



# React nivel básico

Facilitador: Ing. Henry A Duque A  
e-mail: henryaduquea@gmail.com

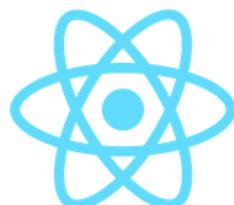
# Contenido

## React Inicio

- 1.1 - ¿Qué es React?
- 1.2 - Virtual DOM
- 1.3 - Pensando en componentes
- 1.4 - Instalar la primera app
- 1.5 - Dependencias
- 1.6 - Scripts
- 1.7 - Recomendaciones

## JSX

- 2.1 - Integrando
- 2.2 - ¿Qué es JSX?
- 2.3 - Restricciones JSX
- 2.4 - JSX a profundidad



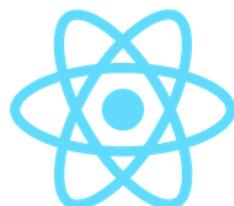
# Contenido

## Componentes

- 3.1 - Expresiones JSX
- 3.2 - Propiedades
- 3.3 - Imprimiendo propiedades
- 3.4 - Componentes presentacionales
- 3.5 - propsTypes y defaultProps
- 3.6 - Recorriendo arrays

## El estado de los componentes

- 4.1 - Creando un componente de clase
- 4.2 - Formulario
- 4.3 - Estado del componente
- 4.4 - Actualizando el estado
- 4.5 - Escribir métodos
- 4.6 - Ciclo de vida de montaje
- 4.7 - Ciclo de vida de actualización
- 4.8 - Ciclo de vida de desmontaje
- 4.9 - Consejos finales



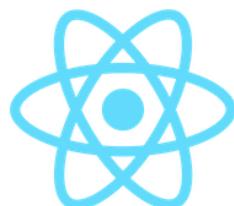
# Contenido

## React Router

- 5.1 - Introducción e instalación de react router
- 5.2 - Declarando Router y Rutas
- 5.3 - Switch y página (error 404)
- 5.4 - Parámetros de la ruta
- 5.5 - Contenido dinámico a partir de la ruta
- 5.6 - Componente Link
- 5.7 - Menú con NavLink
- 5.8 - Props de React Router

## Peticiones HTTP

- 6.1 - Atomic Design
- 6.2 - Refactorizando código
- 6.3 - Metodología atomic design
- 6.4 - Peticiones a una API REST
- 6.5 - Peticiones con fetch
- 6.6 - Actualizando el estado con la respuesta
- 6.7 - Reescribiendo peticiones con axios



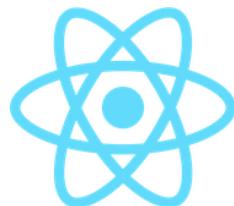
# Contenido

## Componentes de orden superior

- 7.1 - Crear servidor JSON
- 7.2 - Agregando loader a nuestras peticiones
- 7.3 - Eliminando inconsistencias en el proyecto
- 7.4 - Escribiendo un componente de orden superior
- 7.5 - Agregando lógica reutilizable al HOC
- 7.6 - Bonus: Ejemplo de páginas privadas
- 7.7 - Bonus: Subir imágenes con un HOC

## Hooks

- 8.1 - Reescribiendo componentes para utilizar el estado
- 8.2 - Utilizando el hook useState
- 8.3 - Actualizando el estado con hooks
- 8.4 - Utilizando el hook useEffect
- 8.5 - Componentes con múltiples estados
- 8.6 – Escribir un custom hook
- 8.7 - Bonus: Custom hook useFetch
- 8.8 - Bonus: Custom hook useCounter



# React Inicio

1.1 - ¿Qué es React?

1.2 - Virtual DOM

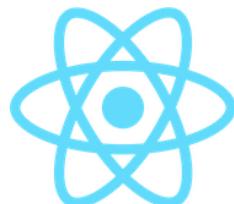
1.3 - Pensando en componentes

1.4 - Instalar la primera app

1.5 - Dependencias

1.6 - Scripts

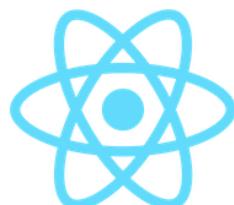
1.7 - Recomendaciones



# 1.1 - ¿Qué es React?

React (también llamada React.js o ReactJS) es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones web. Su mantenimiento depende de Facebook y la comunidad de software libre. Entonces:

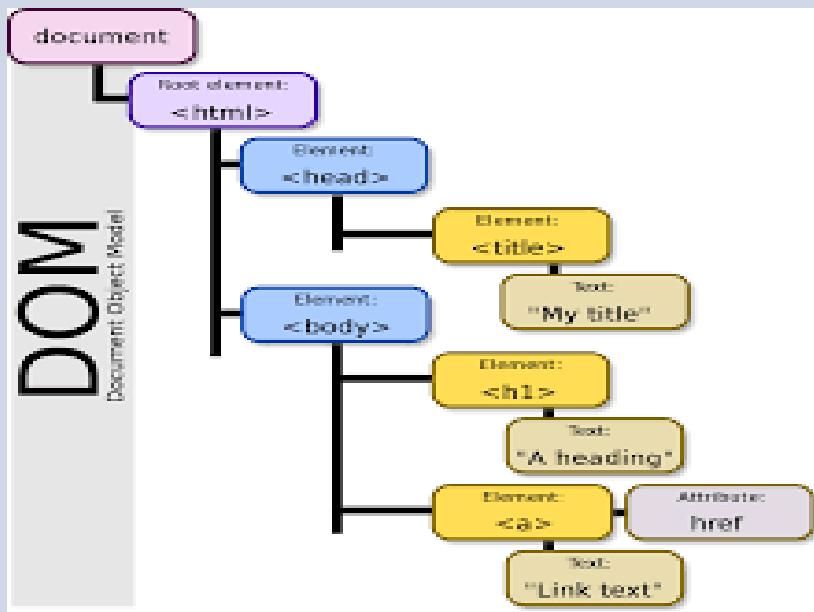
- ✓ React es un proyecto de código abierto creado por Facebook.
- ✓ React se usa para construir interfaces de usuario (UI) en el front-end.
- ✓ React es la capa de vista de una aplicación MVC (Model View Controller)



# 1.2 - Virtual DOM

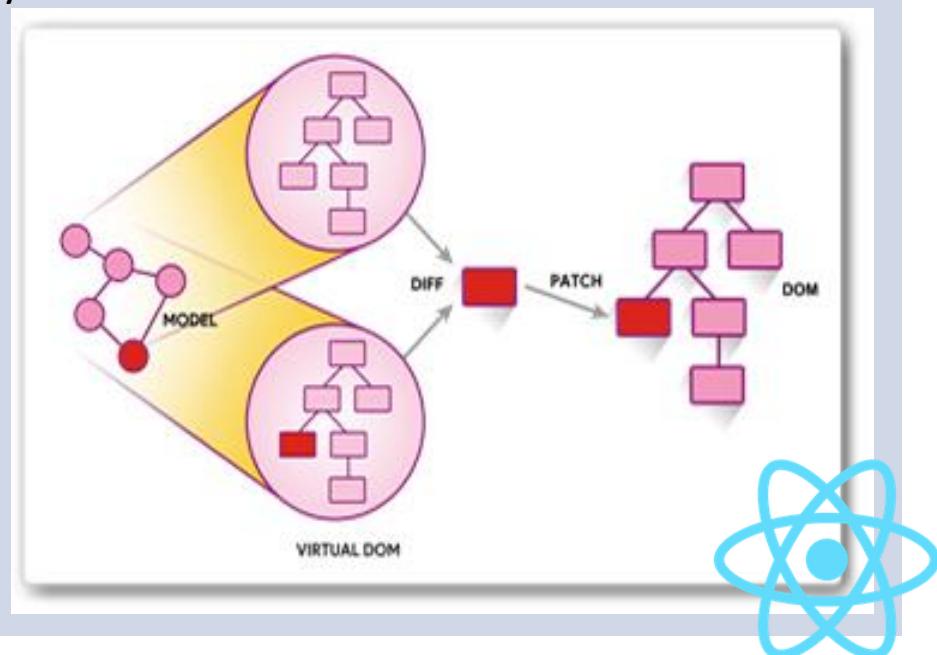
## ¿Qué es el DOM?

El Modelo de Objeto de Documentos (DOM), es una interface de programación para documentos HTML. Facilita una representación estructurada del documento y define de qué manera se puede acceder y modificar su estructura, contenido, estilo y contenido.



## ¿Qué es el DOM virtual?

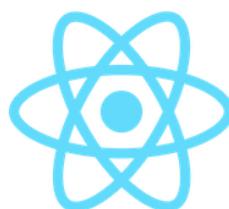
El DOM virtual (VDOM) es un concepto de programación donde una representación ideal o “virtual” de la IU se mantiene en memoria y en sincronía con el DOM “real”, mediante una biblioteca como ReactDOM. Este proceso se conoce como reconciliación y se hace de manera automática.



# 1.3 - Pensando en componentes

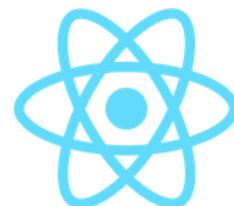
Los componentes permiten separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada.

Conceptualmente, los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas “props”) y devuelven a React elementos que describen lo que debe aparecer en la pantalla.



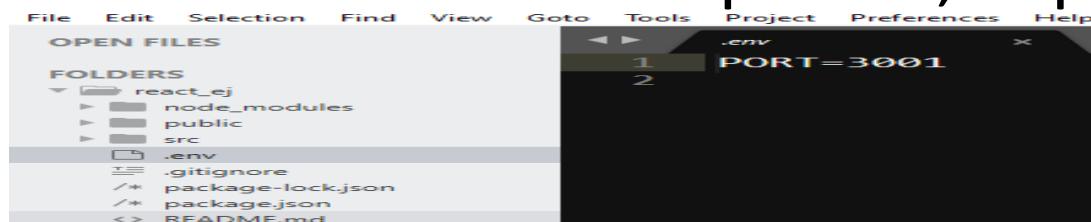
# 1.4 - Instalar la primera app

- Descargar nodejs, desde la página nodejs.org y elegir la opción LTS.
- Seleccionar el Sistema Operativo, sobre el cual será instalado.
- Una vez realizado, verificar la versión instalada de node: `>node --version`
- Posteriormente, verificar que el instalador de paquetes se instaló correctamente: `>npm --version`
- Crear la aplicación react, ejecutando el comando:  
`>npx create-react-app react_ej01` (debe ser escrito en minúscula)
- Al finalizar, muestra comandos básicos para ejecutar la aplicación.

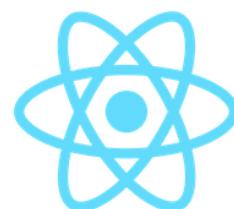


# 1.4 - Instalar la primera app (preparación de ambiente de desarrollo)

- A fin de que cada alumno, tenga control de la publicación de su respectiva aplicación, debe crear el archivo “.env”, a través de sublime text. Este archivo, se encontrará ubicado en el directorio raíz del proyecto. El contenido de este archivo, será el parámetro PORT=número de puerto, que será utilizado.



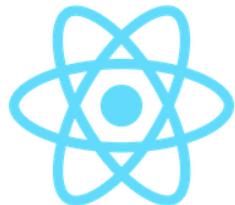
- Acceder al directorio de la aplicación: `>cd react_ej01`
- Ejecutar la aplicación: `>npm start`



# 1.4 - Instalar la primera app (preparación de ambiente de desarrollo)

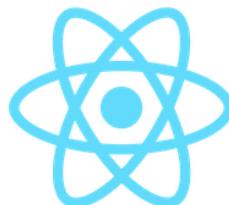
- Desde Sublime Text, para Windows:
  - Oprimir: ctrl+shift+p, (Opción Tools, Command Palette).
  - Escriba “install”, seleccione “Package Control: Install Package”.
  - Escriba “React”, seleccione “ReactJS”.
- Hay una extensión llamada React Developer Tools que te hará la vida mucho más fácil cuando trabajes con React. Descargue [\*\*React DevTools para Chrome\*\*](#)
- Para evitar errores de enrutamiento, ejecutar:

```
>npm install react-router-dom
```



# 1.5 Dependencias

- A fin de entender las dependencias, se efectuará un primer proyecto, para el cual se creará el directorio: \react\_ej00.
- Dentro de este directorio se desarrollará el archivo: index.html. Para esto se utilizará sublime text, elija la opción: File, New File y en la esquina inferior derecha de este editor, elija HTML.
- Elija la opción: File, Save as: **index** (la extensión por omisión es html).
- Desarrolle el siguiente contenido:



# 1.5 Dependencias (continuación)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>React - Ejemplo</title>
6     <!--
7         Vamos a cargar tres CDN en head- React, React DOM y Babel
8     -->
9     <!-- React -->
10    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
11    <!-- React DOM -->
12    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
13    <!-- Babel -->
14    <script src="https://unpkg.com/babel-standalone@6.26.0/babel.js"></script>
15 </head>
```

React

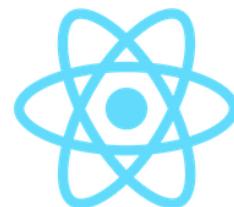
<https://unpkg.com/react@15.3.2/dist/react.js>

React DOM

<https://unpkg.com/react-dom@15.3.2/dist/react-dom.js>

Babel

<https://unpkg.com/babel-core@5.8.38/browser.min.js>



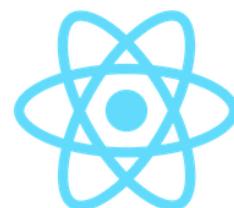
# 1.5 Dependencias (continuación)

```
22 <body>
23   <div id="root"></div>
24   <script type="text/babel">
25     class App extends React.Component{
26       render(){
27         return <h1>Hola React Mis Saludos!!!</h1>
28       }
29     }
30     ReactDOM.render(<App/>, document.getElementById('root'));
31   </script>
32 </body>
33 </html>
```

- Oprima el botón derecho del mouse y elija “Open in browser”.
- Observe el resultado:

← → ⌂ C:/react\_ej00/index.html

Hola React Mis Saludos!!!



# 1.5 Dependencias (continuación)

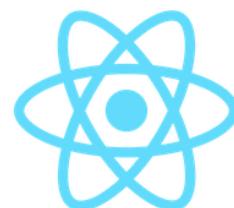
- Considere efectuar esta actualización. Guarde los cambios como **index1.html**:

```
25  </head>
26  <body>
27      <div id="salida"></div>
28      <script type="text/babel">
29          var contenido = <h1>Hola Mundo, bienvenidos a ReactJS!!</h1>
30          var salida = document.getElementById('salida');
31          ReactDOM.render(contenido, salida);
32      </script>
33  </body>
34  </html>
```

- Oprima el botón derecho del mouse y elija “Open in browser”.
- Observe el resultado:

← → C C:/react\_ej00/index1.html

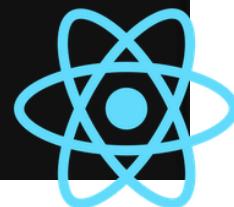
Hola Mundo, bienvenidos a ReactJS!!



# 1.5 Dependencias (continuación)

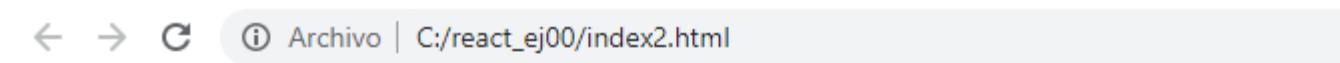
- Considere efectuar esta actualización. Guarde los cambios como **index2.html**:

