

REACT

Es una biblioteca popular para construir interfaces de usuario. Su arquitectura se compone de varios elementos clave que trabajan juntos para crear apps reactivas, modulares y eficientes:

¿Qué necesitamos para trabajar con esta biblioteca?

Requisitos previos

- Conocimientos de Html, CSS y JavaScript.
- Node.js y npm (gestor de paquetes de node). Para instalar <https://nodejs.org/>
- Verificar instalación: node -v // npm -v

Comenzamos:

- Se puede iniciar un proyecto con Vite o a través de Create React App.
- Si creamos proyecto a través de vite:
 - Digitamos: npm create vite
 - Ingresamos el nombre del proyecto
 - Seleccionar framework o librería
 - Seleccionar lenguaje JavaScript
- Verificar si la carpeta node_modules esta implementada, de no ser así, ejecutar el comando npm install
- Si crea un proyecto utilizando create react:
 - npx create-react-app nombre_proyecto
 - cd nombre_proyecto (ingresar a la carpeta)
 - npm start

```
C:\Users\vivis\OneDrive\Desktop\FullStack2>npm create vite
Need to install the following packages:
create-vite@6.5.0
Ok to proceed? (y) y

> npx
> cva

Project name:
myProject2
Package name:
myproject2
Select a framework:
React
Select a variant:
JavaScript
```

```
C:\Users\vivis\OneDrive\Desktop>npx create-react-app myproject
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y

Creating a new React app in C:\Users\vivis\OneDrive\Desktop\myproject.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1323 packages in 2m
269 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.
Installing template dependencies using npm...
-
```

```
Run `npm audit` for details.
Created git commit.
Success! Created myproject at C:\Users\vivis\OneDrive\Desktop\myproject
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd myproject
  npm start

Happy hacking!
C:\Users\vivis\OneDrive\Desktop>
```

Ingresa a la carpeta de proyecto y en ella encontrarás una carpeta llamada src, donde se almacena un archivo llamado App.js.

```
EXPLORER
  MYPROJECT
    > node_modules
    > public
    > src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
      .gitignore
      package-lock.json
      package.json
      README.md

src > JS App.js M X
1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10             Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13            className="App-link"
14            href="https://reactjs.org"
15            target="_blank"
16            rel="noopener noreferrer"
17          >
18            Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24  export default App;
25
```

Iniciar servidor: npm start

```
Microsoft Windows [Versión 10.0.19045.6093]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\vivis\OneDrive\Desktop\myproject>npm start

> myproject@0.1.0 start
> react-scripts start

(node:15440) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:15440) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

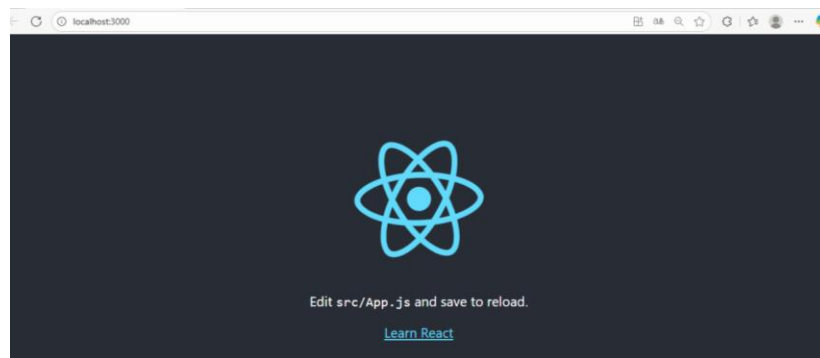
You can now view myproject in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.100.93:3000

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

webpack compiled successfully
```

React se ejecuta en <http://localhost:3000>



Nota: si el proyecto fue creado a través de Vite, para ejecutar el servidor: **npm run dev**.

React trabaja en relación a componentes. Cuando se crea un proyecto, la ejecución comienza en el componente principal, el cual es main.js. Desde este archivo por defecto, mostrará lo que renderiza la función implementada en App.jsx (componente). La visualización, de lo que contiene la función se despliega en el archivo index.html.

Componentes

Un componente es un módulo reutilizable de código que representa una parte de la interfaz de usuario (UI). Un componente puede ser algo pequeño como un botón, o tan grande como una página completa.

Tipos de componentes:

1. **Funcionales:** son los más utilizados hoy en día, corresponden a funciones en JavaScript que reciben propiedades (props) como argumento y devuelven elementos JSX (Html dentro de Javascript).

