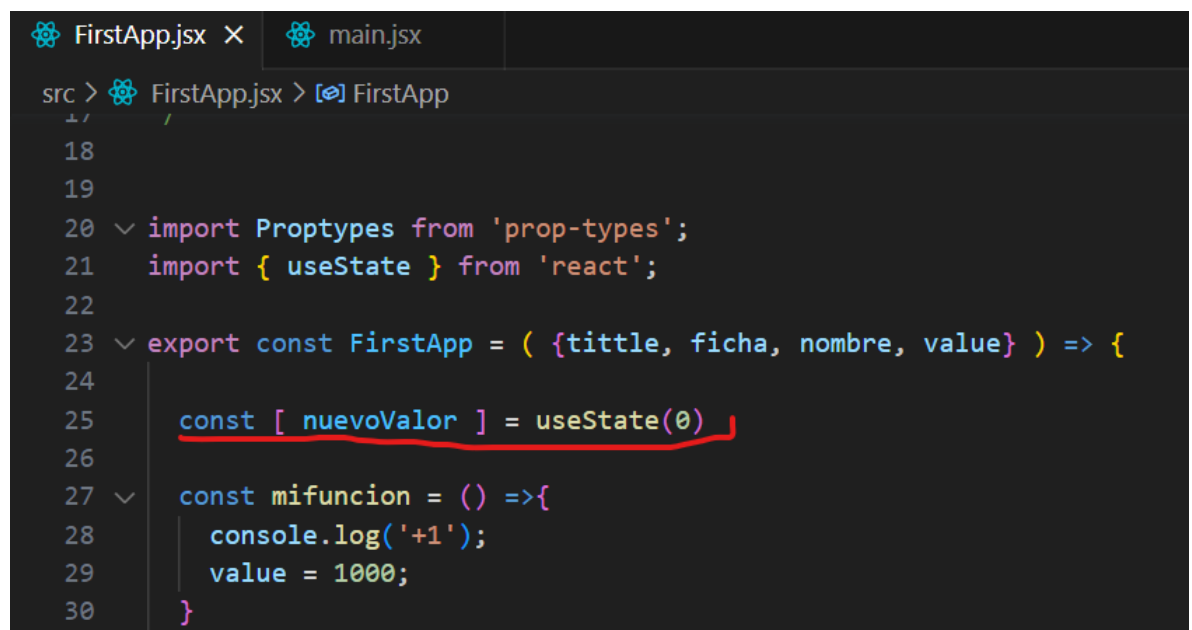
const mifuncion = () =>{
  console.log('+1');
  value = 1000;
}
```

10. useState - Hook

10.1 De acuerdo a las exposiciones de los compañeros y bibliografía consultada, ¿Qué es un Hook?

Solución: un Hooks son funciones que te permiten “enganchar” el estado de React y el ciclo de vida desde componentes de función. Los hooks no funcionan dentro de las clases — te permiten usar React

10.2 Implementa el hook **useState**, que se muestra en la línea 25 del código y complementa con la línea 42, coloca captura de pantalla con el resultado que se visualiza en la interfaz.



```
FirstApp.jsx ×  main.jsx
src > FirstApp.jsx > FirstApp
18
19
20 import PropTypes from 'prop-types';
21 import { useState } from 'react';
22
23 export const FirstApp = ( {tittle, ficha, nombre, value} ) => {
24
25   const [ nuevoValor ] = useState(0)
26
27   const mifuncion = () =>{
28     console.log('+1');
29     value = 1000;
30   }
```

```

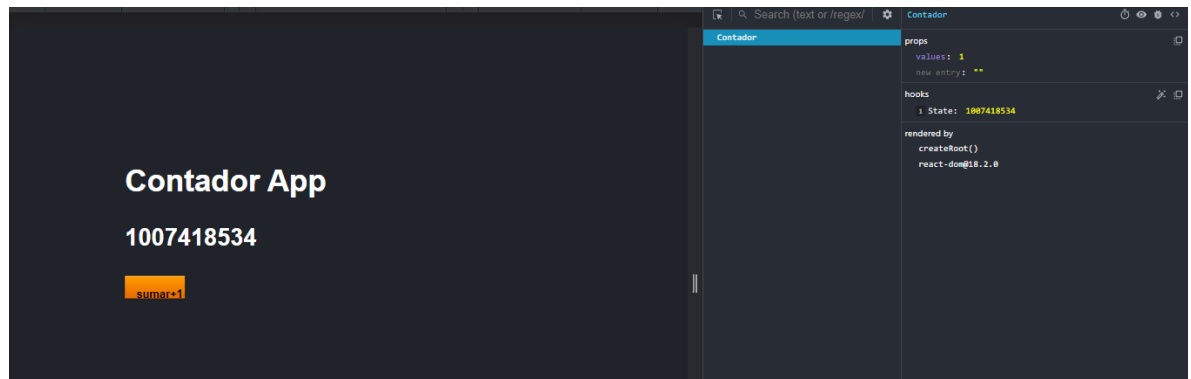
38      <>
39      <h1>{tittle}</h1>
40      <p>{ficha+1}</p>
41      <p>{nombre}</p>
42      <h1>{nuevoValor}</h1>
43      <h1>{value}</h1>
44

```

10.3 En el inspector de código del navegador, ingresar a componentes – hook y modificar el Valor del **State** por su respectivo número de documento, anexar captura de pantalla



Solución:

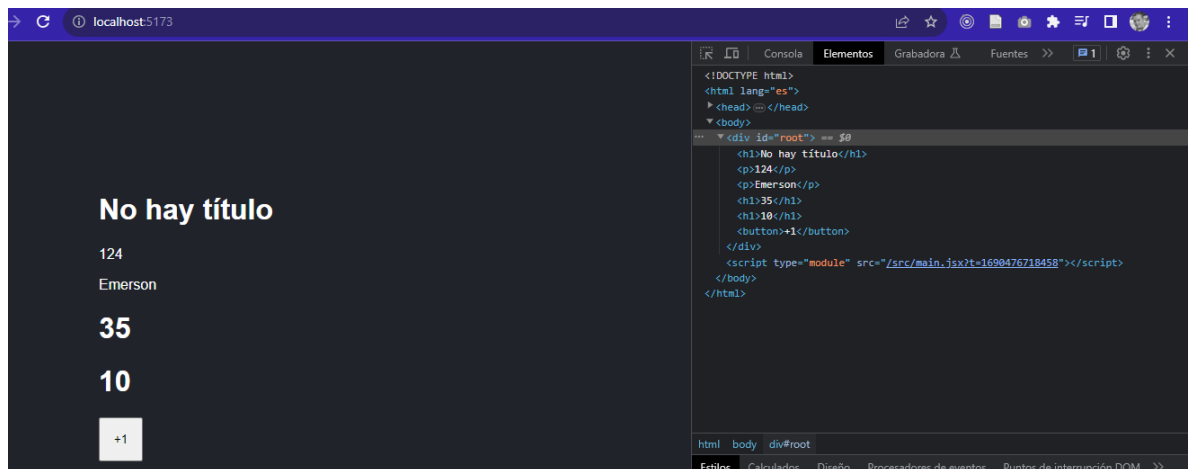


10.4 Para hacer funcionar el botón “+1”, implementar el siguiente código