```
14 <body>
15     <div id="salida"></div>
16     <script type="text/babel">
17         var MiComponente = React.createClass({
18             render:function(){
19                 return(
20                     <div>
21                         <h1>Componente creado con createClass()</h1>
22                         <p>Este componente fue creado con createClass(), como una prueba</p>
23                     </div>
24                 );
25             }
26         });
27         ReactDOM.render(
28             <div>
29                 <MiComponente/>
30                 <MiComponente/>
31                 </div>,
32                 salida);
33         </script>
34     </body>
```



# 1.5 Dependencias (continuación)

- Oprima el botón derecho del mouse y elija “Open in browser”.
- Observe el resultado:

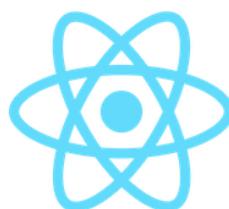


## Componente creado con createClass()

Este componente fue creado con createClass(), como una prueba

## Componente creado con createClass()

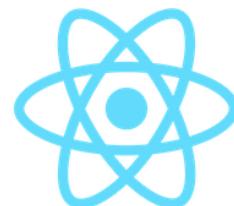
Este componente fue creado con createClass(), como una prueba



# 1.5 Dependencias (continuación)

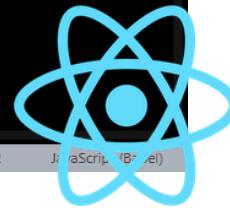
Es el momento de configurar sublime text, para destacar la notación contenida en un proyecto React:

- Seleccione la opción View->Sintax->Open all with current extension as...->Babel->Java Script(Babel).
- Puede elegir una opción de color, seleccionando Preferences->Color Scheme y de las opciones sugeridas seleccione, alguna de las que incluyen “Babel”. Observe el resultado.



# 1.5 Dependencias (continuación)

Ahora volviendo al proyecto \react\_ej01, se observará su estructura de directorios:



C:\react\_ej01\public\index.html (react\_ej01) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

OPEN FILES

FOLDERS

- react\_ej01
- node\_modules
- public
  - favicon.ico
  - index.html
  - logo192.png
  - logo512.png
  - manifest.json
  - robots.txt
- src
  - App.css
  - App.js
  - App.test.js
  - index.css
  - index.js
  - logo.svg
  - pruebajs
  - serviceWorker.js
  - setupTests.js
- .gitignore
- package-lock.json
- package.json
- README.md

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
```

Line 1, Column 1

21%

Spaces: 2

JavaScript (Babel)

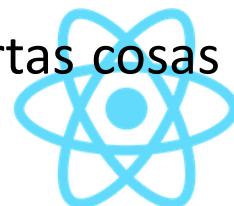
# 1.5 Dependencias (continuación)

El listado de nuestra app recién creada es bastante sencillo. Observarás que tenemos varias carpetas:

- node\_modules: con las dependencias npm del proyecto
- public: esta es la raíz de nuestro servidor donde se podrá encontrar el index.html, el archivo principal y el favicon.ico que sería el icono de la aplicación.
- src: aquí es donde vamos a trabajar principalmente para nuestro proyecto, donde vamos a colocar los archivos de nuestros componentes React.

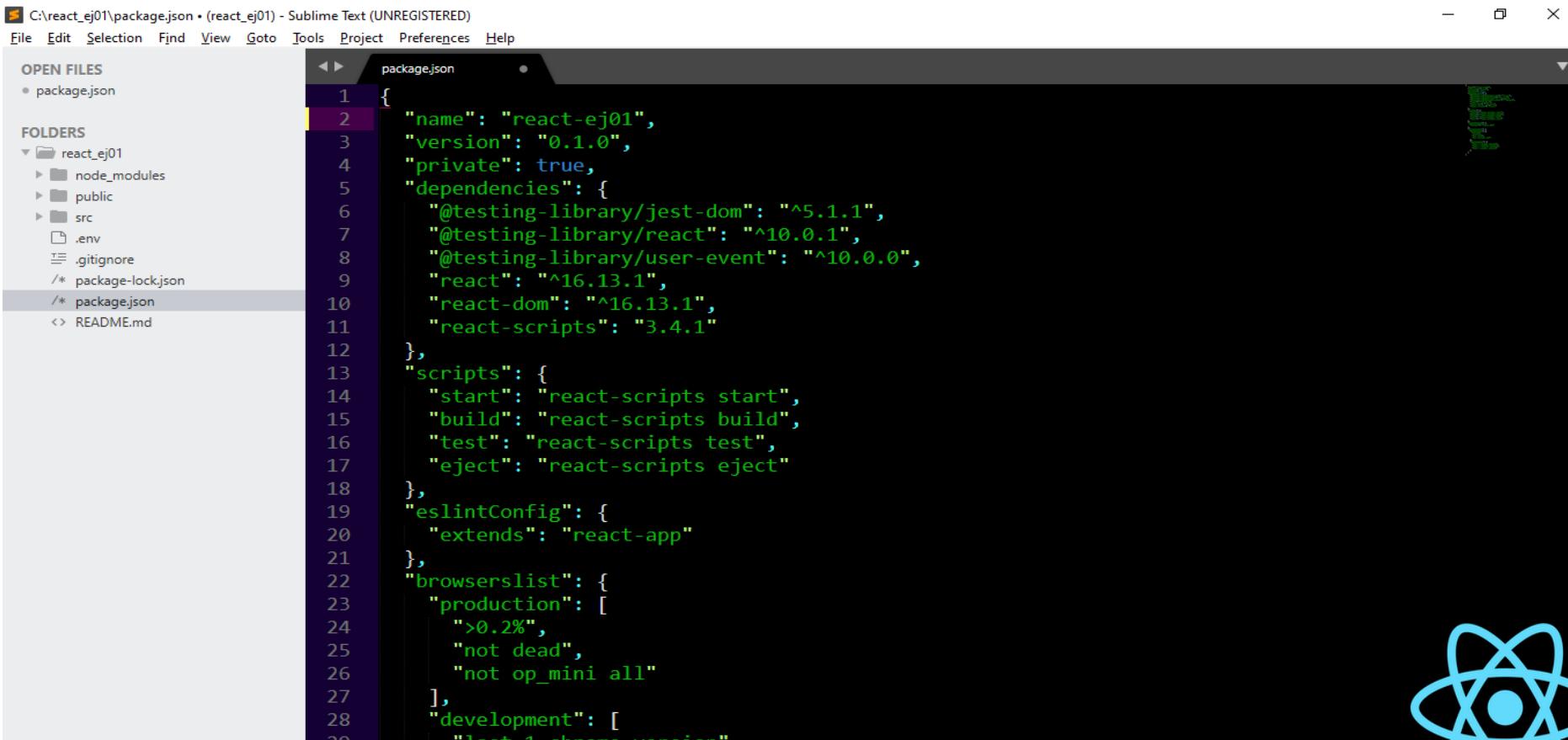
Además encontrarás archivos sueltos como:

- README.md que es el readme de Create React App, con cantidad de información sobre el proyecto y las apps que se crean a partir de él.
- package.json, que contiene información del proyecto, así como enumera las dependencias de npm, tanto para desarrollo como para producción. Si conoces npm no necesitarás más explicaciones.
- .gitignore que es el típico archivo para decirle a git que ignore ciertas cosas del proyecto a la hora de controlar el versionado del código.



# 1.5 Dependencias (continuación)

- Estas dependencias se encuentran contenidas en el archivo package.json



The screenshot shows the Sublime Text editor interface with the following details:

- Title Bar:** C:\react\_ej01\package.json • (react\_ej01) - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find Goto Tools Project Preferences Help
- Left Panel (OPEN FILES):** package.json
- Left Panel (FOLDERS):** react\_ej01, node\_modules, public, src, .env, .gitignore, package-lock.json, package.json, README.md
- Center Panel (Code View):** The code editor displays the contents of the package.json file. The code is color-coded for syntax highlighting, with green for strings and numbers, and blue for object keys.
- Bottom Status Bar:** Line 2, Column 22, 88%, Spaces: 2, a small icon, and a blue circular logo with concentric lines.

```
1 {  
2   "name": "react-ej01",  
3   "version": "0.1.0",  
4   "private": true,  
5   "dependencies": {  
6     "@testing-library/jest-dom": "^5.1.1",  
7     "@testing-library/react": "^10.0.1",  
8     "@testing-library/user-event": "^10.0.0",  
9     "react": "^16.13.1",  
10    "react-dom": "^16.13.1",  
11    "react-scripts": "3.4.1"  
12  },  
13  "scripts": {  
14    "start": "react-scripts start",  
15    "build": "react-scripts build",  
16    "test": "react-scripts test",  
17    "eject": "react-scripts eject"  
18  },  
19  "eslintConfig": {  
20    "extends": "react-app"  
21  },  
22  "browserslist": {  
23    "production": [  
24      ">0.2%",  
25      "not dead",  
26      "not op_mini all"  
27    ],  
28    "development": [  
29      "last 1 chrome version"  
30    ]  
31  }  
32}  
33
```

# 1.5 Dependencias (continuación)

- Dichas dependencias pueden ser actualizadas ejecutando el comando: **>npm install -g npm-check-updates**

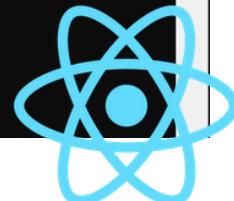
```
Select Command Prompt
C:\react_ej01>npm install -g npm-check-updates
npm WARN deprecated mkdirp@0.5.4: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note
that the API surface has changed to use Promises in 1.x.)
C:\Users\hduqu\AppData\Roaming\npm\npm-check-updates -> C:\Users\hduqu\AppData\Roaming\npm\node_modules\npm-check-updates\bin\npm-check-updates
C:\Users\hduqu\AppData\Roaming\npm\ncu -> C:\Users\hduqu\AppData\Roaming\npm\node_modules\npm-check-updates\bin\ncu
+ npm-check-updates@4.0.5
added 234 packages from 91 contributors in 92.386s

[=====] 6/6 100%
Checking C:\react_ej01\package.json
@testing-library/jest-dom    ^4.2.4  →  ^5.1.1
@testing-library/react       ^9.5.0  →  ^10.0.1
@testing-library/user-event   ^7.2.1  →  ^10.0.0
react-scripts                 3.4.0  →  3.4.1

Run ncu -u to upgrade package.json

[=====] 6/6 100%
Upgrading C:\react_ej01\package.json
@testing-library/jest-dom    ^4.2.4  →  ^5.1.1
@testing-library/react       ^9.5.0  →  ^10.0.1
@testing-library/user-event   ^7.2.1  →  ^10.0.0
react-scripts                 3.4.0  →  3.4.1

Run npm install to install new versions.
```



# 1.6 Scripts

- Los scripts (programas en javascript), en los que se fijará inicialmente la atención son los archivos: App.js



A screenshot of Sublime Text showing the code for `App.js`. The code imports React and a logo from an SVG file, defines a functional component `App` that returns a header with a logo and a link to the official React website, and exports it. The Sublime Text interface includes a sidebar with project files and a status bar at the bottom.

```
C:\react_ej01_init\src\App.js (react_ej01_init) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

OPEN FILES
FOLDERS
  react_ej01_init
    node_modules
    public
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    prueba.js
    serviceWorker.js
    setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md

App.js
1 import React from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 function App() {
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          Edit <code>src/App.js</code> and save to reload.
12        </p>
13        <a
14          className="App-link"
15          href="https://reactjs.org"
16          target="_blank"
17          rel="noopener noreferrer"
18        >
19          Learn React
20        </a>
21      </header>
22    </div>
23  );
24}
25
26 export default App;
27
```

Line 1, Column 1      25%      Spaces: 2      JavaScript (Babel)

# 1.6 Scripts (continuación)

- Y el archivo: Index.js

S C:\react\_ej01\_init\src\index.js (react\_ej01\_init) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

OPEN FILES

FOLDERS

- react\_ej01\_init
  - node\_modules
  - public
  - src
    - /\* App.css
    - /\* App.js
    - /\* App.test.js
    - /\* index.css
    - /\* index.js
    - <> logo.svg
    - /\* prueba.js
    - /\* serviceWorker.js
    - /\* setupTests.js
  - .gitignore
  - /\* package-lock.json
  - /\* package.json
  - <> README.md

index.js

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';

6 ReactDOM.render(<App />, document.getElementById('root'));

7 // If you want your app to work offline and load faster, you can change
8 // unregister() to register() below. Note this comes with some pitfalls.
9 // Learn more about service workers: https://bit.ly/CRA-PWA
10 serviceWorker.unregister();

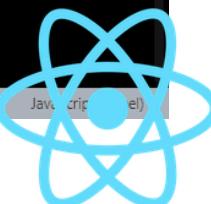
11
12
13
```

Line 1, Column 1

57%

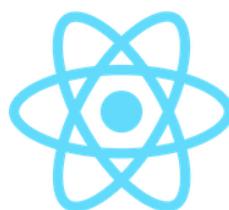
Tab Size: 4

JavaScript (ES6)



# 1.7 Recomendaciones

- Se debe estar familiarizado con HTML y JavaScript. Así mismo manejar conceptos de programación como funciones, objetos, arrays, y en menor medida clases.



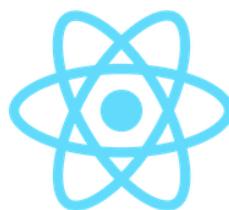
# JSX

2.1 - Integrando

2.2 - ¿Qué es JSX?

2.3 - Restricciones JSX

2.4 - JSX a profundidad

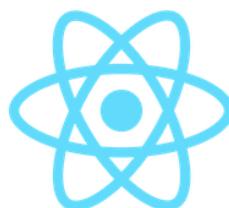


## 2.1 - Integrando

JSX, puede integrarse con Primitive, Bootswatch, Bootstrap, EDgrid y otros. Los cuales son librerías construidas con Sass, CSS y jQuery, para Responsive Web Design (RWD). Son ligeros, personalizables, permitiendo prototipar y crear diseños (layouts) en muy poco tiempo sin conflictos con su proyecto. Ejemplos:

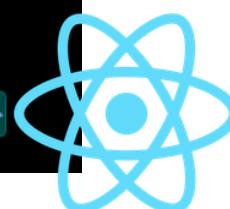
<https://taniarascia.github.io/primitive>

<https://www.bootstrapcdn.com/bootswatch/>



## 2.1 – Integrando (continuación)

Sobre el proyecto, contenido en el directorio \react\_ej01, guarde index.html, como index1.html, posteriormente, acceda al archivo \public\index.html e incluya el siguiente enlace, a fin de definir el estilo de este ejemplo:



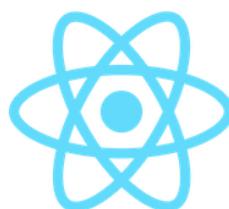
A screenshot of a code editor showing the file "index.html". The left sidebar shows "OPEN FILES" with "index.html" selected, and "FOLDERS" with "react\_ej01" expanded, showing "node\_modules", "public" (containing "favicon.ico"), "index.html", "logo192.png", "logo512.png", "manifest.json", and "robots.txt". The main editor area shows the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <!-- primitive.css -->
    <link rel="stylesheet" type="text/css" href="https://taniarascia.github.io/primitive/css/main.css" />
```

## 2.2 - ¿Qué es JSX?

JSX es una extensión de JavaScript creada por Facebook para el uso con su librería React. Sirve de preprocesador y transforma el código a JavaScript.