Ej: `function Saludo(props) {`
 `return <h1>Hola, {props.nombre}</h1>;`

```
}
```

2. **Componentes de clases:** son menos utilizados actualmente, y son clases que extienden `React.Component` y deben tener un método `render()`.

Ej:

```
class Saludo extends React.Component {  
  render() {  
    return <h1>Hola, {this.props.nombre}</h1>;  
  }  
}
```

Componente Principal:

Es el punto de entrada de la interface, por convención suele llamarse `App()`.

Es el contenedor principal de una aplicación. Organiza la navegación, rutas (en caso de utilizar `React Router`). Desde este contenedor principal, se puede organizar toda la UI, incluyendo encabezados, menús, páginas, etc.

Por ej: si modificamos nuestro archivo `App.jsx`:

```
main.jsx  App.jsx  x  
rc > App.jsx > ...  
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     <>  
11       <div>  
12         <h2>Bienvenido a React</h2>  
13       </div>  
14     </>  
15   )  
16 }  
17 export default App  
18
```

Este archivo se renderiza en el archivo principal de Javascript: **main.jsx**

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

El componente principal, debe estar importado, al ejecutar la aplicación se visualizará en pantalla:



Creando nuestro primer componente

Crearemos una carpeta llamada componentes dentro de la carpeta /src de nuestro proyecto, lo llamaremos 'HelloWorld.jsx'. Para poder utilizar este componente debe contener la palabra reservada export, de tal forma que puede ser renderizado desde el archivo principal.

```
main.jsx  HelloWorld.jsx  App.jsx
src > components > HelloWorld.jsx > ...
1
2 export function HelloWorld(){
3   return(
4     <div>
5       <h1>Hola Mundo</h1>
6     </div>;
7   )
}
```

Es una function JavaScript que retorna codificación html.

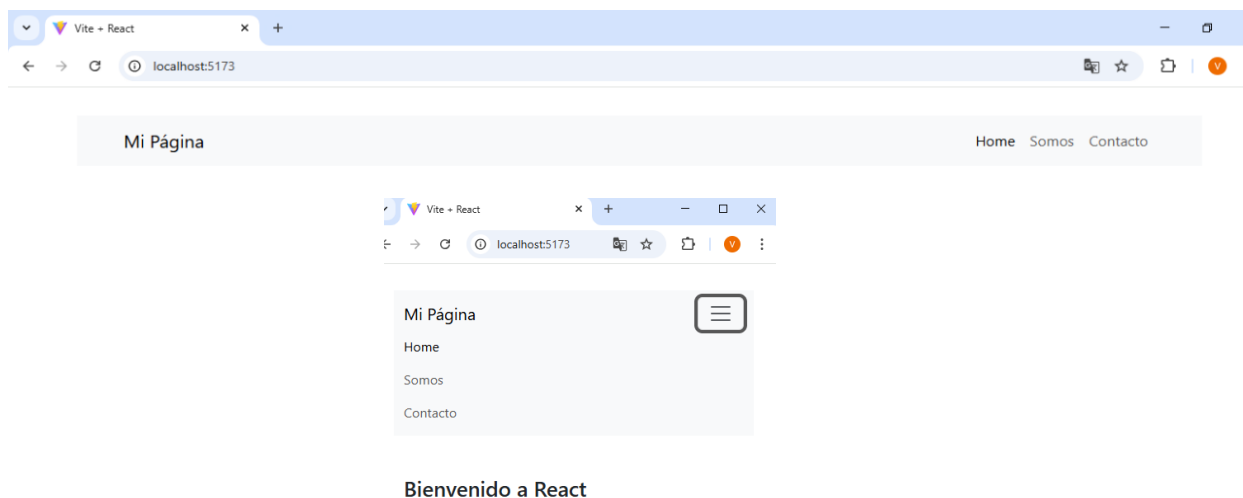
En el archivo principal 'main.jsx' importamos el componente creado y luego lo implementamos en la ruta principal del proyecto:

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

Ahora en la ejecución:



Ahora, crearemos un menú de opciones:



En la carpeta componentes, crearemos **Header.jsx**, el cual tendrá el menú de navegación.

```
function Header() {  
  return (  
    <nav className="navbar navbar-expand-lg navbar-light bg-light">  
      <div className="container">  
        <a className="navbar-brand" href="#">Mi Página</a>  
        <button className="navbar-toggler" type="button" data-bs-toggle="collapse"  
          data-bs-target="#menuNav">  
          <span className="navbar-toggler-icon"></span>  
        </button>  
  
        <div className="collapse navbar-collapse" id="menuNav">  
          <ul className="navbar-nav ms-auto">  
            <li className="nav-item">  
              <a className="nav-link active" href="#">Home</a>  
            </li>  
            <li className="nav-item">  
              <a className="nav-link" href="#">Somos</a>  
            </li>  
            <li className="nav-item">  
              <a className="nav-link" href="#">Contacto</a>  
            </li>  
          </ul>  
        </div>  
      </div>  
    </nav>  
  );  
}  
export default Header;
```