```
//const [ nuevoValor ] = useState(0)
const [ nuevoValor, setNuevoValor ] = useState(0)
```

```
28
29   const mifuncion = () =>{
30     console.log('+1');
31     value = 1000;
32     setNuevoValor( nuevoValor+1 ); ←
33   }
34
```

10.5 ¿Qué observas de especial en “inspector de código - elementos”, cada vez que se hace clic en el botón “+1”?



10.6 Implemente esta opción de código para utilizar el useState:

```
29
30  const mifuncion = () =>{
31    console.log('+1');
32    value = 1000;
33    // setNuevoValor( nuevoValor+1 );
34    setNuevoValor((n)=> n+1 ); ←
35  }
36
```

10.7 Realizar el ajuste necesario para que el valor inicial del useState, tome el valor del prop “value” enviado desde el archivo main.jsx

```
src > main.jsx
6
7
8  //import { HolaAdso } from './HolaAdso';
9
10 import './styles.css';
11
12
13
14 ReactDOM.createRoot(document.getElementById('root')).render(
15   <React.StrictMode>
16     <FirstApp ficha={123} value={70}/>
17   </React.StrictMode>
18 );
```

Solución:

Aplicación Contador

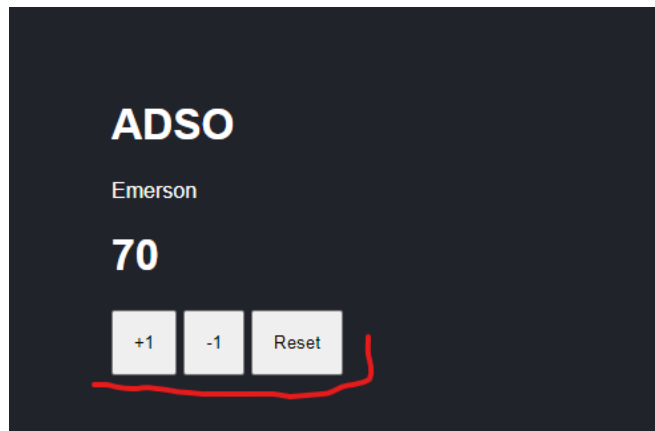
23

sumar+1

```
export default function Contador({values}) {
  const [value, setvalue] = useState(values);
  const onClick = () => {
    setvalue((value) => value + 1)
  }
  return (
    <>
      <h2>Contador App</h2>
      <h3>{value}</h3>
      <button onClick={onClick}>sumar+1</button>
    </>
  )
}
```

10.8 EJERCICIO

- Implementa los botones “+1”, “-1” y “Reset”.



- Al presionar el botón “+1” el valor inicial debe aumentar en una unidad
- Al presionar el botón “+2” el valor debe disminuir en 1
- Al presionar el botón “Reset” el valor que debe mostrar es el inicial.

Solución:

no hay titulo

2505929

Jonathan Becerra

Contador App

0

+1	-1	0
----	----	---

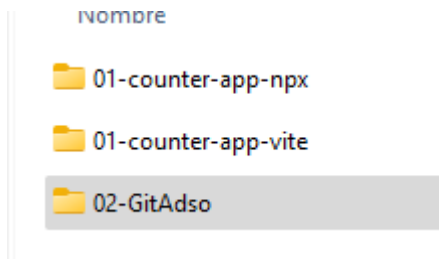
```
import PropTypes from 'prop-types';
import { useState } from 'react';

export default function Contador({values}) {
  const [value, setvalue] = useState(values);
  const onClick = () => {
    setvalue((value) => value + 1)
  }
  const restar = () => {
    setvalue((value) => value -1)
  }
  const volver = () => {
    setvalue((value) => values )
  }
  return (
    <>
      <h2>Contador App</h2>
      <h3>{value}</h3>
      <button onClick={onClick}>+1</button>
      <button onClick={restar}>-1</button>
      <button onClick={volver}>0</button>
    </>
  )
}

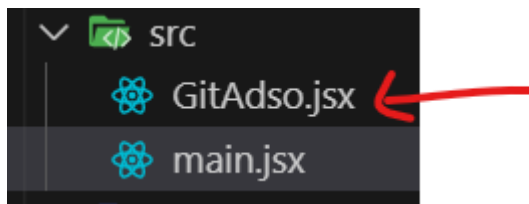
PropTypes
Contador.propTypes = {
  value: PropTypes.number
}
```

11. CREANDO GitAdso

11.1 Crear el directorio "GitAdso"



11.2 Teniendo en cuenta los primeros pasos de esta guía, eliminar todos los archivos que se encuentran dentro de la carpeta "SRC" conservando únicamente el archivo "main.jsx" y creando uno nuevo nombrado "GitAdso.jsx"



11.3 Realizar las instalaciones y parametrizaciones necesarias para obtener el siguiente resultado:



GitAdso

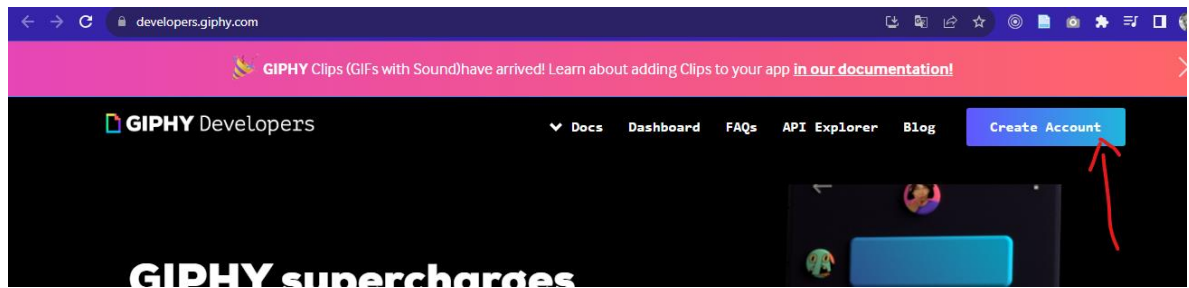
```
GitAdso.jsx  main.jsx  X
02-GitAdso > src > main.jsx
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import GitAdso from './GitAdso';
4  //import App from './App.jsx'
5  //import './index.css'
6
7  ReactDOM.createRoot(document.getElementById('root')).render(
8    <React.StrictMode>
9    |   <GitAdso />
10   </React.StrictMode>,
11  )
12
```