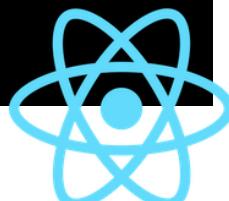
Guarde previamente index.js, como index1.js, posteriormente, proceda a actualizar el archivo: /src/index.js, con el siguiente contenido:



# 2.2 - ¿Qué es JSX? (continuación)

Realizar modificaciones indicadas en recuadros:

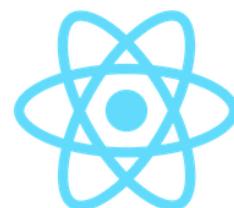
```
index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 //import App from './App';
5 import * as serviceWorker from './serviceWorker';
6
7 const nombre = 'HENRY';
8 const encabezado = <h1 className='site-heading'>{nombre}, otro saludo, desde React!!</h1>;
9
10 ReactDOM.render(
11   encabezado,
12   document.getElementById('root')
13 );
14
15 // If you want your app to work offline and load faster, you can change
16 // unregister() to register() below. Note this comes with some pitfalls.
17 // Learn more about service workers: https://bit.ly/CRA-PWA
18
19 serviceWorker.unregister();
20
```



## 2.2 - ¿Qué es JSX? (continuación)

Guarde App.js como App1.js, posteriormente proceda a modificar el archivo /src/App.js, con el siguiente contenido:

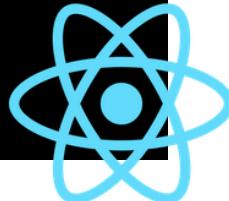
```
App.js
1 import React, {Component} from 'react';
2
3 class App extends Component{
4   render(){
5     return (
6       <div className="App">
7         <h1>Este es el nuevo saludo desde, React!</h1>
8       </div>
9     )
10  }
11}
12
13 export default App;
```



## 2.2 - ¿Qué es JSX? (continuación)

Ahora proceda a guardar Index.js como Index1.js, posteriormente proceda a modificar el archivo /src/Index.js, con el siguiente contenido:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4 import './index.css';
5
6 /*
7 import * as serviceWorker from './serviceWorker';
8 */
9
10 ReactDOM.render(<App />, document.getElementById('root'));
11
12 // If you want your app to work offline and load faster, you can change
13 // unregister() to register() below. Note this comes with some pitfalls.
14 // Learn more about service workers: https://bit.ly/CRA-PWA
15
16 /* serviceWorker.unregister(); */
```

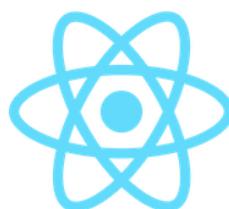


## 2.2 - ¿Qué es JSX? (continuación)

Se está usando lo que parece HTML en el código React, pero no es exactamente HTML. Este es JSX , que significa JavaScript XML.

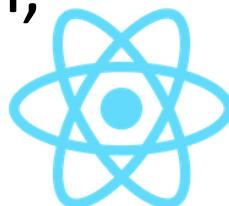
Con JSX, puede escribir lo que parece HTML, y también puede crear y usar etiquetas propias tipo XML.

Así es como se ve JSX asignado a una variable.



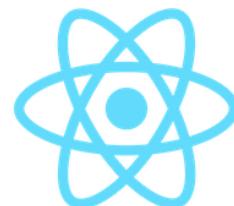
## 2.3 - Restricciones JSX

- Siempre los elementos HTML deben tener su marca de comienzo y fin, y en el caso que solo tengan una etiqueta que es tanto de comienzo como fin debemos agregar el carácter '/'.
- Se recomienda dividir el JSX en varias líneas para facilitar la lectura, también recomiendan envolverlo entre paréntesis.
- Para definir valores a las propiedades de un elementos HTML mediante expresiones no debemos colocar comillas.
- Una restricción de JSX es que no puedes utilizar if, else, while o for.



## 2.4 - JSX a profundidad

- `className` se usa en lugar de `class` agregar clases CSS, como `classes` una palabra clave reservada en JavaScript.
- Las propiedades y métodos en JSX son camelCase - `onclick` se convertirán `onClick`.
- Las etiquetas de cierre automático deben terminar en una barra inclinada, p. Ej.`<img />`
- Las expresiones de JavaScript también se pueden incrustar dentro de JSX utilizando llaves, incluidas variables, funciones y propiedades.



# Componentes

3.1 - Expresiones JSX

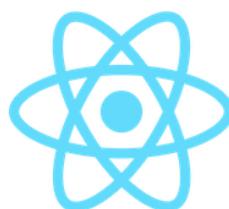
3.2 - Propiedades

3.3 - Imprimiendo propiedades

3.4 - Componentes presentacionales

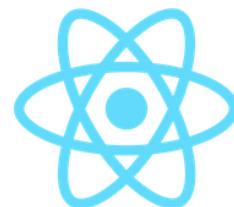
3.5 - propTypes y defaultProps

3.6 - Recorriendo arrays



## 3.1 - Expresiones JSX

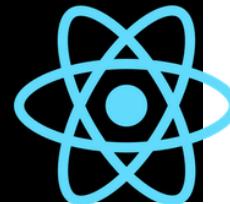
En el archivo `src/App.js`, se observa que la mayoría del código que aparece es como si fuera HTML, aunque encerrado dentro de un script Javascript. Es la manera en la que ReactJS trabaja con las vistas de la aplicación. En realidad no es código HTML sino "JSX", una extensión a la sintaxis de Javascript que permite de una manera amigable crear y mantener el HTML que se necesita para "renderizar" (pintar) los componentes.



# 3.1 - Expresiones JSX (continuación)

Guarde el archivo src/App.js como src/App2.js y realice la siguiente actualización, sobre el archivo src/App.js:

```
1 import React, {Component} from 'react';
2
3 class App extends Component{
4     render(){
5         return (
6             <div className="App">
7                 <h1>Este es el nuevo saludo desde, React!</h1>
8                 <OtroSaludo />
9             </div>
10        );
11    }
12}
13
14 class OtroSaludo extends Component{
15     render(){
16         return(
17             <p>Hola, este es un saludo desde otro componente</p>
18         );
19    }
20}
21
22 export default App;
```



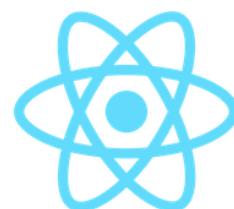
# 3.1 - Expresiones JSX (continuación)

Al ejecutar observe el resultado::

← → C ⓘ localhost:3001

**Este es el nuevo saludo desde, React!**

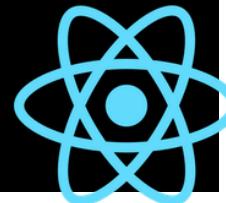
Hola, este es un saludo desde otro componente



## 3.2 - Propiedades

Las propiedades son atributos de la clase, los cuales se pueden acceder a través de su instancia u objeto. Por ejemplo, guarde el archivo src/App.js como src/App3.js y realice la siguiente actualización, sobre el archivo src/App.js. Al ejecutar observe el resultado :

```
1 import React, {Component} from 'react';
2 class App extends Component{
3     render(){
4         return (
5             <div className="App">
6                 <h1>Este es el nuevo saludo desde, React!</h1>
7                 <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>
8             </div>
9         );
10    }
11 }
12 class OtroSaludo extends Component{
13     render(){
14         return(
15             <p>Hola, este es un saludo desde otro componente:
16                 {this.props.nombre} y {this.props.app}
17             </p>
18         );
19    }
20 }
21 export default App;
```



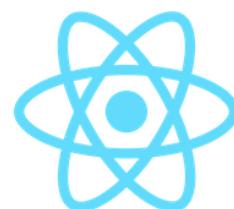
## 3.2 – Propiedades (continuación)

Como se puede observar el valor contenido, en las propiedades definidas se mostrarán en la salida:

← → ⌂ ⓘ localhost:3001

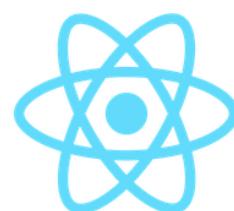
**Este es el nuevo saludo desde, React!**

Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS



### 3.3 – Imprimiendo propiedades

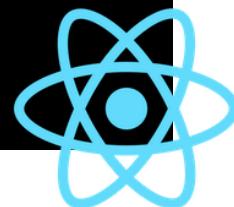
Como se observó en el ejemplo anterior las propiedades para ser impresas, se requiere acceder a las mismas a través del atributo {this.props.nombre}, esta propiedad debe ser colocada entre llaves {}. Así mismo pueden, ser definidas variables e impresas solo empleando llaves {} sin emplear this.props. Para el siguiente ejemplo, guarde el archivo src/App.js como src/App4.js e ingrese el siguiente código:



# 3.3 – Imprimiendo propiedades (continuación)

Actualizar en src/App.js:

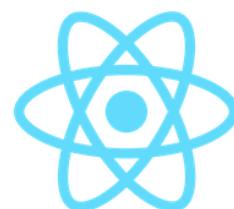
```
1 import React, {Component} from 'react';
2 class App extends Component{
3     render(){
4         return (
5             <div className="App">
6                 <h1>Este es el nuevo saludo desde, React!</h1>
7                 <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>
8                 <TimestampToDate timestamp = {1585258546000}/>
9             </div>
10        );
11    }
12 }
13 class OtroSaludo extends Component{
14     render(){
15         return(
16             <p>Hola, este es un saludo desde otro componente:
17                 {this.props.nombre} y {this.props.app}
18             </p>
19         );
20     }
21 }
22 class TimestampToDate extends Component{
23     render(){
24         var date = new Date(parseInt(this.props.timestamp,10));
25         var fecha = date.getDate()+'/'+(date.getMonth()+1)+'/'+date.getFullYear();
26         return(
27             <span>{fecha}<br/></span>
28         );
29     }
30 }
31 export default App;
```



## 3.4 - Componentes presentacionales

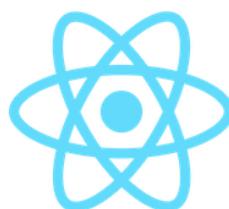
Existen dos tipos de componentes, con estado (statefull components) y sin estado (stateless components).

- Componentes sin estado: Los componentes sin estado no guardan ninguna información y por ello no necesitan de datos locales.
- Componentes con estado: Los componentes con estado son aquellos que almacenan datos de manera local al componente. Estos datos pueden variar a lo largo del tiempo bajo diversas circunstancias, por ejemplo por la interacción del usuario con el componente.



## 3.4 - Componentes presentacionales (continuación)

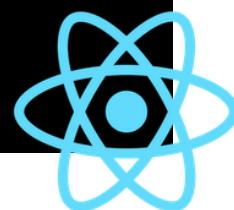
Los componentes desarrollados hasta el momento han sido todos sin estado (stateless component). Siguiendo el mismo procedimiento hasta el momento, guarde las actualizaciones realizadas como src/App4.js y realice la siguiente actualización sobre src/App.js:



# 3.4 - Componentes presentacionales (continuación)

Componente sin estado (stateless component) :

```
33 class MostrarFechaHora extends Component{
34   render(){
35     var date = new Date();
36     var aa = date.getFullYear();
37     var mm = (date.getMonth()+1);
38     var dd = date.getDate();
39     var hr = date.getHours();
40     var mi = date.getMinutes();
41     var se = date.getSeconds();
42     var meridiano = 'am';
43     if (mm<10){
44       mm='0'+mm;
45     }
46     if (dd<10){
47       dd='0'+dd;
48     }
49     if (hr>12){
50       hr = hr-12;
51       meridiano = 'pm';
52     }
53     if (mi<10){
54       mi='0'+mi;
55     }
56     if (se<10){
57       se='0'+se;
58     }
59     var fechaHora = aa+'/'+mm+'/'+dd+' '+hr+':'+mi+':'+se+' '+meridiano;
60     return(
61       <span>FECHA Y HORA ACTUAL:{fechaHora}<br/></span>
62     );
63   }
64 }
```

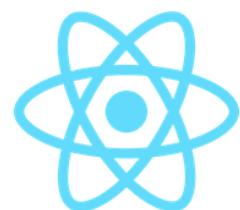


# 3.4 - Componentes presentacionales (continuación)

Componente sin estado (stateless component) :

```
65 class GenerarSerie extends Component{  
66   render(){  
67     var resultado = '';  
68     for (var i = 0; i < 10; i++) {  
69       resultado = resultado+i+',';  
70     }  
71     return(  
72       <span>{resultado}<br/></span>  
73     );  
74   }  
75 }
```

```
1 import React, {Component} from 'react';  
2 class App extends Component{  
3   render(){  
4     return (  
5       <div className="App">  
6         <h1>Este es el nuevo saludo desde, React!</h1>  
7         <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>  
8         <TimestampToDate timestamp = {1585258546000}/>  
9           <MostrarFechaHora />  
10          <GenerarSerie />  
11        </div>  
12      );  
13    }  
14 }
```



## 3.4 - Componentes presentacionales (continuación)

Componente sin estado (stateless component) :

← → ⌂ ⓘ localhost:3001

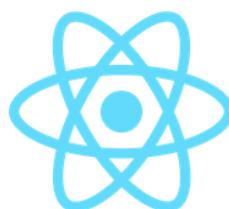
# Este es el nuevo saludo desde, React!

Hola, este es un saludo desde otro componente: HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020

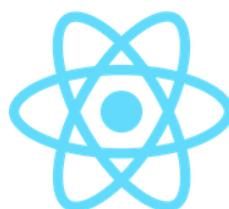
FECHA Y HORA ACTUAL: 2020/03/27 8:43:49 pm

0,1,2,3,4,5,6,7,8,9,



## 3.4 - Componentes presentacionales (continuación)

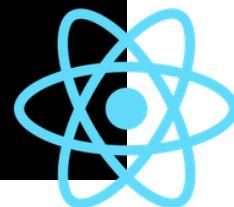
Ahora se procederá a desarrollar un ejemplo de un componente con estado (statefull component). Siguiendo el mismo procedimiento hasta el momento, guarde las actualizaciones realizadas como src/App5.js y realice la siguiente actualización sobre src/App.js:



# 3.4 - Componentes presentacionales (continuación)

Componente con estado (statefull component) :

```
77  class Contador extends Component{  
78      constructor(...args){  
79          super(...args)  
80          this.state = {  
81              contador : 0  
82          }  
83      }  
84      incrementar(){  
85          this.setState({  
86              contador: this.state.contador + 1  
87          })  
88      }  
89      render(){  
90          return(  
91              <div>  
92                  <span>Cuenta actual: {this.state.contador}</span>  
93                  <button onClick={this.incrementar.bind(this)}>+</button>  
94              </div>  
95          )  
96      }  
97  }  
98  export default App;
```



# 3.4 - Componentes presentacionales (continuación)

Componente con estado (statefull component) :