Este menú utiliza Bootstrap, por lo tanto, no debemos olvidarnos de implementar las bibliotecas que corresponden. En el archivo **index.html**, en la sección <head> realizamos lo siguiente:

```
<head>  
  <meta charset="UTF-8" />  
  <link rel="icon" type="image/svg+xml" href="/vite.svg" />  
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  
    rel="stylesheet"/>  
  <script  
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
  <title>Vite + React</title>  
</head>
```

En el archivo App.jsx implementamos los componentes creados, debiendo ser previamente importados donde corresponde:

```
Header.jsx  index.html  App.jsx  main.jsx
src > App.jsx > App
1  import { useState } from 'react'
2  import './App.css'
3  import Header from './components/Header';
4  import { HelloWorld } from './components/HelloWorld'
5
6
7  function App() {
8    const [count, setCount] = useState(0)
9    return (
10     <>
11       <Header/>
12       <HelloWorld/>
13     </>
14   )
15 }
16 export default App
17
```

En el archivo main.jsx:

```
Header.jsx  index.html  App.jsx  main.jsx  HelloWorld.js
main.jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App';

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App/>
  </StrictMode>,
)
```

Nota: si el texto del menú se visualiza 'centrado' al igual que el del componente HelloWorld, debemos modificar el atributo 'text-align' del [archivo App.css](#).

Trabajando con variables

1. Variables y propiedades en un componente:

```
HelloWorld.jsx  App.jsx
src > components > HelloWorld.jsx > ...
1
2  export function HelloWorld(props){
3    const nombre='Cachupin'
4    return(
5      <>
6        <div className="container mt-5">
7          <h2>Bienvenido a React {nombre}</h2>
8        </div>
9      </>
10    );
11  }
12
```

2. Traspasando parámetros de padre a hijo:
 - a. En el componente Padre, App.jsx, enviamos como parámetro la información que mostraremos en el componente hijo:

```
HelloWorld.jsx  App.jsx
src > App.jsx > ...
1  import { useState } from 'react'
2  import './App.css'
3  import Header from './components/Header';
4  import { HelloWorld } from './components/HelloWorld'
5
6
7  function App() {
8    const [count, setCount] = useState(0)
9    return (
10     <>
11       <Header/>
12       <HelloWorld user='Cachupina'/>
13     </>
14   )
15 }
16 export default App
17
```

En el componente hijo:

```
HelloWorld.jsx  App.jsx
src > components > HelloWorld.jsx > HelloWorld
1
2  export function HelloWorld(props){
3    //const nombre='Cachupin'
4    return(
5      <>
6        <div className="container mt-5">
7          <h2>Bienvenido a React {props.user}</h2>
8        </div>
9      </>
10    );
11  }
```

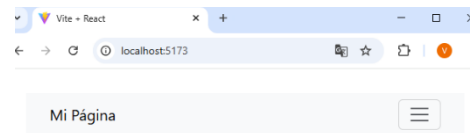
- b. A través de props podemos enviar varios parámetros

```
HelloWorld.jsx  App.jsx  x
src > App.jsx > ...
1 import { useState } from 'react'
2 import './App.css'
3 import Header from './components/Header';
4 import { HelloWorld } from './components/HelloWorld'
5
6
7 function App() {
8   const [count, setCount] = useState(0)
9   return (
10     <>
11       <Header/>
12       <HelloWorld user='Cachupina' id='1' />
13     </>
14   )
15 }
16 export default App
17
18
```

En el componente hijo:

```
loWorld.jsx  App.jsx
components > HelloWorld.jsx > ...
export function HelloWorld(props){
  return(
    <>
      <div className="container mt-5">
        <h2>Bienvenido a React {props.user}, con id= {props.id}</h2>
      </div>
    </>
  );
}
```

- c. Omitiendo props, transformando function en función => (flecha)
Modificamos el componente hijo;



Bienvenido a React Cachupina, con id= 1

```
HelloWorld.jsx  App.jsx
src > components > HelloWorld.jsx > ...
1
2 export const HelloWorld = ({user, id}) =>{
3   return(
4     <>
5       <div className="container mt-5">
6         <h2>Bienvenido a React {user}, con id= {id}</h2>
7       </div>
8     </>
9   );
10 }
11
```