```
GitAdso.jsx  X  main.jsx
02-GitAdso > src > GitAdso.jsx > GitAdso
1  //comando rfc para crear el functional component
2  export default function GitAdso() {
3    return (
4      <>
5      |   <h1>GitAdso</h1>
6      </>
7    )
8  }
```

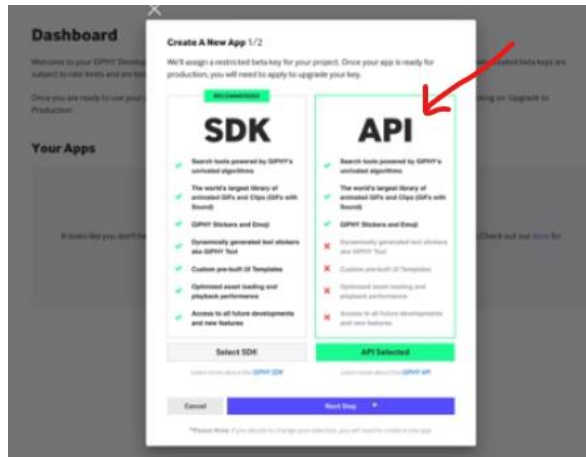
11.4 Crear cuenta en GIPHY

<https://developers.giphy.com/>

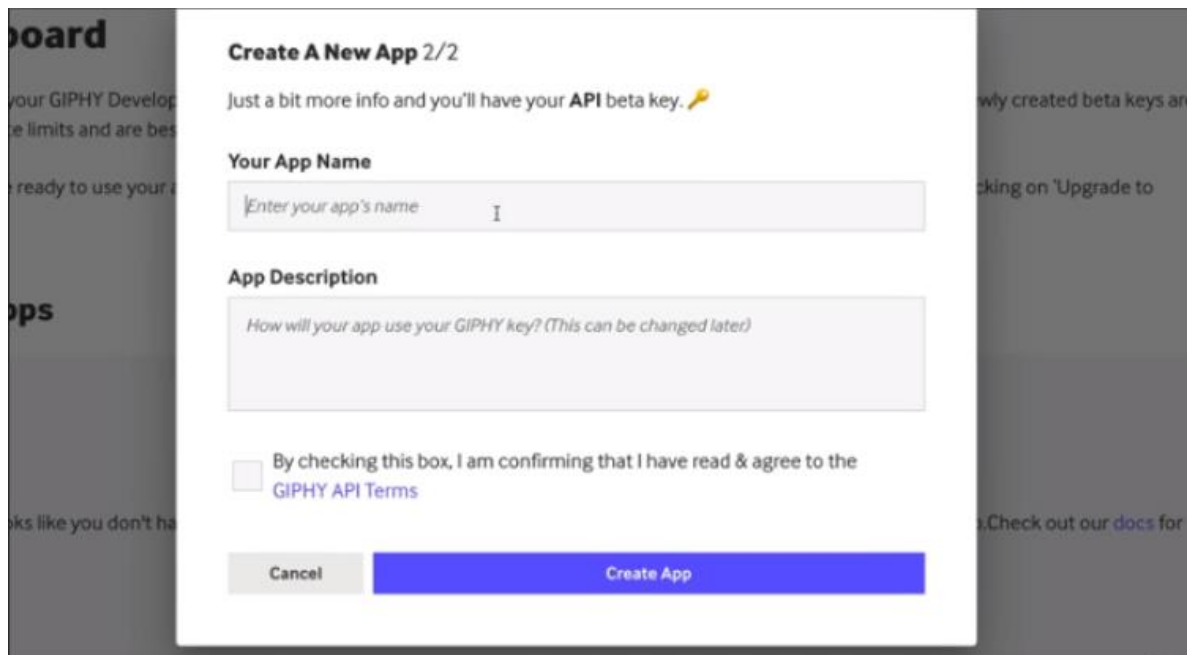
- ✓ Clic en create account



✓ Seleccionar tipo API



✓ Colocar el nombre a la app, ejemplo GitAdso




✓ Conservar el API Key

Edit Your App

App Name

GitADSO

API Key

Ad83qHVIA7Sw1wuomC5Tk2R

Description

App prueba ADSO

Solución:

Production.

Your Apps

GITADSO API Edit

API Key

P9z94JoLr4ZipEJERzER2IH8Ulj4q2

Upgrade to Production

+

Ready to build another app already?
Click anywhere to get started!

Create an App

<https://developers.airbnb.com/apps/413463/edit/>

12. Crear lista de categorías

- 12.1 En el archivo "GitAdso.jsx", crear el useState a utilizar. Puede usar el snippet "useStateSnippet" para crearlo.

```
3 //usar useStateSnippet  
4 const [first, setfirst] = useState(second)
```

- 12.2 Importar el useState

```

02-GitAdso > src > GitAdso.jsx > ...
1  import { useState } from 'react';
2
3  //comando rfc para crear el functional component
4  export default function GitAdso() {

```

12.3 Consultar la función “.map” en javascript

12.4 En el useState crear un array de personajes

```

5  //usar useStateSnippet
6  const [categories, setCategories] = useState(['Vegeta', 'Picoro']);
7

```

12.5 Colocar los personajes en una lista ordenada de html “”, recorriendo el arreglo con “.map” de javascript

```

<ol>
  {
    categories.map( category => {
      return <li key={ category }>{ category }</li>
    })
  }
</ol>

```

12.6 Agrega a la lista un tercer elemento el cual debe ser su primer nombre, adjunta captura de pantalla.



Solución:

GITADSO

1. vegeta
2. picoro
3. Jonathan

- 12.7 Agregar un botón a la interfaz que permita agregar una nueva categoría(s) a la lista, modificando el estado del hook - "useState". (E

Solución:

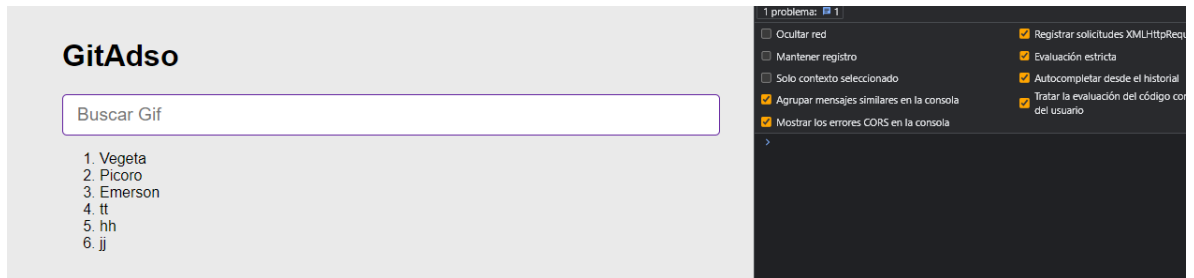
GITADSO

1. vegeta
2. picoro
3. Jonathan
4. pedro
5. juan

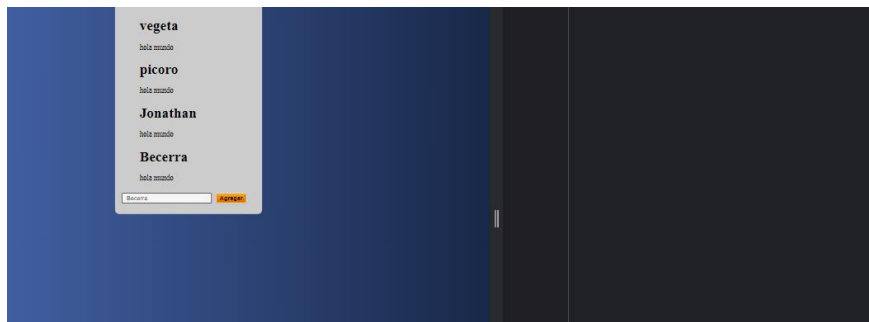
```
import React from 'react'
import { useState } from 'react'