```
1 import React, {Component} from 'react';
2 class App extends Component{
3     render(){
4         return (
5             <div className="App">
6                 <h1>Este es el nuevo saludo desde, React!</h1>
7                 <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>
8                 <TimestampToDate timestamp = {1585258546000}/>
9                 <MostrarFechaHora />
10                <GenerarSerie />
11                <Contador />
12            </div>
13        );
14    }
15 }
```

Prueba(statefull component) :

← → ⌂ ⓘ localhost:3001

**Este es el nuevo saludo desde, React!**

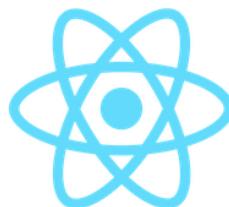
Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020

FECHA Y HORA ACTUAL:2020/03/27 9:35:37 pm

0,1,2,3,4,5,6,7,8,9,

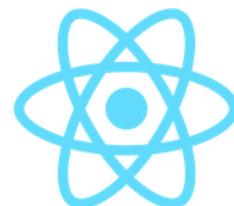
Cuenta actual: 37



## 3.5 - propTypes y defaultProps

Permiten definir valores predeterminados de propiedades y estados. Se debe tener en cuenta el siguiente procedimiento:

- La inicialización de las propiedades se realiza una vez para todos los elementos de la clase. Si tenemos un componente que se usa 5 veces, los valores de las propiedades predeterminadas se definirán una única vez.
- La inicialización del estado se realiza una vez para cada instancia del componente. Si tenemos un componente que se usa en 5 ocasiones, la inicialización de su estado se realizará 5 veces, una para cada instancia del componente.

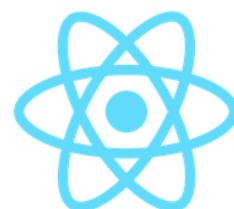


## 3.5 - propTypes y defaultProps (continuación)

**defaultProps:** En el caso de la inicialización de propiedades, en componentes creados con clases de ES6, se debe realizar definiendo una propiedad "defaultProps" en la clase que implementa el componente. En el caso de la inicialización del estado, la operativa se realiza en el constructor.

Ejemplo:

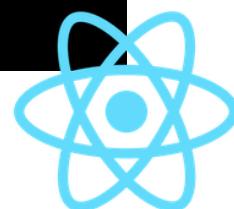
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App6.js. Posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js



# 3.5 - propsTypes y defaultProps (continuación)

Ejemplo defaultProps:

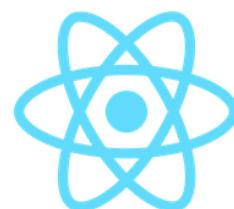
```
99  class CicloVida extends Component {  
100    constructor(...args) {  
101      super(...args)  
102      console.log('Ejecuto constructor', ...args)  
103      this.state = {  
104        estado: 'Inicializado en el constructor'  
105      }  
106    }  
107    render() {  
108      return (  
109        <div>  
110          <p>Componente con propiedades y estado inicializado</p>  
111          <p>Estado: {this.state.estado}</p>  
112          <p>Propiedad: {this.props.propiedad}</p>  
113        </div>  
114      )  
115    }  
116  }  
117 CicloVida.defaultProps = {  
118   propiedad: 'Valor por defecto definido para la propiedad'  
119 }  
120 export default App;
```



# 3.5 - propsTypes y defaultProps (continuación)

Ejemplo defaultProps:

```
1 import React, {Component} from 'react';
2 class App extends Component{
3     render(){
4         return (
5             <div className="App">
6                 <h1>Este es el nuevo saludo desde, React!</h1>
7                 <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>
8                 <TimestampToDate timestamp = {1585258546000}/>
9                 <MostrarFechaHora />
10                <GenerarSerie />
11                <Contador />
12                <CicloVida />
13            </div>
14        );
15    }
16 }
```



# 3.5 - propsTypes y defaultProps (continuación)

## Prueba defaultProps:

### Este es el nuevo saludo desde, React!

Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020  
FECHA Y HORA ACTUAL:2020/03/28 1:27:02 pm

0,1,2,3,4,5,6,7,8,9,

Cuenta actual: 0 

Componente con propiedades y estado inicializado

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad

### Este es el nuevo saludo desde, React!

Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020  
FECHA Y HORA ACTUAL:2020/03/28 1:27:02 pm

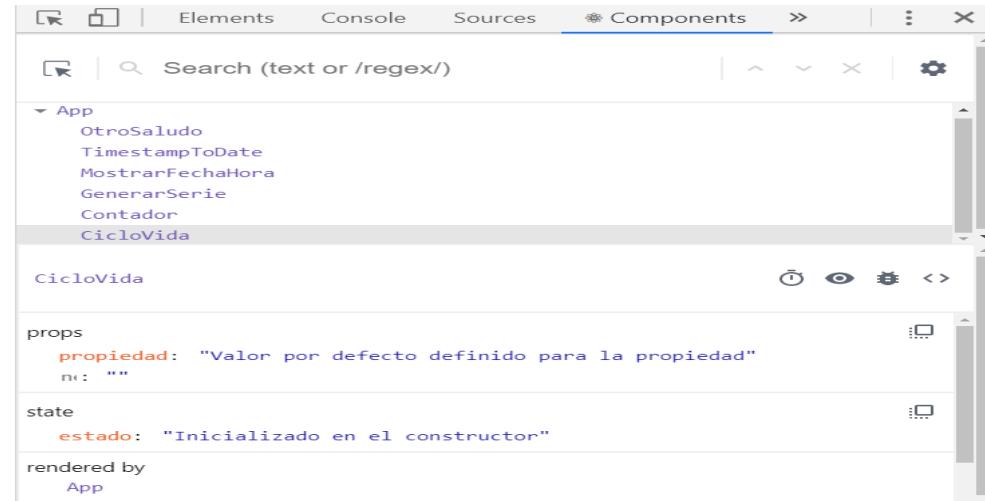
0,1,2,3,4,5,6,7,8,9,

Cuenta actual: 0 

Componente con propiedades y estado inicializado

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad



Components

Search (text or /regex/)

App

- OtroSaludo
- TimestampToDate
- MostrarFechaHora
- GenerarSerie
- Contador
- CicloVida

CicloVida

props

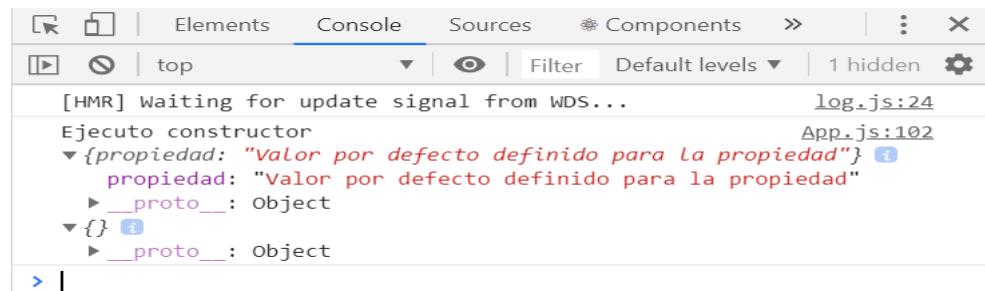
```
propiedad: "Valor por defecto definido para la propiedad"
n: ""
```

state

```
estado: "Inicializado en el constructor"
```

rendered by

App



Console

[HMR] Waiting for update signal from WDS... log.js:24

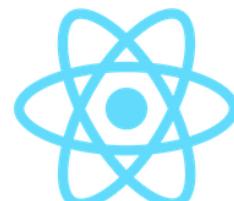
Ejecuto constructor

{propiedad: "Valor por defecto definido para la propiedad"} App.js:102

```
propiedad: "Valor por defecto definido para la propiedad"
__proto__: Object
```

{}

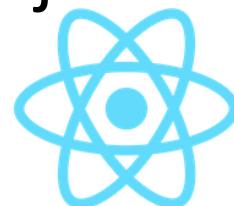
```
__proto__: Object
```



## 3.5 - propTypes y defaultProps (continuación)

**PropTypes:** React proporciona un mecanismo interno para agregar la verificación de tipos a los componentes. Los componentes de React, usan una propiedad especial llamada propTypes para configurar la verificación de tipos. Cuando las propiedades se pasan a un componente React , se comparan con las definiciones de tipo configuradas en la propiedad propTypes.

Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App7.js. Posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js



# 3.5 - propsTypes y defaultProps (continuación)

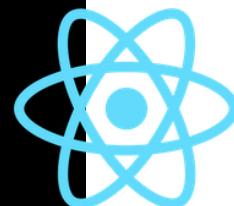
**Ejemplo PropsTypes:** Acceda al directorio del proyecto e instale el paquete “prop-types”:

```
C:\>cd react_ej01
C:\react_ej01>npm install prop-types
+ prop-types@15.7.2
removed 1596 packages, updated 1 package and audited 27 packages in 128.256s
found 0 vulnerabilities

C:\react_ej01>
```

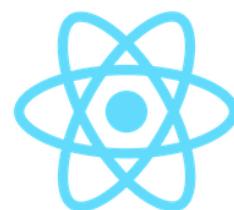
Efectúe las actualizaciones indicadas en los recuadros rojos, en el archivo src/App.js:

```
1 import React, {Component} from 'react';
2 import PropTypes from 'prop-types';
3 class App extends Component{
4     render(){
5         return (
6             <div className="App">
7                 <h1>Este es el nuevo saludo desde, React!</h1>
8                 <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>
9                 <TimestampToDate timestamp = {1585258546000}/>
10                <MostrarFechaHora />
11                <GenerarSerie />
12                <Contador />
13                <CicloVida />
14                <VerSalario />
15            </div>
16        );
17    }
18}
19 class OtroSaludo extends Component{
```



## 3.5 - propsTypes y defaultProps (continuación)

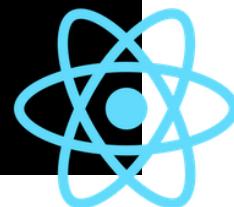
**Ejemplo PropsTypes:** Cuando los props se pasan a un componente React, se comparan con las definiciones de tipo configuradas en la propiedad propTypes. Cuando se pasa un valor no válido para un prop, se muestra una advertencia en la consola de JavaScript:



# 3.5 - propsTypes y defaultProps (continuación)

## Ejemplo PropsTypes:

```
122  /* PropTypes */
123  class Salario extends Component {
124      render() {
125          return (
126              <div>
127                  Salario Anual: US${this.props.salarioAnual}<br/>
128                  Salario Mensual: US${this.props.salarioAnual/12}<br/>
129              </div>
130          )
131      }
132      Salario.defaultProps = {
133          salarioAnual: 0
134      }
135      Salario.propTypes = {
136          salarioAnual: PropTypes.number
137      }
138  class VerSalario extends Component {
139      render() {
140          return <div><Salario salarioAnual={12600} /></div>
141          //return <div><Salario salarioAnual='12600' /></div>
142          //return <div><Salario salarioAnual='abcdef' /></div>
143      }
144  }
145  export default App;
```



# 3.5 - propsTypes y defaultProps

## (continuación)

### Prueba Ejemplo PropTypes:

#### Este es el nuevo saludo desde, React!

Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020

FECHA Y HORA ACTUAL:2020/03/29 7:49:28 pm

0,1,2,3,4,5,6,7,8,9,

Cuenta actual: 0 

Componente con propiedades y estado inicializado

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad

Salario Anual: US\$12600

Salario Mensual: US\$1050

#### Este es el nuevo saludo desde, React!

Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS

26/3/2020

FECHA Y HORA ACTUAL:2020/03/29 8:03:02 pm

0,1,2,3,4,5,6,7,8,9,

Cuenta actual: 0 

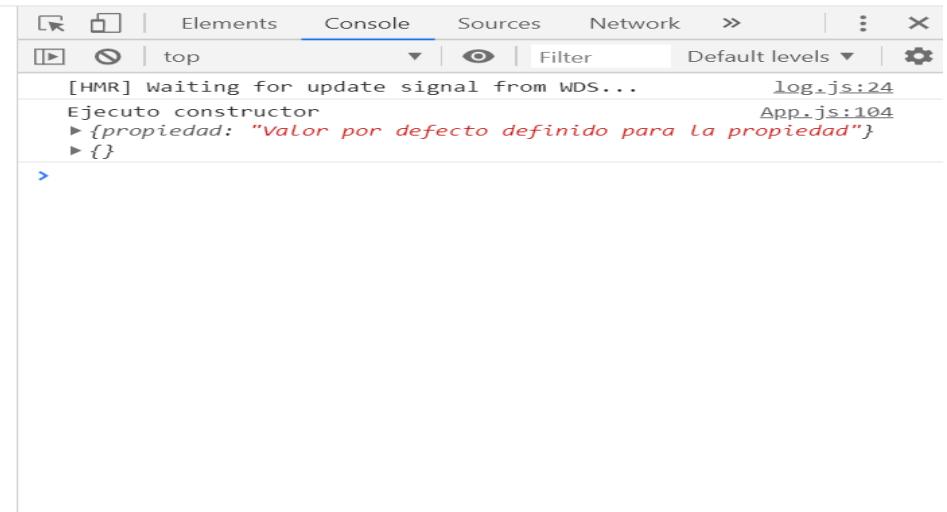
Componente con propiedades y estado inicializado

Estado: Inicializado en el constructor

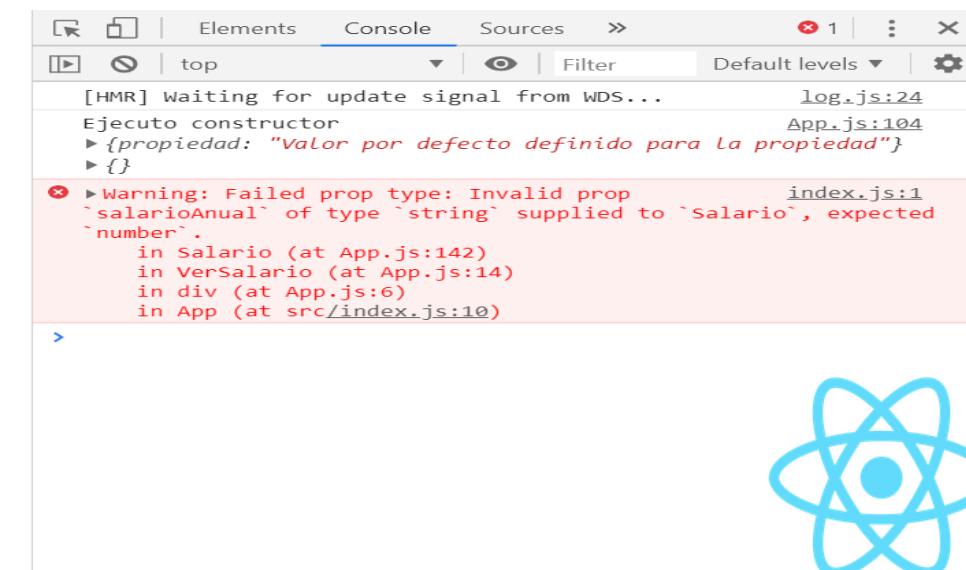
Propiedad: Valor por defecto definido para la propiedad

Salario Anual: US\$abcdef

Salario Mensual: US\$NaN

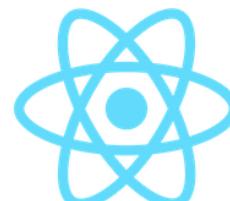


```
[HMR] Waiting for update signal from WDS... log.js:24
Ejecuto constructor
▶ {propiedad: "Valor por defecto definido para la propiedad"}
▶ {}
```



```
[HMR] Waiting for update signal from WDS... log.js:24
Ejecuto constructor
▶ {propiedad: "Valor por defecto definido para la propiedad"}
▶ {}

✖ Warning: Failed prop type: Invalid prop `salarioAnual` of type `string` supplied to `Salario`, expected `number`.
  in Salario (at App.js:142)
  in Versalario (at App.js:14)
  in div (at App.js:6)
  in App (at src/index.js:10)
```



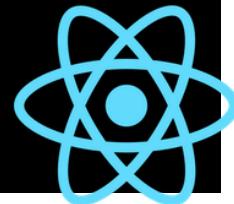
# 3.5 - propsTypes y defaultProps

## (continuación)

### Otro Ejemplo PropTypes:

```
146 class Saludo extends Component {  
147   render() {  
148     return (  
149       <h1>Hola, {this.props.nombre}</h1>  
150     );  
151   }  
152 }  
153  
154 Saludo.propTypes = {  
155   nombre: PropTypes.string  
156 };  
157  
158 export default App;
```

```
1 import React, {Component} from 'react';  
2 import PropTypes from 'prop-types';  
3 class App extends Component{  
4   render(){  
5     return (  
6       <div className="App">  
7         <h1>Este es el nuevo saludo desde, React!</h1>  
8         <OtroSaludo nombre = "HENRY DUQUE" app = "MI CURSO DE REACT JS"/>  
9         <TimestampToDate timestamp = {1585258546000}/>  
10        <MostrarFechaHora />  
11        <GenerarSerie />  
12        <Contador />  
13        <CicloVida />  
14        <VerSalario />  
15        <Saludo nombre = 'HENRY DUQUE' />  
16      </div>  
17    );  
18  }  
19}
```



# 3.5 - propsTypes y defaultProps (continuación)

## Prueba Otro Ejemplo PropTypes:

