# 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:

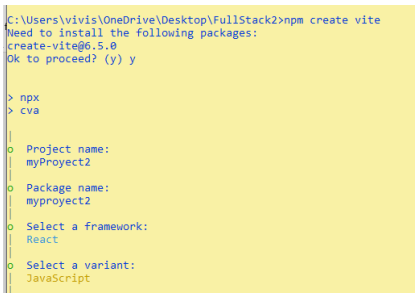
¿Qué necesitamos para trabajar con esta biblioteca?

**Requisitos previos**

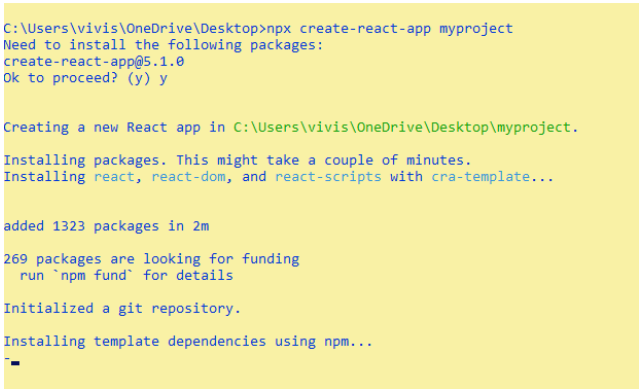
- Conocimientos de Html, CSS y JavaScript.
- Node.js y npm (gestor de paquetes de node). Para instalar https://nodejs.org/
- Verificar instalación: node -v // npm -v

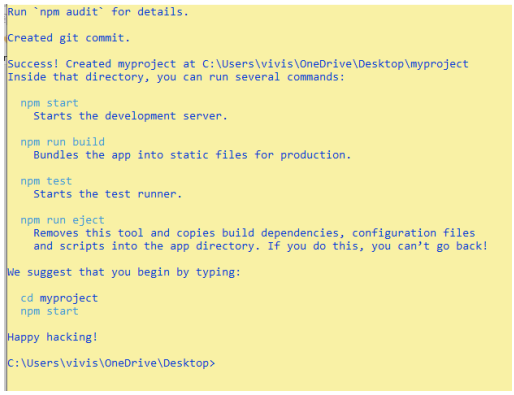
**Comenzamos:**

- 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

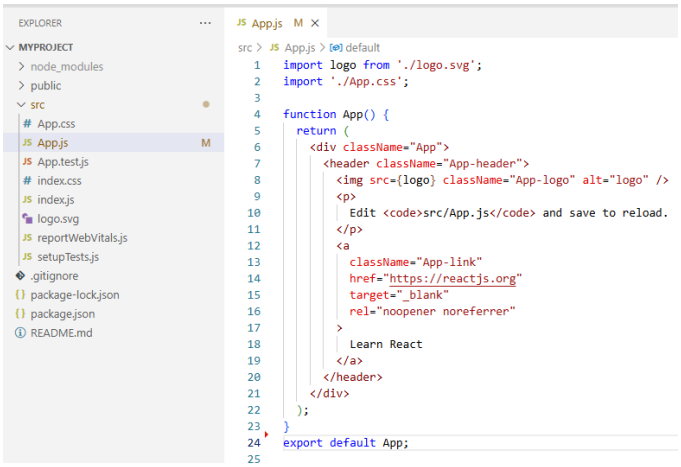



- Si crea un proyecto utilizando create react:
  - npx create-react-app nombre_proyecto
  - cd nombre_proyecto (ingresar a la carpeta)
  - npm start

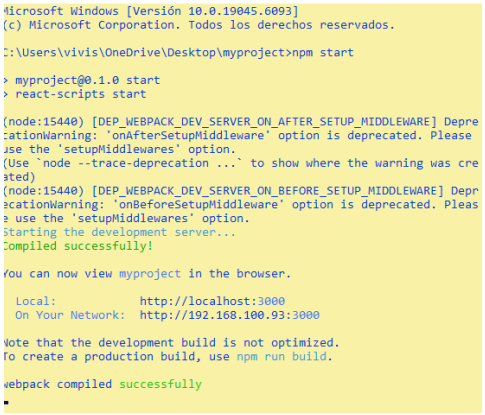


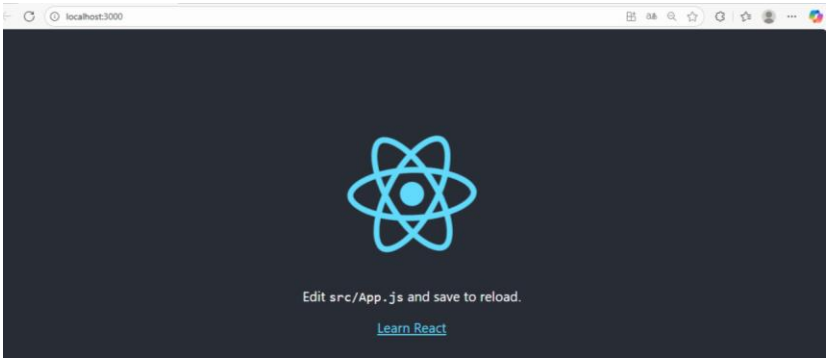

Ingresa a la carpeta de proyecto y en ella encontrarás una carpeta llamada **src**, donde se almacena un archivo llamado **App.js.**