export default function GitAdso() {
  const [categorias, setCategorias] = useState(['vegeta', 'picoro', 'Jonathan'])
  const agregar = () => {
    let añadir = document.getElementById('adicionar').value;
    setCategorias([...categorias, añadir])
    // if (añadir !== '') {
    //   añadir = false;
    //   alert('Por favor inserte un personaje');
    // }
  }
  return (
    <>
      <h2>GITADSO</h2>
      <ol>
        {
          categorias.map(categoria => {
            return <li key={categoria}>{categoria}</li>
          })
        }
      </ol>
      <input type="text" id="adicionar" />
      <button onClick={agregar}>Agregar</button>
    </>
  )
}
```

- 12.8 Generar la siguiente funcionalidad, la cual consiste en crear un input type “text”, y al presionar la tecla “Enter” solo debe agregarla a la lista Sí y solo Sí esta palabra aun no se encuentra en la lista.

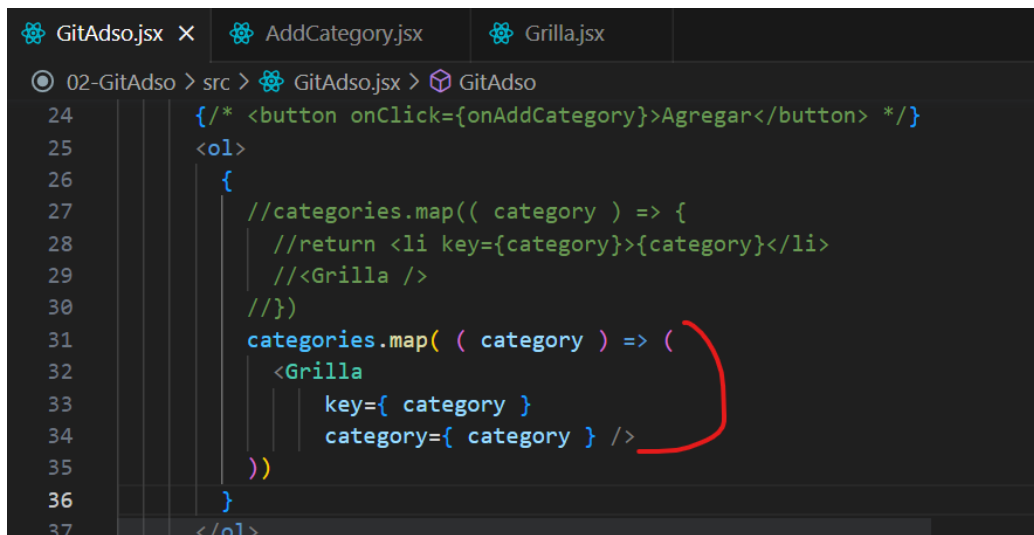


Solucion



13. VISUALIZAR IMÁGENES CONSUMIENDO API GIPHY

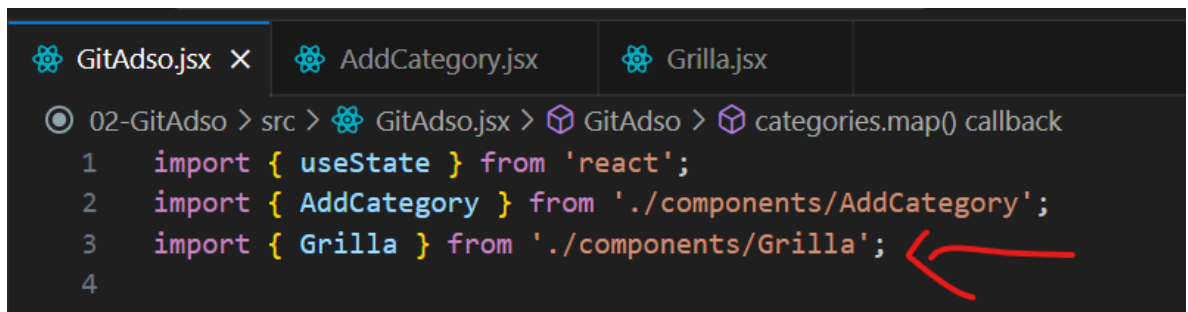
- 13.1 Dentro de la ruta “SRC - components” crear el archivo “Grilla.jsx”
- 13.2 En el archivo “GitAdso.jsx”, modificar la siguiente línea de código



```
24  /* <button onClick={onAddCategory}>Agregar</button> */
25  <ol>
26  {
27    //categories.map( ( category ) => {
28      //return <li key={category}>{category}</li>
29      //<Grilla />
30    //})
31    categories.map( ( category ) => (
32      <Grilla
33      key={ category }
34      category={ category } />
35    )
36  }
37  </ol>
```

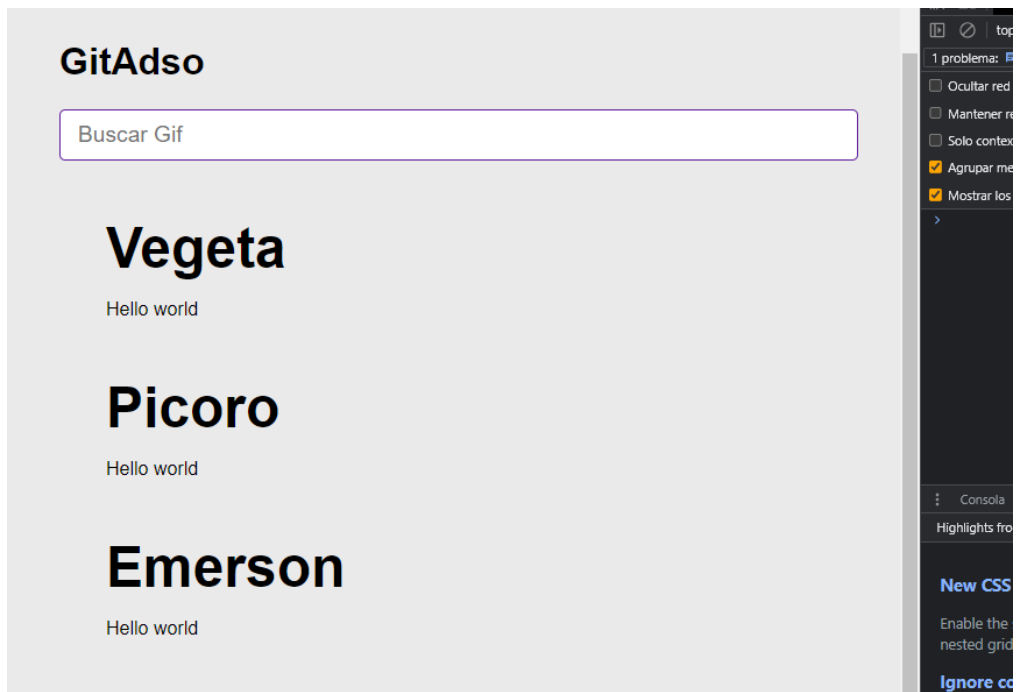
Hemos comentado la línea 27, 28, 29 y 30, ahora solamente vamos a devolver el nuevo componente creado (**Grilla**).

- 13.3 Verificamos que se haya importado automáticamente **Grilla**, de no ser así, debemos realizarlo manualmente

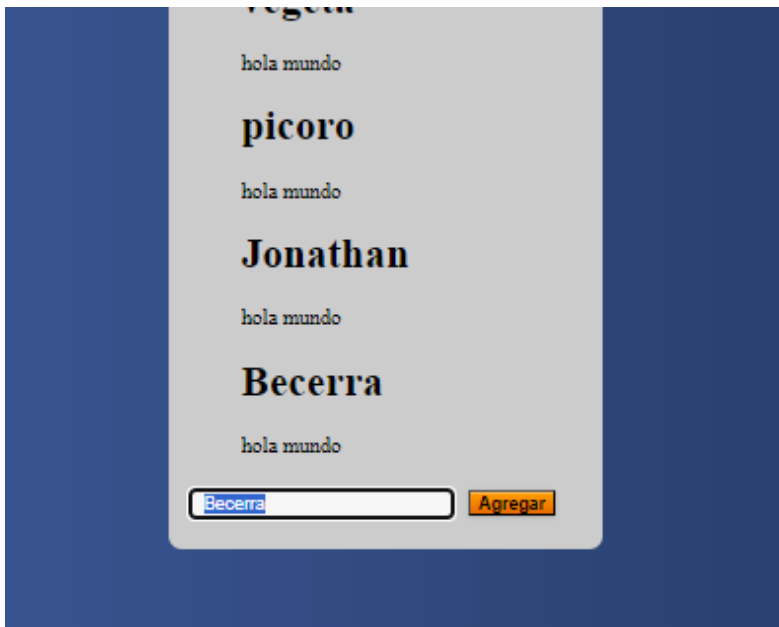