```
localhost:3001
Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS
26/3/2020
FECHA Y HORA ACTUAL:2020/03/29 10:31:58 pm
0,1,2,3,4,5,6,7,8,9,
Cuenta actual: 0 + 
Componente con propiedades y estado inicializado
Estado: Inicializado en el constructor
Propiedad: Valor por defecto definido para la propiedad
Salario Anual: US$12600
Salario Mensual: US$1050
Hola, HENRY DUQUE
```

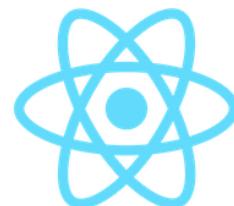
```
localhost:3001
[HMR] Waiting for update signal from WDS...
Ejecuto constructor
▶ {propiedad: "Valor por defecto definido para la propiedad"}
▶ {}
>
```

## Asignando un número a la propiedad 'nombre'

```
<Saludo nombre = {1234} />
```

```
localhost:3001
Hola, este es un saludo desde otro componente:HENRY DUQUE y MI CURSO DE REACT JS
26/3/2020
FECHA Y HORA ACTUAL:2020/03/29 10:35:10 pm
0,1,2,3,4,5,6,7,8,9,
Cuenta actual: 0 + 
Componente con propiedades y estado inicializado
Estado: Inicializado en el constructor
Propiedad: Valor por defecto definido para la propiedad
Salario Anual: US$12600
Salario Mensual: US$1050
Hola, 1234
```

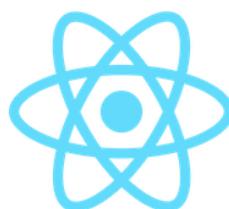
```
localhost:3001
[HMR] Waiting for update signal from WDS...
✖ Warning: Failed prop type: Invalid prop `nombre` supplied to `Saludo`, expected `string`.
  in Saludo (at App.js:15)
  in App (at src/index.js:10)
Ejecuto constructor
▶ {propiedad: "Valor por defecto definido para la propiedad"}
▶ {}
>
```



## 3.6 - Recorriendo arrays

Un arreglo (vector) es una estructura de datos dinámica que permite guardar temporalmente un conjunto de datos, los cuales se pueden acceder a través de un índice. En React, usamos el método map(), para presentar arrays.

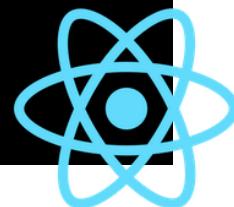
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App8.js. Posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js



# 3.6 - Recorriendo arrays

Ejemplo:

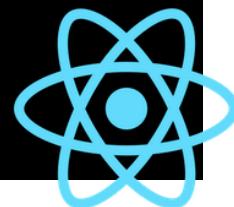
```
1 import React, {Component} from 'react';
2 import PropTypes from 'prop-types';
3 class App extends Component{
4   render(){
5     return (
6       <div className="App">
7         <h1>ESTUDIO DE ARRAY DESDE REACT JS</h1>
8         <Arreglo1 />
9         <Arreglo2 />
10        <Arreglo3 />
11        <Arreglo4 />
12        <Arreglo5 />
13        <Arreglo6 />
14        <Arreglo7 />
15        <Arreglo8 />
16        <Arreglo9 />
17        <Arreglo10 />
18        <Arreglo11 />
19        <Arreglo12 />
20       </div>
21     );
22   }
23 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

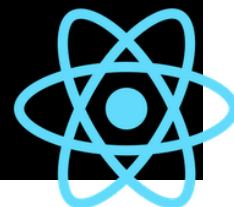
```
25 class Arreglo1 extends Component{  
26     constructor(props){  
27         super(props);  
28         this.state = {  
29             list:[1,2,3],  
30         };  
31     }  
32     render(){  
33         return(  
34             <div>  
35                 <label style={{color:'red', background:'black'}}>Arreglo1</label>  
36                 <ul>  
37                     {this.state.list.map(item =>(  
38                         <li key={item}>{item}</li>  
39                     ))}  
40                 </ul>  
41             </div>  
42         );  
43     }  
44 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

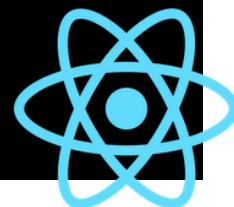
```
46 class Arreglo2 extends Component {  
47     constructor(props) {  
48         super(props);  
49         this.state = {  
50             list: [],  
51         };  
52     }  
53     render() {  
54         return (  
55             <div>  
56                 <label style={{color:'green', background:'black'}}>Arreglo2</label>  
57                     <ul>  
58                         {this.state.list.map(item => (  
59                             <li key={item}>{item}</li>  
60                         ))}  
61                     </ul>  
62             </div>  
63         );  
64     }  
65 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

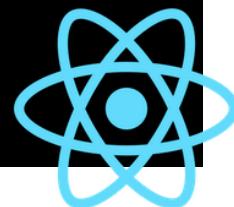
```
67 class Arreglo3 extends Component {  
68     constructor(props) {  
69         super(props);  
70         this.state = {  
71             list: null,  
72         };  
73     }  
74     render() {  
75         return (  
76             <div>  
77                 <label style={{color:'blue', background:'black'}}>Arreglo3</label>  
78                 <ul>  
79                     {((this.state.list || [])).map(item => (  
80                         <li key={item}>{item}</li>  
81                     ))}  
82                 </ul>  
83             </div>  
84         );  
85     }  
86 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

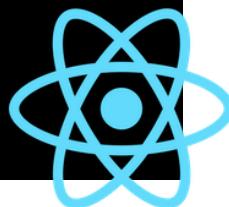
```
88 class Arreglo4 extends Component {  
89     constructor(props) {  
90         super(props);  
91         this.state = {  
92             list: [1, 2, 3],  
93         };  
94     }  
95     onClearArray = () => {  
96         this.setState({ list: [] });  
97     };  
98     render() {  
99         return (  
100             <div>  
101                 <label style={{color:'yellow', background:'black'}}>Arreglo4</label>  
102                     <ul>  
103                         {this.state.list.map(item => (  
104                             <li key={item}>{item}</li>  
105                         ))}  
106                     </ul>  
107                     <button type="button" onClick={this.onClearArray}>  
108                         Limpiar arreglo  
109                     </button>  
110                 </div>  
111             );  
112     }  
113 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

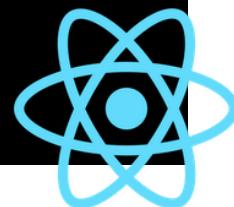
```
116 const list = [1, 2, 3];
117 class Arreglo5 extends Component {
118     constructor(props) {
119         super(props);
120         this.state = {
121             list,
122         };
123     }
124     onClearArray = () => {
125         this.setState({ list: [] });
126     };
127     onResetArray = () => {
128         this.setState({ list });
129     };
130     render() {
131         return (
132             <div>
133                 <label style={{color:'blue', background:'black'}>Arreglo5</label>
134                 <ul>
135                     {this.state.list.map(item => (
136                         <li key={item}>{item}</li>
137                     ))}
138                 </ul>
139                 <button type="button" onClick={this.onClearArray}>
140                     Limpiar arreglo (vector)
141                 </button>
142                 <button type="button" onClick={this.onResetArray}>
143                     Re iniciar arreglo (vector)
144                 </button>
145             </div>
146         );
147     }
148 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

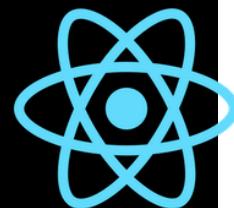
```
150 const INITIAL_STATE = {  
151   list: [1, 2, 3],  
152 };  
153 class Arreglo6 extends Component {  
154   constructor(props) {  
155     super(props);  
156     this.state = INITIAL_STATE;  
157   }  
158   onClearArray = () => {  
159     this.setState({ list: [] });  
160   };  
161   onResetArray = () => {  
162     this.setState({ ...INITIAL_STATE });  
163   };  
164   render() {  
165     return (  
166       <div>  
167         <label style={{color:'red', background:'black'}}>Arreglo6</label>  
168         <ul>  
169           {this.state.list.map(item => (  
170             <li key={item}>{item}</li>  
171           ))}  
172         </ul>  
173         <button type="button" onClick={this.onClearArray}>  
174           Limpiar arreglo (vector)  
175         </button>  
176         <button type="button" onClick={this.onResetArray}>  
177           Re iniciar arreglo (vector)  
178         </button>  
179       </div>  
180     );  
181   }  
182 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

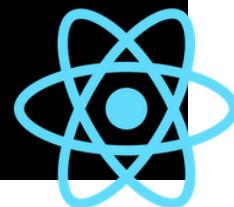
```
184 class Arreglo7 extends Component {
185   constructor(props) {
186     super(props);
187     this.state = {
188       value: '',
189       list: ['a', 'b', 'c'],
190     };
191   }
192   onChangeValue = event => {
193     this.setState({ value: event.target.value });
194   };
195   onAddItem = () => {
196     // NO PERMITIDO, NO FUNCIONARÁ
197     this.setState(state => {
198       const list = state.list.push(state.value);
199       return {
200         list,
201         value: '',
202       };
203     });
204   };
205   render() {
206     return (
207       <div>
208         <label style={{color:'purple', background:'black'}}>Arreglo7</label>
209         <ul>
210           {this.state.list.map(item => (
211             <li key={item}>{item}</li>
212           ))}
213         </ul>
214         <input
215           type="text"
216           value={this.state.value}
217           onChange={this.onChangeValue}
218         />
219         <button
220           type="button"
221           onClick={this.onAddItem}
222           disabled={!this.state.value}
223         >
224           Añadir
225         </button>
226       </div>
227     );
228   }
229 }
230 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

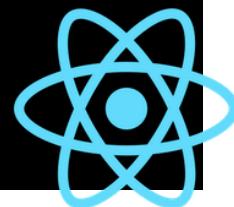
```
231  class Arreglos extends Component {
232      constructor(props) {
233          super(props);
234          this.state = {
235              list: [42, 33, 68],
236          };
237      }
238      onUpdateItems = () => {
239          this.setState(state => {
240              const list = state.list.map(item => item + 1);
241              return {
242                  list,
243              };
244          });
245      };
246      render() {
247          return (
248              <div>
249                  <label style={{color:'cyan', background:'black'}}>Arreglos</label>
250                  <ul>
251                      {this.state.list.map(item => (
252                          <li key={item}>La persona tiene {item} años.</li>
253                      )));
254                  </ul>
255                  <button type="button" onClick={this.onUpdateItems}>
256                      Incrementar a todos un año en su edad
257                  </button>
258              </div>
259          );
260      }
261  }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

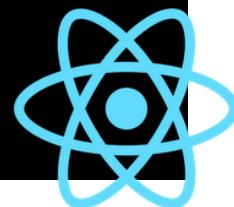
```
263 class Arreglo9 extends Component {
264     constructor(props) {
265         super(props);
266         this.state = {
267             list: [42, 33, 68],
268         };
269     }
270     onUpdateItem = i => {
271         this.setState(state => {
272             const list = state.list.map((item, j) => {
273                 if (j === i) {
274                     return item + 1;
275                 } else {
276                     return item;
277                 }
278             });
279             return {
280                 list,
281             };
282         });
283     };
284     render() {
285         return (
286             <div>
287                 <label style={{color:'violet', background:'black'}}>Arreglo9</label>
288                 <ul>
289                     {this.state.list.map((item, index) => (
290                         <li key={item}>
291                             La persona tiene {item} años.
292                             <button
293                                 type="button"
294                                 onClick={() => this.onUpdateItem(index)}
295                             >
296                             Incrementar un año de edad
297                             </button>
298                         </li>
299                     ))}
300                 </ul>
301             </div>
302         );
303     }
304 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

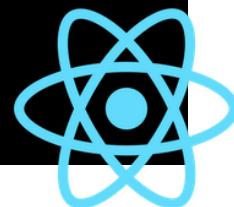
```
306  class Arreglo10 extends Component {
307    constructor(props) {
308      super(props);
309      this.state = {
310        list: [42, 33, 68],
311      };
312    }
313    onRemoveItem = i => {
314      this.setState(state => {
315        const list = state.list.filter((item, j) => i !== j);
316        return {
317          list,
318        };
319      });
320    };
321    render() {
322      return (
323        <div>
324          <label style={{color:'gray', background:'black'}}>Arreglo10</label>
325          <ul>
326            {this.state.list.map((item, index) => (
327              <li key={item}>
328                La persona tiene {item} años.
329                <button
330                  type="button"
331                  onClick={() => this.onRemoveItem(index)}
332                >
333                  Borrar
334                </button>
335              </li>
336            ))}
337          </ul>
338        </div>
339      );
340    }
341  }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

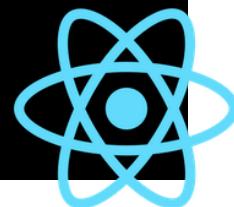
```
343 class Arreglo11 extends Component {
344     constructor(props) {
345         super(props);
346         this.state = {
347             list: [42, 33, 68],
348         };
349     }
350     onRemoveFirstItem = () => {
351         this.setState(state => {
352             const [first, ...rest] = state.list;
353             return {
354                 list: rest,
355             };
356         });
357     };
358     render() {
359         return (
360             <div>
361                 <label style={{color:'orange', background:'black'}}>Arreglo11</label>
362                 <ul>
363                     {this.state.list.map(item => (
364                         <li key={item}>La persona tiene {item} años.</li>
365                     ))}
366                     <button type="button" onClick={this.onRemoveFirstItem}>
367                         Borrar el primer elemento
368                     </button>
369                 </ul>
370             </div>
371         );
372     }
373 }
374 }
```



# 3.6 - Recorriendo arrays

Ejemplo continuación:

```
375 class Arreglo12 extends Component {
376   constructor(props) {
377     super(props);
378     this.state = {
379       list: [
380         { id: '1', edad: 42 },
381         { id: '2', edad: 33 },
382         { id: '3', edad: 68 },
383       ],
384     };
385   }
386   onRemoveItem = id => {
387     this.setState(state => {
388       const list = state.list.filter(item => item.id !== id);
389       return {
390         list,
391       };
392     });
393   };
394   render() {
395     return (
396       <div>
397         <label style={{color:'white', background:'black'}>Arreglo12</label>
398         <ul>
399           {this.state.list.map(item => (
400             <li key={item.id}>
401               La persona tiene {item.edad} años.
402               <button
403                 type="button"
404                 onClick={() => this.onRemoveItem(item.id)}
405               >
406                 Borrar
407               </button>
408             </li>
409           ))}
410         </ul>
411       </div>
412     );
413   }
414 }
415
416
417 export default App;
```



# 3.6 - Recorriendo arrays

Prueba Ejemplo:

← → C ⓘ localhost:3001



## ESTUDIO DE ARRAY DESDE REACT JS

Arreglo1

- 1
- 2
- 3

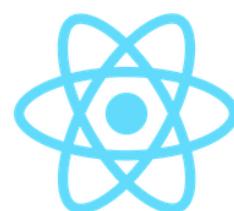
Arreglo2

Arreglo3

Arreglo4

- 1
- 2
- 3

Limpiar arreglo



# El estado de los componentes

4.1 - Creando un componente de clase

4.2 - Formulario

4.3 - Estado del componente

4.4 - Actualizando el estado

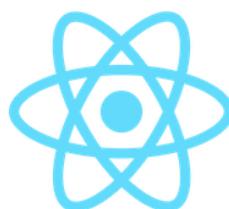
4.5 - Escribir métodos

4.6 - Ciclo de vida de montaje

4.7 - Ciclo de vida de actualización

4.8 - Ciclo de vida de desmontaje

4.9 - Consejos finales



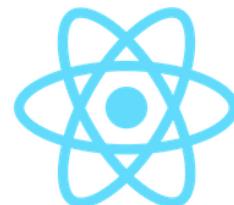
## 4.1 - Creando un componente de clase

Se habla de componentes funcionales y de clase válidos, cuando estos aceptan argumentos de objeto “props” (que proviene de propiedades) con datos y devuelven un elemento de React.