**Iniciar servidor:  npm start**




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

```jsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <>
    <div>
     <h2>Bienvenido a React</h2>
    </div>
    </>
  )
}
export default App
```

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

main.jsx App.jsx

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

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

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 > ...

```jsx
export function HelloWorld(){
    return(
        <div>
            <h1>Hola Mundo</h1>
        </div>);
}
```

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

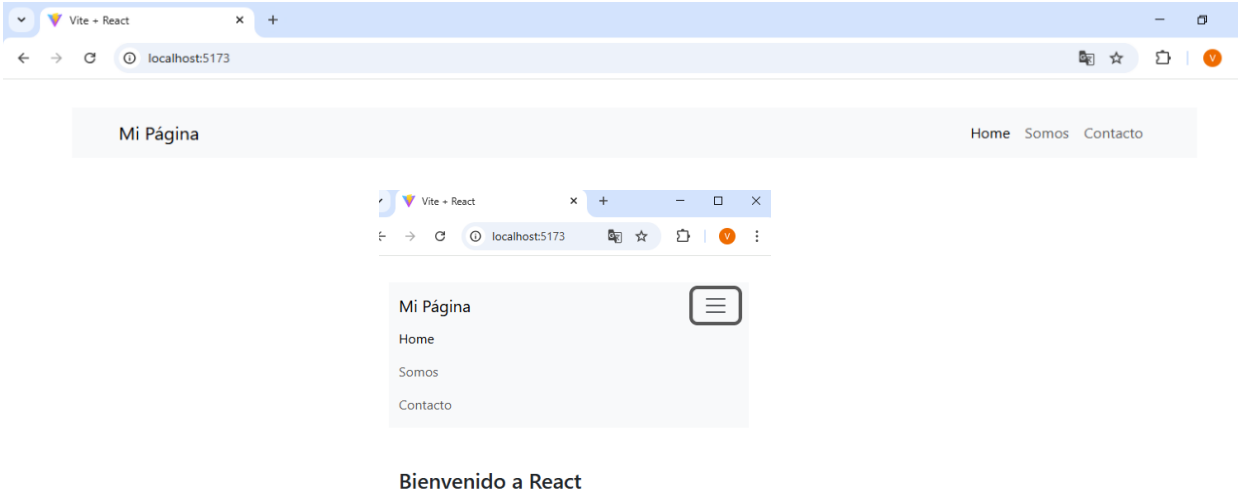
src > main.jsx

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

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

**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

```jsx
import { useState } from 'react'
import './App.css'
import Header from './components/Header';
import { HelloWorld } from './components/HelloWorld'


function App() {
  const [count, setCount] = useState(0)
  return (
    <>
    <Header/>
    <HelloWorld/>
    </>
  )
}
export default App
```

En el archivo main.jsx:

eader.jsx | index.html | App.jsx | main.jsx | HelloWorld.js

main.jsx

```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 > ...

```jsx
export function HelloWorld(props){
  const nombre='Cachupin'
    return(
        <>
            <div className="container mt-5">
                <h2>Bienvenido a React {nombre}</h2>
            </div>
        </>
    );
    }
```

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 > ...

```jsx
import { useState } from 'react'
import './App.css'
import Header from './components/Header';
import { HelloWorld } from './components/HelloWorld'


function App() {
  const [count, setCount] = useState(0)
  return (
    <>
    <Header/>
    <HelloWorld user='Cachupina'/>
    </>
  )
}
export default App

```

**En el componente hijo**:

HelloWorld.jsx | App.jsx

src > components > HelloWorld.jsx > HelloWorld

```jsx
export function HelloWorld(props){
  //const nombre='Cachupin'
    return(
        <>
            <div className="container mt-5">
                <h2>Bienvenido a React {props.user}</h2>
            </div>
        </>
    );
}
```

b. A través de props podemos enviar varios parámetros

HelloWorld.jsx App.jsx

src > App.jsx > ...

```jsx
import { useState } from 'react'
import './App.css'
import Header from './components/Header';
import { HelloWorld } from './components/HelloWorld'


function App() {
  const [count, setCount] = useState(0)
  return (
    <>
    <Header/>
    <HelloWorld user='Cachupina' id='1'/>
    </>
  )
}
export default App
```

**En el componente hijo:**

loWorld.jsx App.jsx

components > HelloWorld.jsx > ...

```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;

HelloWorld.jsx App.jsx

src > components > HelloWorld.jsx > ...

```jsx

export const HelloWorld =({user, id}) =>{
    return(
        <>
            <div className="container mt-5">
                <h2>Bienvenido a React {user}, con id= {id}</h2>
            </div>
        </>
    );
}

```