```
02-GitAdso > src > GitAdso.jsx > GitAdso > categories.map() callback
1  import { useState } from 'react';
2  import { AddCategory } from './components/AddCategory';
3  import { Grilla } from './components/Grilla';
4
```

El resultado debe ser:



Solución:



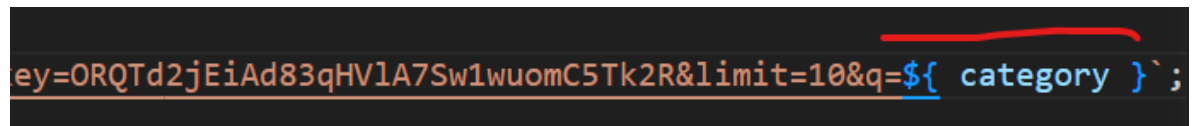
14. OBTENER IMÁGENES

14.1 En el componente “Grilla.jsx” crear las líneas:



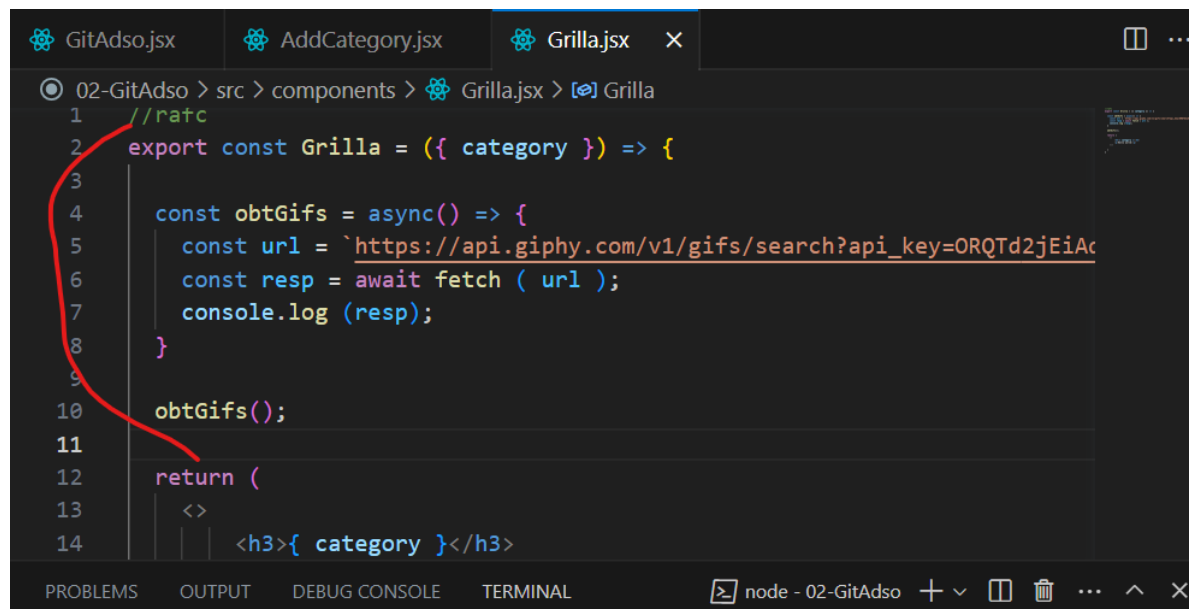
```
1 //ratc
2 export const Grilla = ({ category }) => {
3
4   const obtGifs = () => {
5     const url = `https://api.giphy.com/v1/gifs/search?api_key=ORQTd2jEiAc
6   }
7   return (
8     <>
9       <h3>{ category }</h3>
10      <p>Hello world</p>
11    </>
12  )
13 }
```

En “url” copiaremos la dirección proporcionada por GIPHY (ten presente el final de la línea de código de la url:

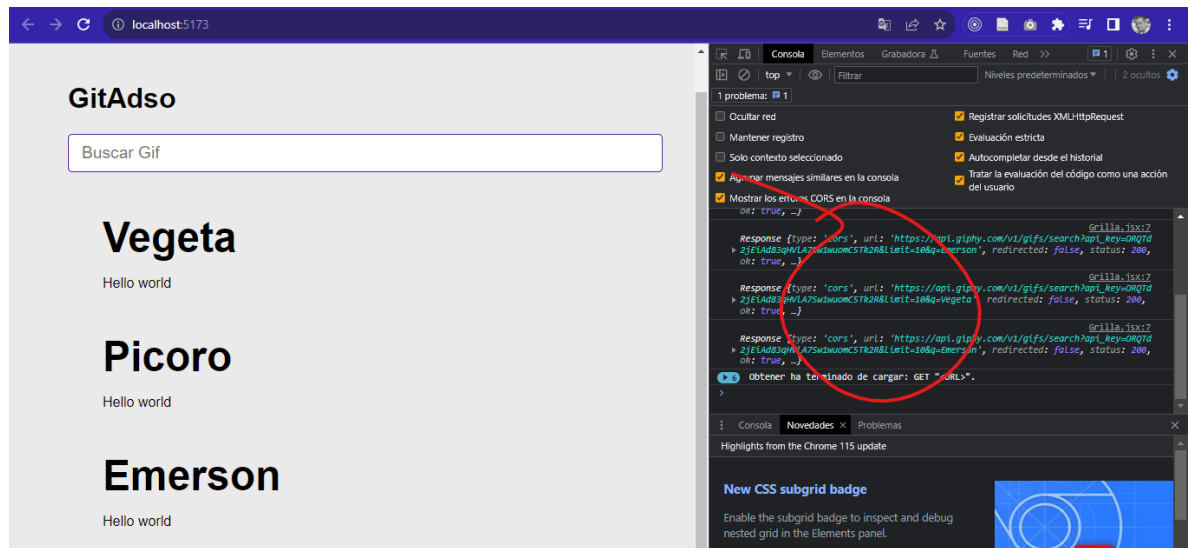