Se llama a dichos componentes “funcionales” porque literalmente son funciones JavaScript.

Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App9.js.

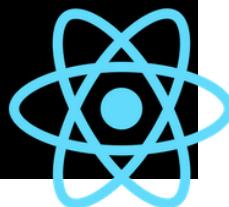
Posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js



# 4.1 - Creando un componente de clase

Ejemplo:

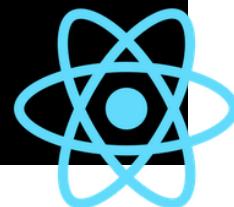
```
1 import React, {Component} from 'react';
2 import Table from './Table'
3
4 class App extends Component{
5   render(){
6     return (
7       <div className="container">
8         <Table/>
9       </div>
10    )
11  }
12}
13
14 export default App;
```



# 4.1 - Creando un componente de clase

Ejemplo Table.js:

```
1 import React, {Component} from 'react';
2
3 class Table extends Component{
4   render(){
5     return (
6       <table>
7         <thead>
8           <tr>
9             <th>Nombre</th>
10            <th>Trabajo</th>
11          </tr>
12        </thead>
13        <tbody>
14          <tr>
15            <td>JOSÉ PEREZ</td>
16            <td>ANALISTA</td>
17          </tr>
18          <tr>
19            <td>PEDRO RUIZ</td>
20            <td>SUPERVISOR</td>
21          </tr>
22          <tr>
23            <td>MARÍA SUAREZ</td>
24            <td>ASISTENTE</td>
25          </tr>
26        </tbody>
27      </table>
28    )
29  }
30}
31
32 export default Table;
```



# 4.1 - Creando un componente de clase

Prueba Ejemplo:

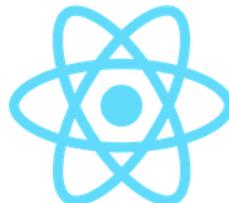
The screenshot shows a browser window at localhost:3001 displaying a simple table with three rows. The columns are labeled 'Nombre' and 'Trabajo'. The data is as follows:

Nombre	Trabajo
JOSÉ PEREZ	ANALISTA
PEDRO RUÍZ	SUPERVISOR
MARÍA SUAREZ	ASISTENTE

To the right of the browser is a component inspector tool. The 'Components' tab is selected, showing a tree structure under the 'App' component. The 'Table' component is highlighted with a blue bar. Below the tree, detailed information about the 'Table' component is shown:

- Table**: The component name.
- props**: A prop named 'new prop' with a value of '""'.
- rendered by**: The 'App' component.
- source**: App.js:8

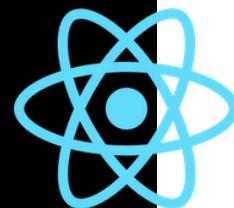
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App10.js y src/Table.js, como src/Table1.js, posteriormente realice las siguientes actualizaciones sobre los archivo src/App.js y src/Table.js



# 4.1 - Creando un componente de clase

Continuación Ejemplo src/App.js:

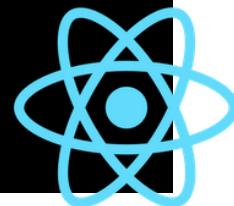
```
1 import React, {Component} from 'react';
2 import Table from './Table'
3
4 class App extends Component{
5   render(){
6     const datos = [
7       {
8         nombre: 'Pedro Martínez',
9         trabajo: 'Analista de Sistemas',
10      },
11      {
12        nombre: 'Julio Pereira',
13        trabajo: 'Planificador',
14      },
15      {
16        nombre: 'Vanessa Parra',
17        trabajo: 'Gerente',
18      },
19      {
20        nombre: 'Victor Luengo',
21        trabajo: 'Coodinador',
22      },
23    ]
24
25    return (
26      <div className="container">
27        <Table tablaDatos = {datos}/>
28      </div>
29    )
30  }
31}
32
33 export default App;
```



# 4.1 - Creando un componente de clase

Continuación Ejemplo src/Table.js:

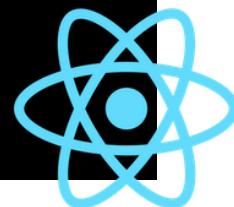
```
1 import React, {Component} from 'react';
2 
3 /* 
4     Componentes simples
5 */
6 
7 const TableHeader = () =>{
8     return(
9         <thead>
10            <tr>
11                <th>Nombre</th>
12                <th>Trabajo</th>
13            </tr>
14        </thead>
15    )
16 }
17 
18 const TableBody = () =>{
19     return(
20         <tbody>
21            <tr>
22                <td>JOSE PEREZ</td>
23                <td>ANALISTA</td>
24            </tr>
25            <tr>
26                <td>PEDRO RUÍZ</td>
27                <td>SUPERVISOR</td>
28            </tr>
29            <tr>
30                <td>MARÍA SUAREZ</td>
31                <td>ASISTENTE</td>
32            </tr>
33            <tr>
```



# 4.1 - Creando un componente de clase

Continuación Ejemplo src/Table.js:

```
34           <td>VANESSA TORRES</td>
35           <td>VENDEDOR</td>
36       </tr>
37   </tbody>
38 )
39 }
40
41 /**
42     Componente de clase
43 */
44
45
46 class Table extends Component{
47     render(){
48         return (
49             <table>
50                 <TableHeader />
51                 <TableBody />
52             </table>
53         )
54     }
55 }
56
57 export default Table;
```



# 4.1 - Creando un componente de clase

Prueba Ejemplo:

Nombre	Trabajo
JOSÉ PEREZ	ANALISTA
PEDRO RUÍZ	SUPERVISOR
MARÍA SUAREZ	ASISTENTE
VANESSA TORRES	VENDEDOR

The screenshot shows a developer tool interface for a React application. On the left, there is a component tree with the following structure:

- App
  - Table
    - TableHeader
    - TableBody

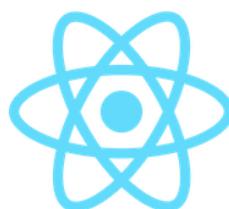
Below the component tree, the word "Table" is highlighted in blue. To the right of the component tree is a "props" panel displaying the following data:

```
tablaDatos: [{}]
  0: {nombre: "Pedro Martínez", trabajo: "A..."}
  1: {nombre: "Julio Pereira", trabajo: "Pl..."}
  2: {nombre: "Vanessa Parra", trabajo: "Ge..."}
  3: {nombre: "Victor Luengo", trabajo: "Co..."}  
new prop : ""
```

At the bottom right of the interface, there is a blue React logo icon.

## 4.3 - Estado del componente

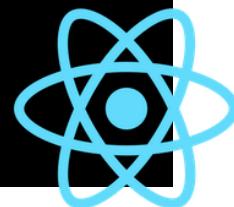
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App11.js y src/Table.js, como src/Table2.js, posteriormente realice las siguientes actualizaciones sobre los archivo src/App.js y src/Table.js



# 4.3 - Estado del componente

Ejemplo src/App.js:

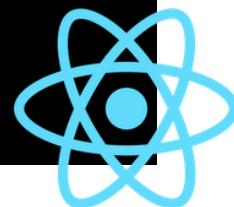
```
1 import React, {Component} from 'react';
2 import Table from './Table';
3
4 /**
5  * Atención: Palabras reservadas en este aplicación
6  * state
7  * setState
8 */
9
10 class App extends Component{
11     state = {
12         datos : [
13             {
14                 nombre: 'Pedro Martínez',
15                 trabajo: 'Analista de Sistemas',
16             },
17             {
18                 nombre: 'Julio Pereira',
19                 trabajo: 'Planificador',
20             },
21             {
22                 nombre: 'Vanessa Parra',
23                 trabajo: 'Gerente',
24             },
25             {
26                 nombre: 'Victor Luengo',
27                 trabajo: 'Coodinador',
28             },
29         ]
30     };
31
32     borrarDatos = index => {
33         const { datos } = this.state
```



# 4.3 - Estado del componente

Continuación Ejemplo src/App.js:

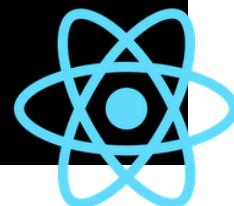
```
34     this.setState({
35         datos: datos.filter((datos, i) => {
36             return i !== index
37
38         })
39     });
40 }
41
42 render(){
43     const { datos } = this.state
44     return (
45         <div className = "container">
46             <Table
47                 tablaDatos={datos}
48                 borrarDatos={this.borrarDatos}
49             />
50         </div>
51     )
52 }
53
54
55 export default App;
56 }
```



# 4.3 - Estado del componente

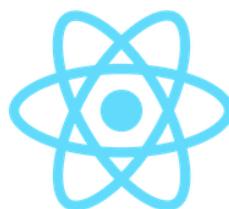
Ejemplo src/Table.js. Efectuar la siguiente actualización:

```
42  /*
43   * Componente de clase
44   * Los datos estarán contenidos en el virtual DOM, pero sin ser mostrados
45   * a través de la página
46  */
47
48 class Table extends Component{
49   render(){
50     const {tablaDatos} = this.props
51     return (
52       <table>
53         <TableHeader />
54         <TableBody tablaDatos={tablaDatos}/>
55       </table>
56     )
57   }
58 }
59
60 export default Table;
```



## 4.4 - Actualizando el estado

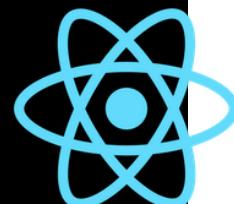
Siguiendo el procedimiento conocido, guarde el archivo `src/App.js`, como `src/App12.js` y `src/Table.js`, como `src/Table3.js`, posteriormente realice la siguiente actualización sobre el archivo `src/Table.js`.



## 4.4 - Actualizando el estado

Ejemplo src/Table.js. Efectué la siguiente actualización (no modificar src/App.js):

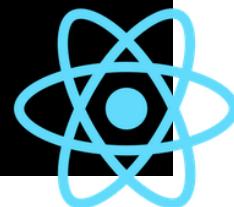
```
1 import React, {Component} from 'react';
2
3 // Accesorios
4 /*
5   Componentes simples
6 */
7
8 const TableHeader = () =>{
9   return(
10     <thead>
11       <tr>
12         <th>Nombre</th>
13         <th>Trabajo</th>
14       </tr>
15     </thead>
16   )
17 }
18
19 /*
20   Componente de clase
21   Los datos estarán contenidos en el virtual DOM, pero sin ser mostrados
22   a través de la página
23 */
24
25 const TableBody = props => {
26   const rows = props.tablaDatos.map((row,index) => {
27     return (
28       <tr key={index}>
29         <td>{row.nombre}</td>
30         <td>{row.trabajo}</td>
31       </tr>
32     )
33   })
34 }
```



## 4.4 - Actualizando el estado

Continuación Ejemplo src/Table.js. Efectué las siguientes actualizaciones:

```
34     return <tbody>{rows}</tbody>
35   }
36
37   class Table extends Component{
38     render(){
39       const {tablaDatos} = this.props
40       return (
41         <table>
42           <TableHeader />
43           <TableBody tablaDatos={tablaDatos}/>
44         </table>
45       )
46     }
47   }
48
49   export default Table;
```



# 4.4 - Actualizando el estado

Prueba: Resultado de la actualización.

localhost:3001

Nombre	Trabajo
Pedro Martínez	Analista de Sistemas
Julio Pereira	Planificador
Vanessa Parra	Gerente
Victor Luengo	Coordinador

Elements Console Components

Search (text or /regex/)

App

Table

TableHeader

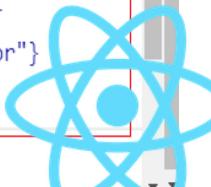
TableBody

TableBody

props

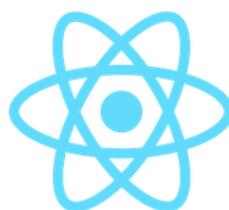
```
tablaDatos: [...], ...]
  0: {nombre: "Pedro Martínez", trabajo: "Analista de Si..."}
  1: {nombre: "Julio Pereira", trabajo: "Planificador"}
  2: {nombre: "Vanessa Parra", trabajo: "Gerente"}
  3: {nombre: "Victor Luengo", trabajo: "Coordinador"}
new prop : ""
```

rendered by



## 4.5 - Escribir métodos

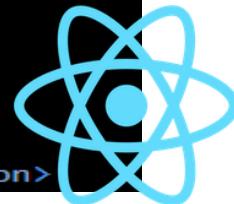
Siguiendo el procedimiento conocido, guarde el archivo `src/Table.js`, como `src/Table4.js`, posteriormente realice las siguientes actualizaciones sobre el archivo `src/Table.js`.



# 4.5 - Escribir métodos

Inicialmente se modificará el archivo src/Table.js:

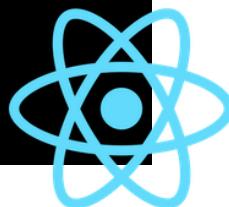
```
1 import React from 'react';
2
3 // Accesorios
4 /*
5   Componentes simples
6 */
7
8 const TableHeader = () =>{
9   return(
10     <thead>
11       <tr>
12         <th>Nombre</th>
13         <th>Trabajo</th>
14         <th>Acción</th>
15       </tr>
16     </thead>
17   )
18 }
19
20 /*
21   Componente de clase
22   Los datos estarán contenidos en el virtual DOM, pero sin ser mostrados
23   a través de la página
24 */
25
26 const TableBody = props => {
27   const rows = props.tablaDatos.map((row,index) => {
28     return (
29       <tr key={index}>
30         <td>{row.nombre}</td>
31         <td>{row.trabajo}</td>
32         <td>
33           <button onClick = {() => props.borrarDatos(index)}>Borrar</button>
```



# 4.5 - Escribir métodos

Continuación actualización archivo src/Table.js:

```
34         </td>
35     </tr>
36   )
37 }
38 return <tbody>{rows}</tbody>
39 }
40
41 const Table = props => {
42   const { tablaDatos, borrarDatos } = props
43
44   return (
45     <table>
46       <TableHeader />
47       <TableBody tablaDatos={tablaDatos} borrarDatos={borrarDatos}/>
48     </table>
49   )
50 }
51
52
53 export default Table;
```



# 4.5 - Escribir métodos

## Resultado de la Actualización:

localhost:3001

Nombre	Trabajo	Acción
Pedro Martínez	Analista de Sistemas	<b>Borrar</b>
Julio Pereira	Planificador	<b>Borrar</b>
Vanessa Parra	Gerente	<b>Borrar</b>
Victor Luengo	Coordinador	<b>Borrar</b>

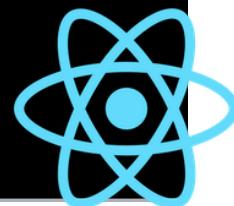
The screenshot shows a browser window displaying a table of employee data. The table has three columns: Nombre, Trabajo, and Acción. Each row contains a 'Borrar' button in the Acción column. To the right of the table is the React DevTools component inspector. The 'Components' tab is selected, showing the component tree: App > Table > TableBody. The 'TableBody' node is highlighted. Below the component tree, the 'props' section is expanded, showing the state variable 'tablaDatos' which contains an array of four objects, each representing an employee with 'nombre' and 'trabajo' properties. A blue atom icon is visible in the bottom right corner of the inspector.

```
borrarData: f () {}
▼ tablaDatos: [...], [...], [...], [...]
  ▶ 0: {nombre: "Pedro Martínez", trabajo: "Analista de Si..."}
  ▶ 1: {nombre: "Julio Pereira", trabajo: "Planificador"}
  ▶ 2: {nombre: "Vanessa Parra", trabajo: "Gerente"}
  ▶ 3: {nombre: "Victor Luengo", trabajo: "Coordinador"}
new prop : ""
```