```
ey=ORQTd2jEiAd83qHV1A7Sw1wuomC5Tk2R&limit=10&q=${ category }`;
```

14.2 Implementar (adjuntar captura de pantalla):



```
1 //ratc
2 export const Grilla = ({ category }) => {
3
4   const obtGifs = async() => {
5     const url = `https://api.giphy.com/v1/gifs/search?api_key=ORQTd2jEiAc
6     const resp = await fetch ( url );
7     console.log (resp);
8   }
9
10  obtGifs();
11
12  return (
13    <>
14      <h3>{ category }</h3>
15    </>
16  )
17 }
```

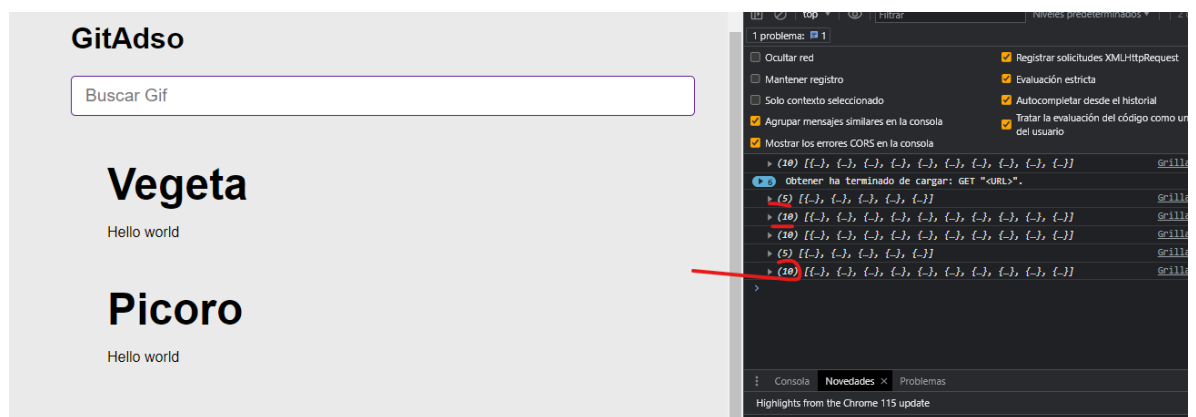



14.3 Ahora podemos mostrar la data que se esta trayendo:

```

2  export const Grilla = ({ category }) => {
3
4      const obtGifs = async() => {
5          const url = `https://api.giphy.com/v1/gifs/search?api_key=0RQtd
6          const resp = await fetch ( url );
7          //console.log (resp);
8          const { data } = await resp.json();
9          console.log ( data )
10     }
11
12     obtGifs().

```

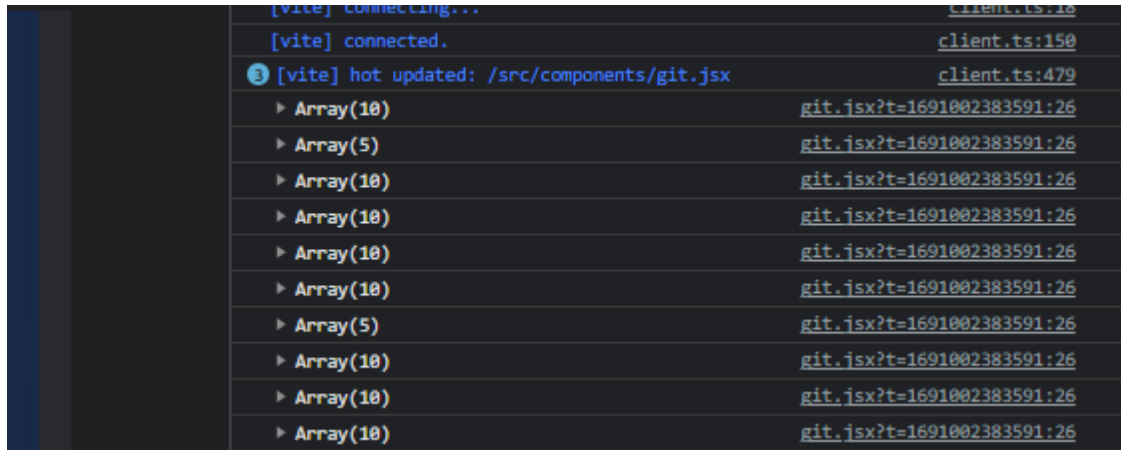


¿Qué debemos implementar si deseamos ampliar el rango de la búsqueda a 15 resultados?

Solución: debemos implementar un limit de 15 donde esta la url con el key que nos da postman de la siguiente manera

```
const gifs = async () => {  
  const url = `https://api.giphy.com/v1/gifs/search?api_key=P9z94Jolr4ZipEJERzER2IH8U1jpJ4q2&limit=15&q=${categoria}`;  
}
```

Antes de poner categoría pones el limite hay y se mostrara por con sola de la siguiente manera

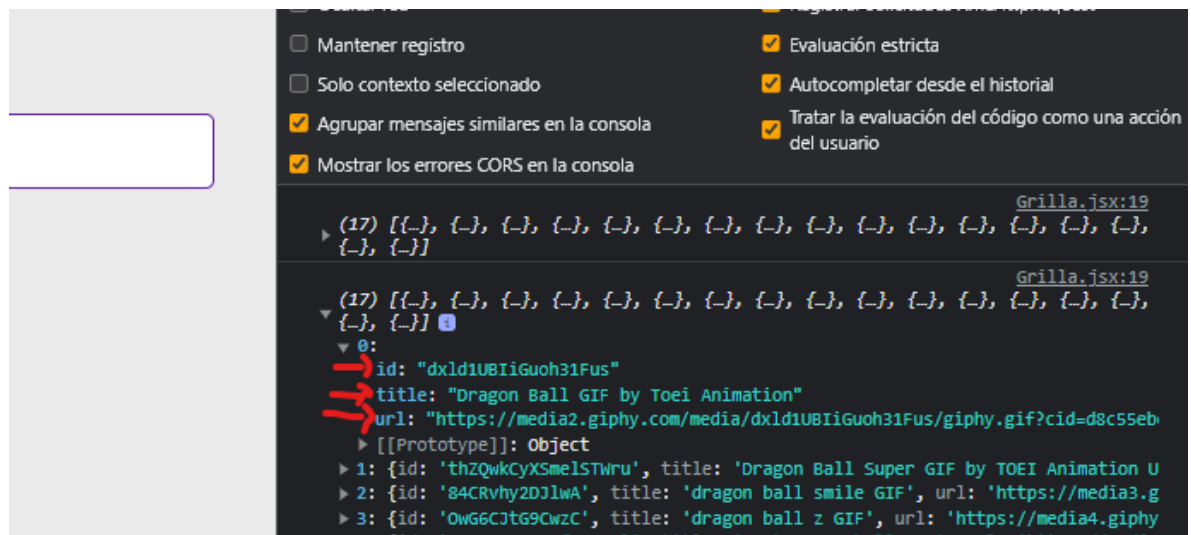


```
[vite] connecting... client.ts:18  
[vite] connected. client.ts:150  
3 [vite] hot updated: /src/components/git.jsx client.ts:479  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(5) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(5) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26  
  ▶ Array(10) git.jsx?t=1691002383591:26
```

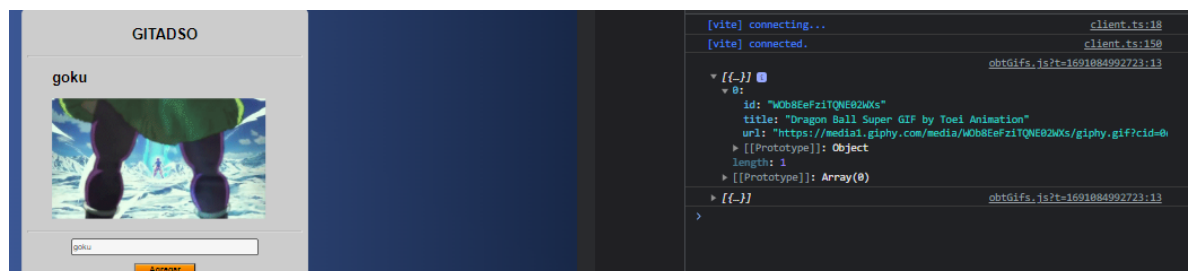
14.4 Implementar el siguiente código en “Grilla.jsx”

```
GitAdso.jsx  AddCategory.jsx  Grilla.jsx  X
02-GitAdso > src > components > Grilla.jsx > Grilla > obtGifs
8      const { data } = await resp.json();
9      //console.log ( data )
10
11      const imagenes = data.map( img => {
12          return{
13              id: img.id,
14              title: img.title,
15              url: img.images.downsized_medium.url
16          }
17      })
18      console.log( imagenes );
19
20
21
```

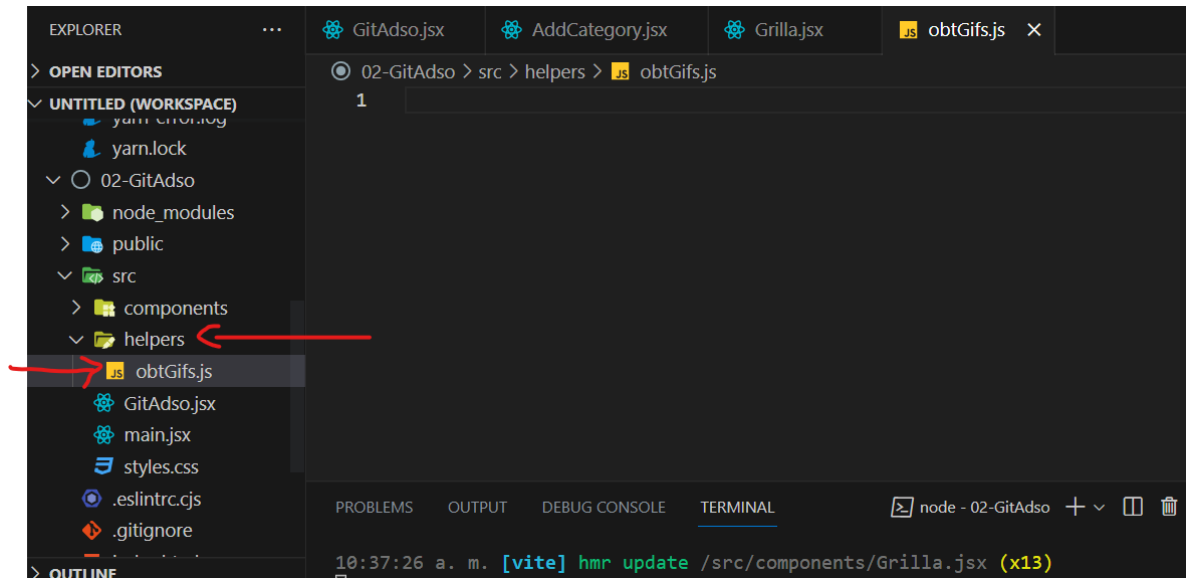
Comprobar por consola que ahora solo traiga id, title y url:



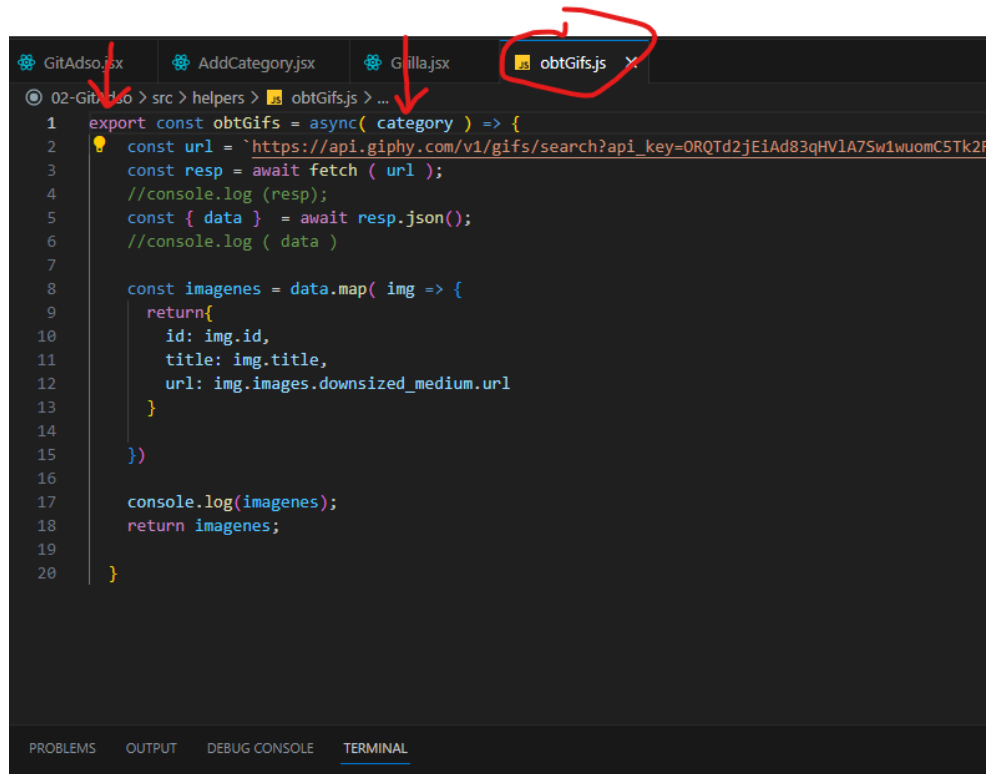
Solución:



- 14.5 En la raíz del folder SRC, crear un nuevo directorio llamado “helpers”, y dentro un archivo nombrado “obtGifs.js”



- 14.6 Vamos a separar la función “**obtGifs**”, y colocarla en el nuevo archivo obtGifs.js, teniendo en cuenta agregar la palabra “**export**” para poder utilizarla en otra parte del aplicativo y ahora recibir el parámetro **category**