## 4.2 - Formulario

Ahora se procederá a crear el formulario, archivo src/Form.js:

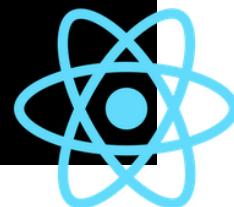
```
1 import React, { Component } from 'react';
2
3 class Form extends Component{
4     constructor(props){
5         super(props);
6
7         this.initialState = {
8             nombre : '',
9             trabajo : ''
10    };
11
12    this.state = this.initialState;
13 }
14
15 manejarCambio = event => {
16     const { name, value } = event.target;
17
18     this.setState({
19         [name] : value,
20     })
21 }
22
23 onFormSubmit = (event) => {
24     event.preventDefault();
25
26     this.props.manejarEnvio(this.state)
27     this.setState(this.initialState)
28 }
29
30 render(){
31     const { nombre, trabajo} = this.state;
32
33     return(
```



# 4.2 - Formulario

Continuación src/Form.js:

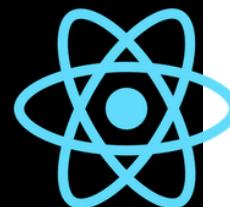
```
34      <form onSubmit={this.onFormSubmit}>
35        <label for="nombre">Nombre:</label>
36        <input
37          type="text"
38          name="nombre"
39          id="nombre"
40          value={nombre}
41          onChange={this.manejarCambio} />
42        <label for="trabajo">Trabajo:</label>
43        <input
44          type="text"
45          name="trabajo"
46          id="trabajo"
47          value={trabajo}
48          onChange={this.manejarCambio} />
49        <button type="submit">
50          Enviar
51        </button>
52      </form>
53    );
54  }
55
56 export default Form;
```



# 4.2 - Formulario

Actualización del archivo src/App.js:

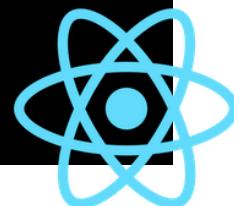
```
1 import React, {Component} from 'react';
2 import Table from './Table';
3 import Form from './Form';
4
5 /**
6  * Atención: Palabras reservadas en este aplicación
7  * state
8  * setState
9 */
10
11 /**
12  * Se creará un formulario
13 */
14
15 class App extends Component{
16     state = {
17         datos : []
18     };
19
20     borrarDatos = index => {
21         const { datos } = this.state
22
23         this.setState({
24             datos: datos.filter((datos, i) => {
25                 return i !== index
26
27             })
28         );
29     }
30
31     manejarEnvio = dato =>{
32         this.setState({datos:[...this.state.datos, dato]});
33     }
}
```



# 4.2 - Formulario

Continuación archivo src/App.js:

```
34
35     render(){
36         const { datos } = this.state;
37
38         return (
39             <div className = "container">
40                 <h1>Curso de React</h1>
41                 <p>Añadir nombre y actividad del trabajador</p>
42                 <Table
43                     tablaDatos={datos}
44                     borrarDato={this.borrarDato}
45                 />
46                 <h3>Ingresar un nuevo registro</h3>
47                 <Form manejarEnvio={this.manejarEnvio} />
48             </div>
49         );
50     }
51 }
52
53 export default App;
54 }
```



# 4.2 - Formulario

Prueba:

← → ⌛ ⓘ localhost:3001 🔍 ⭐ 🗑️ 🖼 🕵️ 🕵️ 🕵️ 🕵️ :

## Curso de React

Añadir nombre y actividad del trabajador

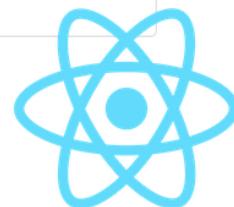
Nombre	Trabajo	Acción
HENRY DUQUE	PROFESOR	<button>Borrar</button>

## Ingresar un nuevo registro

Nombre:

Trabajo:

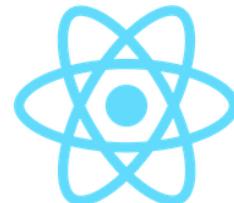
Enviar



## 4.6 – Ciclo de vida

El ciclo de vida no es más que una serie de estados por los cuales pasa todo componente a lo largo de su existencia. Esos estados tienen correspondencia en diversos métodos, que nosotros podemos implementar para realizar acciones cuando se van produciendo.

En React es fundamental el ciclo de vida, porque hay determinadas acciones que debemos necesariamente realizar en el momento correcto de ese ciclo. Ese es el motivo por el que hay que aprenderse muy bien cuáles son las distintas etapas por las que pasa la ejecución de un componente React



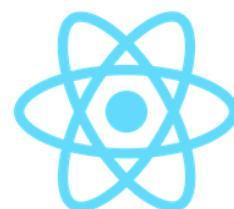
## 4.6 – Ciclo de vida

Antes de comenzar cabe decir que esto es algo específico de los componentes con estado, ya que los componentes sin estado tienen apenas un método que se usará para renderizar el componente y React no controlará su ciclo de vida a través de los métodos que veremos a continuación:

4.6 - Ciclo de vida : montaje.

4.7 - Ciclo de vida : actualización.

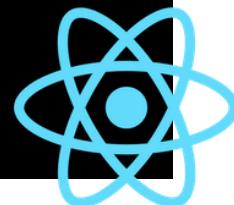
4.8 - Ciclo de vida : desmontaje.



## 4.6 – Ciclo de vida

Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App13.js, posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js:

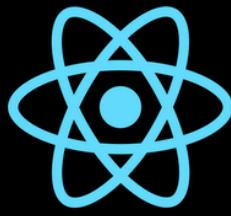
```
1 import React, {Component} from 'react';
2
3 class App extends Component{
4     render(){
5         return (
6             <div className="App">
7                 <h1>CICLO DE VIDA</h1>
8                 <CicloVida />
9             </div>
10        );
11    }
12 }
```



# 4.6 - Ciclo de vida : montaje

- El montaje se produce la primera vez que un componente va a generarse, incluyéndose en el DOM.

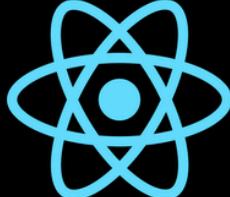
```
13
14 class CicloVida extends Component {
15   constructor(...args) {
16     console.log('Ejecuto constructor', ...args)
17     super(...args)
18     this.state = {
19       estado: 'Inicializado en el constructor'
20     }
21   }
22
23 //componentWillMount() {
24 UNSAFE_componentWillMount(){
25   console.log('Se ejecuta UNSAFE_componentWillMount')
26 }
27 //}
28
29 componentDidMount() {
30   console.log('Se ejecuta componentDidMount')
31 }
32
33 //componentWillReceiveProps(nextProps) {
34 UNSAFE_componentWillReceiveProps(nextProps) {
35   console.log('Se ejecuta UNSAFE_componentWillReceiveProps con las propiedades futuras', nextProps)
36 }
```



# 4.7 - Ciclo de vida : actualización

- La actualización se produce cuando el componente ya generado se está actualizando.

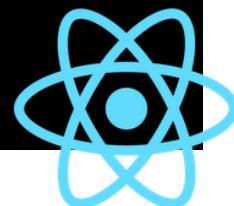
```
37
38 shouldComponentUpdate(nextProps, nextState) {
39   console.log('Ejecutando shouldComponentUpdate. Próximas propiedades y estado: ', nextProps, nextState)
40   // debo devolver un booleano
41   return true
42 }
43
44 //componentWillUpdate(nextProps, nextState) {
45 UNSAFE_componentWillUpdate(nextProps, nextState) {
46   console.log('Ejecutando UNSAFE_componentWillUpdate. Próximas propiedades y estado: ', nextProps, nextState)
47 }
48
49 componentDidUpdate(prevProps, prevState) {
50   console.log('Ejecutando componentDidUpdate. Anteriores propiedades y estado: ', prevProps, prevState)
51 }
```



# 4.8 - Ciclo de vida : desmontaje

- El desmontaje se produce cuando el componente se elimina del DOM.

```
52
53     componentWillMount() {
54         console.log('Se desmonta el componente...')
55     }
56
57     render() {
58         return (
59             <div>
60                 <p>Componente del ciclo de vida</p>
61                 <p>Estado: {this.state.estado}</p>
62                 <p>Propiedad: {this.props.propiedad}</p>
63             </div>
64         )
65     }
66 }
67
68 CicloVida.defaultProps = {
69     propiedad: 'Valor por defecto definido para la propiedad'
70 }
71
72 export default App;
73 }
```



# 4.8 - Ciclo de vida : desmontaje

- Prueba:

localhost:3001

## CICLO DE VIDA

Componente del ciclo de vida

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad

localhost:3001

Console tab selected.

```
[HMR] Waiting for update signal from WDS... log.js:24
Ejecuto constructor App.js:16
  ▶ {propiedad: "Valor por defecto definido para la propiedad"} ▶ {}
Se ejecuta UNSAFE_componentWillMount App.js:25
Se ejecuta componentDidMount App.js:30
>
```

## CICLO DE VIDA

Componente del ciclo de vida

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad

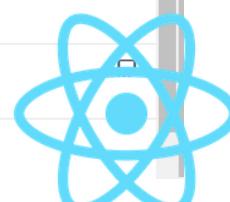
localhost:3001

Components tab selected.

Search bar: CicloVida

Component tree:  
App  
 CicloVida

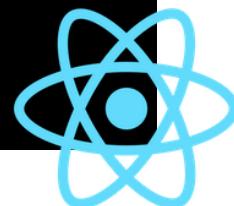
CicloVida component details:  
props:  
 propiedad: "Valor por defecto definido para la propiedad"  
 nr: ""  
state:  
 estado: "Inicializado en el constructor"  
rendered by:  
 App



## 4.8 - Ciclo de vida : desmontaje

Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App14.js, posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js:

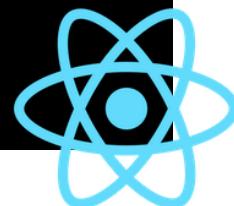
```
1 import React, {Component} from 'react';
2
3 class App extends Component{
4   render(){
5     return (
6       <div className="App">
7         <h1>CICLO DE VIDA</h1>
8         <CicloVida />
9         <UsarCicloVida />
10      </div>
11    );
12  }
13}
```



# 4.8 - Ciclo de vida : desmontaje

Cambio dinámico de una de las propiedades:

```
73 class UsarCicloVida extends Component {  
74   constructor(...args) {  
75     super(...args)  
76     this.state = {  
77       valorPropiedad: 'Test de valor de propiedad'  
78     }  
79   }  
80  
81   cambiarPropiedad() {  
82     this.setState({  
83       valorPropiedad: 'Otro valor'  
84     })  
85   }  
86  
87   render() {  
88     return (  
89       <div>  
90         <button onClick={this.cambiarPropiedad.bind(this)}>Cambiar propiedad</button>  
91         <CicloVida propiedad={this.state.valorPropiedad} />  
92       </div>  
93     )  
94   }  
95 }  
96  
97 export default App;
```



# 4.8 - Ciclo de vida : desmontaje

Cambio dinámico de una de las propiedades.

Prueba:

localhost:3001

## CICLO DE VIDA

Componente del ciclo de vida

Estado: Inicializado en el constructor

Propiedad: Valor por defecto definido para la propiedad

**Cambiar propiedad**

Componente del ciclo de vida

Estado: Inicializado en el constructor

Propiedad: Otro valor

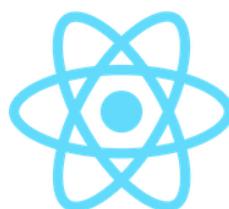
Screenshot of the Chrome DevTools Console tab showing logs related to component lifecycle events and prop changes.

Log Message	File	Line
[HMR] Waiting for update signal from WDS...	log.js	:24
Ejecuto constructor	App.js	:17
▶{propiedad: "Valor por defecto definido para la propiedad"} ▶{}		
Se ejecuta UNSAFE_componentWillMount	App.js	:26
Ejecuto constructor ▶{propiedad: "Test de valor de propiedad"} ▶{}	App.js	:17
Se ejecuta UNSAFE_componentWillMount	App.js	:26
② Se ejecuta componentDidMount	App.js	:31
Se ejecuta UNSAFE_componentWillReceiveProps con las propiedades futuras ▶{propiedad: "Otro valor"}	App.js	:36
Ejecutando shouldComponentUpdate. Próximas propiedades y estado:	App.js	:40
▶{propiedad: "Otro valor"} ▶{estado: "Inicializado en el constructor"}		
Ejecutando UNSAFE_componentWillUpdate. Próximas propiedades y estado:	App.js	:47
▶{propiedad: "Otro valor"} ▶{estado: "Inicializado en el constructor"}		
Ejecutando componentDidUpdate. Anteriores propiedades y estado:	App.js	:51
▶{propiedad: "Test de valor de propiedad"}		
▶{estado: "Inicializado en el constructor"}		

A blue atomic symbol icon is located in the bottom right corner of the screenshot.

# React Router

- 5.1 - Introducción e instalación de react router
- 5.2 - Declarando Router y Rutas
- 5.3 - Switch y página (error 404)
- 5.4 - Parámetros de la ruta
- 5.5 - Contenido dinámico a partir de la ruta
- 5.6 - Componente Link
- 5.7 - Menú con NavLink
- 5.8 - Props de React Router



## 5.1 - Introducción e instalación de react router

- Inicie una sesión de comandos del sistema operativo, siguiendo la siguiente secuencia de comandos:

Inicio → cmd → cd \ → cd react\_ej01

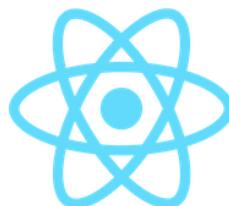
- Proceda a instalar el react router:

npm install react-router-dom

```
C:\Users\hduqu>cd\

C:\>cd react_ej01

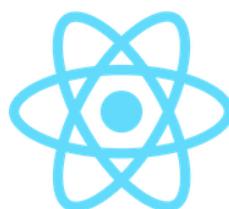
C:\react_ej01>npm install react-router-dom
[.....] \ rollbackFailedOptional: verb npm-session c544b54b367b49dd
```