```
02-GitAdso.jsx  AddCategory.jsx  Grilla.jsx  obtGifs.js X
02-GitAdso > src > helpers > obtGifs.js > ...
1  export const obtGifs = async( category ) => {
2    const url = `https://api.giphy.com/v1/gifs/search?api_key=ORQTd2jEiAd83qHV1A7Sw1wuomC5Tk2F
3    const resp = await fetch ( url );
4    //console.log (resp);
5    const { data } = await resp.json();
6    //console.log ( data )
7
8    const imagenes = data.map( img => {
9      return{
10        id: img.id,
11        title: img.title,
12        url: img.images.downsized_medium.url
13      }
14    })
15
16    console.log(imagenes);
17    return imagenes;
18  }
19
20
```

El valor del parámetro **category**, ahora debe ser enviado desde el llamado a la función “**obtGifs**” en el archivo “**Grilla.jsx**” y asegurarnos de que contemos con el import, todo debe funcionar sin errores.

```
02-GitAdso > src > components > Grilla.jsx > Grilla
1 //rafc
2
3 import { obtGifs } from "../helpers/obtGifs";
4
5 export const Grilla = ({ category }) => {
6
7
8
9   obtGifs( category );
10
11   return (
12     <>
13       <h3>{ category }</h3>
14       <p>Hello world</p>
15     </>
16   )
17 }
18
19
```

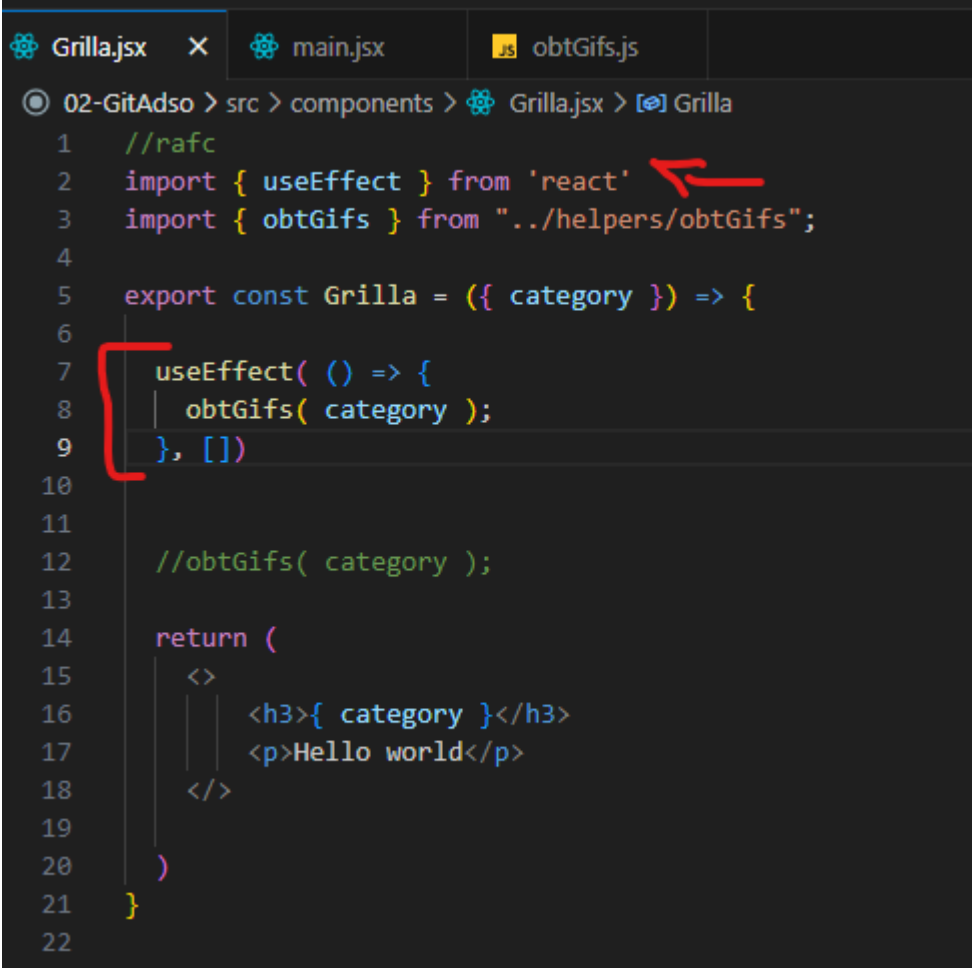
15. useEffect

useEffect en React es un tipo de hook que se incorporó en la versión de React 16.8. Como su nombre indica, este hook nos permite definir efectos. Los efectos en esta

librería de JavaScript nos permiten ejecutar un trozo de código según el momento en el que se encuentre el ciclo de vida de nuestro componente.

<https://react.dev/reference/react/useEffect>

15.1 Implementar el hook useEffect y comprobar por consola



```
Grilla.jsx  X  main.jsx  JS obtGifs.js
02-GitAdso > src > components > Grilla.jsx > Grilla
1  //rafc
2  import { useEffect } from 'react'
3  import { obtGifs } from "../helpers/obtGifs";
4
5  export const Grilla = ({ category }) => {
6
7    useEffect( () => {
8      obtGifs( category );
9    }, [])
10
11
12    //obtGifs( category );
13
14    return (
15      <>
16        <h3>{ category }</h3>
17        <p>Hello world</p>
18      </>
19    )
20  }
21
22
```

The screenshot shows a code editor with three tabs: Grilla.jsx, main.jsx, and obtGifs.js. The active file is Grilla.jsx, showing the implementation of the Grilla component. The component is a function that takes a category prop and returns a JSX element. It uses the useEffect hook to call the obtGifs function when the component mounts. The code is annotated with red markings: a red arrow points to the import of useEffect, and a red bracket highlights the useEffect call.

15.2 Finalmente para mostrar la imagen, implemente y complemente con su propio código

```
Grilla.jsx  obtGifs.js
02-GitAdso > src > components > Grilla.jsx > Grilla > images.map() callback
2 import { useEffect, useState } from 'react'
3 import { obtGifs } from "../helpers/obtGifs";
4
5 export const Grilla = ({ category }) => {
6
7   const [images, setImages] = useState([]);
8
9   const getImg = async()=>{
10     const newImg = await obtGifs(category);
11     setImages(newImg);
12   }
13
14   useEffect( () => {
15     //obtGifs( category );
16     getImg()
17   }, [])
18
19   //obtGifs( category );
20
21   return (
22     <>
23       <h3>{ category }</h3>
24       <div>
25         {
26           // ...
27         }
28     </>
29   )
30 }
```

Solución:

```
    <div>
      {
        images.map(Image => (
          <img key={Image.url} src={Image.url} width='350' />
        ))
      }
    </div>
  </>
)
```