## 5.2 - Declarando Router y Rutas

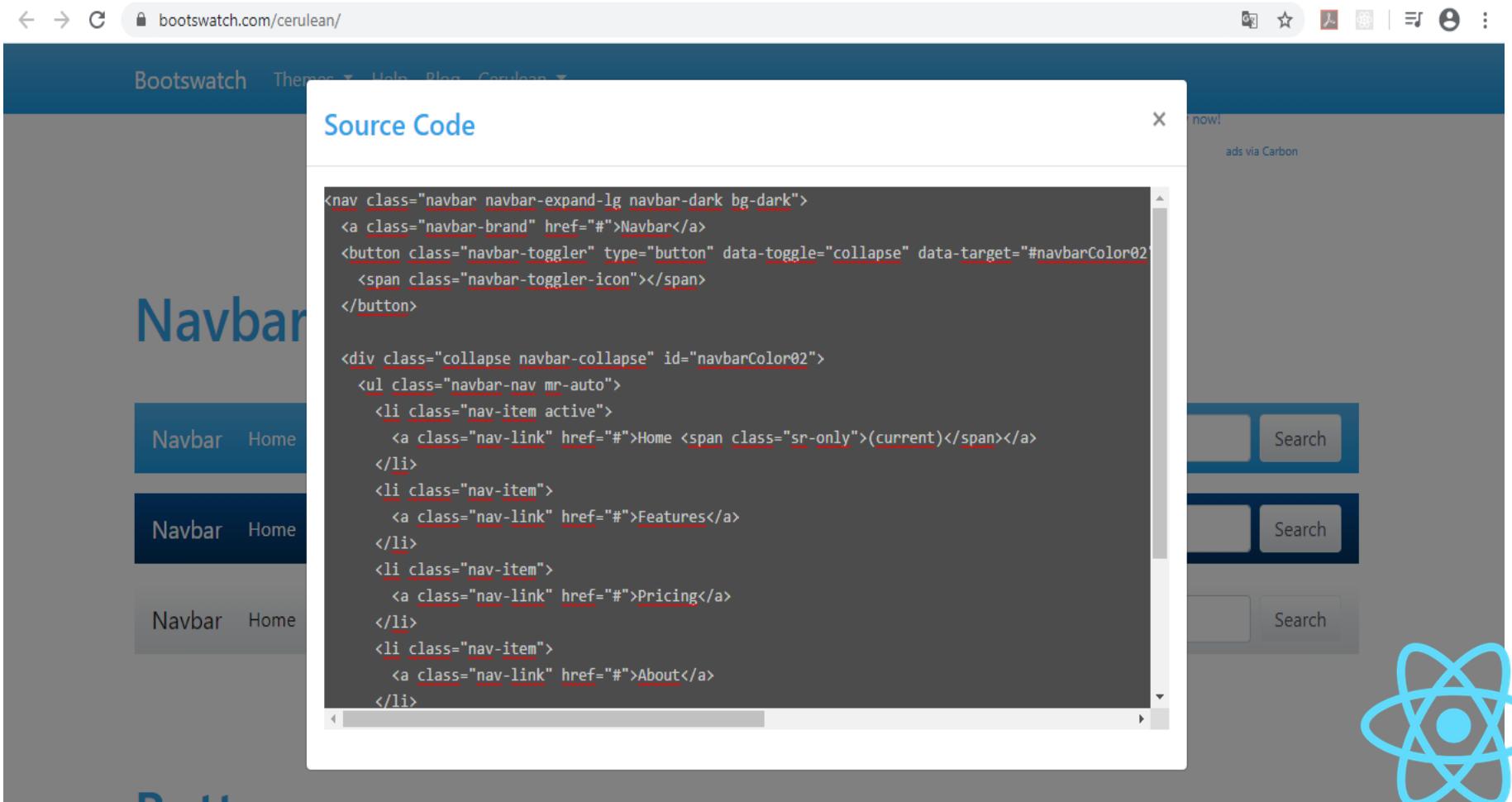
Siguiendo el procedimiento conocido, guarde el archivo `src/App.js`, como `src/App15.js`, posteriormente realice las siguientes actualizaciones sobre el archivo `src/App.js`. Tomando en cuenta la siguiente url, para seleccionar un estilo a aplicar en nuestro ejemplo:

<https://www.bootstrapcdn.com/bootswatch/>



## 5.2 - Declarando Router y Rutas

Seleccionar el estilo, de interés a fin de modificar la apariencia del menú a producir:



The screenshot shows a web browser displaying the [Bootswatch](https://bootswatch.com/cerulean/) theme "Cerulean". The page features a dark blue header with the "Bootswatch" logo and navigation links for "Themes", "Help", "Blog", and "Contribute". Below the header, there are three examples of the "Navbar" component. A modal window titled "Source Code" is open over the third example, displaying the HTML and CSS source code for the navbar. The code includes classes like "navbar navbar-expand-lg navbar-dark bg-dark" for the main container, "navbar-brand" for the logo, "navbar-toggler" for the collapse button, and "nav-item active" for the selected menu item. The modal has a white background and a close button in the top right corner. In the bottom right corner of the slide, there is a blue stylized atom or orbital icon.

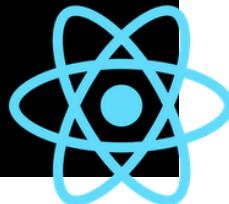
```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor02">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarColor02">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Features</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Pricing</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
    </ul>
  </div>
</nav>
```

## 5.2 - Declarando Router y Rutas

Modificar en el archivo public/index.html, la ruta de acceso al estilo e incorporar la siguiente ubicación mediante la etiqueta <link>:

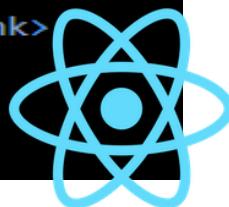
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <link rel="stylesheet"
7       type="text/css"
8       href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/cerulean/bootstrap.min.css">
9     <!-- primitive css
10    <link rel="stylesheet"
11      type="text/css"
12      href="https://taniarascia.github.io/primitive/css/main.css">
13    -->
```



## 5.2 - Declarando Router y Rutas

Modificar en el archivo src/App.js, con el siguiente contenido:

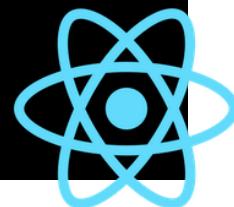
```
1 import React, {Component} from 'react';
2 import {
3   BrowserRouter as Router,
4   Switch,
5   Route,
6   Link
7 } from "react-router-dom";
8
9 class App extends Component{
10   render(){
11     return (
12       <div className="App">
13         <Router>
14           <div>
15             <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
16               <div className="collapse navbar-collapse" id="navbarColor02">
17                 <ul className="navbar-nav mr-auto">
18                   <li className="nav-item active">
19                     <Link className="nav-link" to="/">Inicio</Link>
20                   </li>
21                   <li className="nav-item">
22                     <Link className="nav-link" to="/acerca-de">Acerca de</Link>
23                   </li>
24                   <li className="nav-item">
25                     <Link className="nav-link" to="/usuarios">Usuarios</Link>
26                   </li>
27                 </ul>
28               </div>
29             </nav>
```



## 5.3 - Switch y página (error 404)

Continuación actualización en el archivo src/App.js:

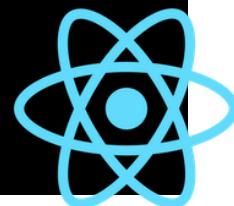
```
30      /* Un <Switch> mira a través de sus elementos secundarios <Route>s y
31      representa el primero que coincide con la URL actual. */
32      <Switch>
33          <Route path="/acercade">
34              <About />
35          </Route>
36          <Route path="/usuarios">
37              <Users />
38          </Route>
39          <Route path="/">
40              <Home />
41          </Route>
42      </Switch>
43      </div>
44  </Router>
45 </div>
46 );
47 }
48 }
```



## 5.3 - Switch y página (error 404)

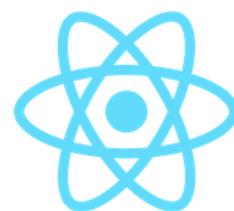
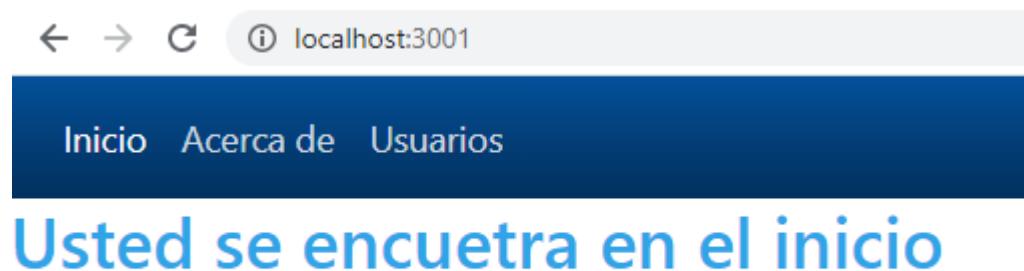
Continuación actualización en el archivo src/App.js:

```
49
50  function Home() {
51    return <h2>Usted se encuentra en el inicio</h2>;
52  }
53
54  function About() {
55    return <h2>Ud. selecciono acerca de</h2>;
56  }
57
58  function Users() {
59    return <h2>Ud. selecciono usuarios</h2>;
60  }
61
62  export default App;
63
```



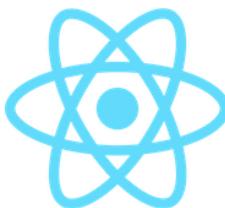
## 5.3 - Switch y página (error 404)

Prueba:



## 5.4 - Parámetros de la ruta

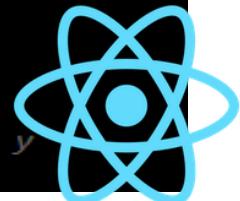
Siguiendo el procedimiento conocido, guarde el archivo `src/App.js`, como `src/App16.js`, posteriormente realice las siguientes actualizaciones sobre el archivo `src/App.js`.



## 5.4 - Parámetros de la ruta

Ejemplo Src/App.js:

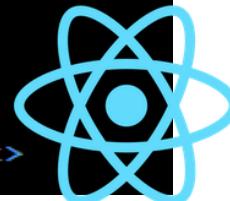
```
1 import React, {Component} from "react";
2 import {
3     BrowserRouter as Router,
4     Switch,
5     Route,
6     Link,
7     useRouteMatch,
8     useParams
9 } from "react-router-dom";
10
11 class App extends Component{
12     render(){
13         return (
14             <div className="App">
15                 <Router>
16                     <div>
17                         <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
18                             <div className="collapse navbar-collapse" id="navbarColor02">
19                                 <ul className="navbar-nav mr-auto">
20                                     <li className="nav-item active">
21                                         <Link className="nav-link" to="/">Inicio</Link>
22                                     </li>
23                                     <li className="nav-item">
24                                         <Link className="nav-link" to="/acerca-de">Acerca de</Link>
25                                     </li>
26                                     <li className="nav-item">
27                                         <Link className="nav-link" to="/temas">Temas</Link>
28                                     </li>
29                                 </ul>
30                             </div>
31                         </nav>
32                         {/* Un <Switch> mira a través de sus elementos secundarios <Route>s y
33                         representa el primero que coincide con la URL actual. */}
34                     </div>
35                 </Router>
36             </div>
37         )
38     }
39 }
```



# 5.4 - Parámetros de la ruta

## Continuación Src/App.js

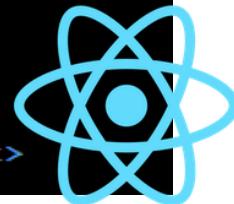
```
34          <Switch>
35            <Route path="/acercade">
36              <About />
37            </Route>
38            <Route path="/temas">
39              <Topics />
40            </Route>
41            <Route path="/">
42              <Home />
43            </Route>
44          </Switch>
45        </div>
46      </Router>
47    </div>
48  );
49}
50}
51
52 function Home() {
53   return <h2>Usted se encuentra en el inicio</h2>;
54 }
55
56 function About() {
57   return <h2>Ud. selecciono acerca de</h2>;
58 }
59
60 function Topics() {
61   let match = useRouteMatch();
62   return (
63     <div>
64       <h2>Usted ha seleccionad temas</h2>
65       <ul>
66         <li>
67           <Link to={`${match.url}/components`}>Componentes</Link>
68         </li>
```



# 5.5 - Contenido dinámico a partir de la ruta

## Continuación Src/App.js

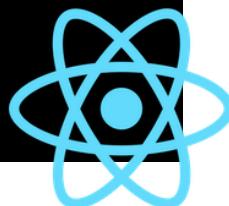
```
34          <Switch>
35            <Route path="/acercade">
36              <About />
37            </Route>
38            <Route path="/temas">
39              <Topics />
40            </Route>
41            <Route path="/">
42              <Home />
43            </Route>
44          </Switch>
45        </div>
46      </Router>
47    </div>
48  );
49}
50}
51
52 function Home() {
53   return <h2>Usted se encuentra en el inicio</h2>;
54 }
55
56 function About() {
57   return <h2>Ud. selecciono acerca de</h2>;
58 }
59
60 function Topics() {
61   let match = useRouteMatch();
62   return (
63     <div>
64       <h2>Usted ha seleccionad temas</h2>
65       <ul>
66         <li>
67           <Link to={`${match.url}/components`}>Componentes</Link>
68         </li>
```



# 5.6 - Componente Link

## Continuación Src/App.js

```
69      <li>
70        <Link to={`${match.url}/props-y-state`}>
71          Props y State
72        </Link>
73      </li>
74    </ul>
75
76    /* La página Temas tiene su propio <Switch> con más rutas
77     que se basan en la ruta URL / topics. Puedes pensar en el
78     2da <Ruta> aquí como una página de "índice" para todos los temas, o
79     la página que se muestra cuando no se selecciona ningún tema */
80  <Switch>
81    <Route path={`${match.path}/:topicId`}>
82      <Topic />
83    </Route>
84    <Route path={match.path}>
85      <h3>Por favor seleccione un tema.</h3>
86    </Route>
87  </Switch>
88 </div>
89  );
90}
91
92 function Topic() {
93   let { topicId } = useParams();
94   return <h3>Tema requerido ID: {topicId}</h3>;
95 }
96
97 export default App;
98
```



## 5.7 - Menú con NavLink

Cuando necesite usar atributos de estilo o clase en activo <Link>, entonces puede usar <NavLink>

Veamos el ejemplo:

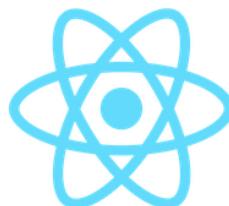
Enlace

```
<Link to="/">Home</Link>NavLink
```

```
<NavLink to="/" activeClassName="active">Home</NavLink>
```

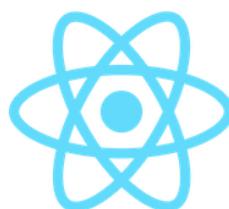
Simplemente, cuando usa <Link> no hay ninguna clase **activa** en el elemento seleccionado.

En contraste, con <NavLink> el elemento seleccionado se resalta porque a este elemento se le agrega una clase activa.



# Peticiones HTTP

- 6.1 - Atomic Design
- 6.2 - Refactorizando código
- 6.3 - Metodología atomic design
- 6.4 - Peticiones a una API REST
- 6.5 - Peticiones con fetch
- 6.6 - Actualizando el estado con la respuesta
- 6.7 - Reescribiendo peticiones con axios
- 6.8 - API Ejemplo



## 6.1 - Atomic Design

Atomic Design es una metodología de diseño basada en el principio de separar los componentes en diferentes carpetas dependiendo de su función en la aplicación.

<https://bradfrost.com/blog/post/atomic-web-design/>

The screenshot shows a web browser displaying the URL <https://bradfrost.com/blog/post/atomic-web-design/>. The page features a decorative header with a pattern of overlapping circles in white, orange, and blue. On the right side, there's a sidebar with the text "my name is brad frost" and a navigation menu with links to "work", "workshops", "book", "blog", and "contact". The main content area has a large, bold title "atomic design". Below the title, there's a paragraph about the author's book and a quote by Stephen Hay. A small atomic symbol icon is in the bottom right corner.

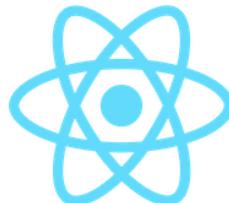
Hey there! I wrote a book called *Atomic Design* that dives into this topic in more detail, which you can buy as an [ebook](#).

*We're not designing pages, we're designing systems of components.—[Stephen Hay](#)*

## 6.2 - Refactorizando código

Para reestructurar un componente de clase en un componente de función, debemos considerar que un componente de función de JavaScript simple no tiene estado , con la intención de mantener el enfoque en la presentación y mejorar la reutilización a través de un enfoque modular. Esto significa que tenemos que mantener nuestro estado en un componente principal y transmitirlo a nuestros componentes de función como accesorios (props) .

Para ilustrar esto en nuestro componente de formulario, encapsulemos los elementos de presentación, devueltos por el método de representación, dentro de un componente de función separado que recibe el nombre de usuario y la contraseña como accesorios. Ok, pero ¿cómo cambiamos el estado de los padres? ¡Accesorios (props)!

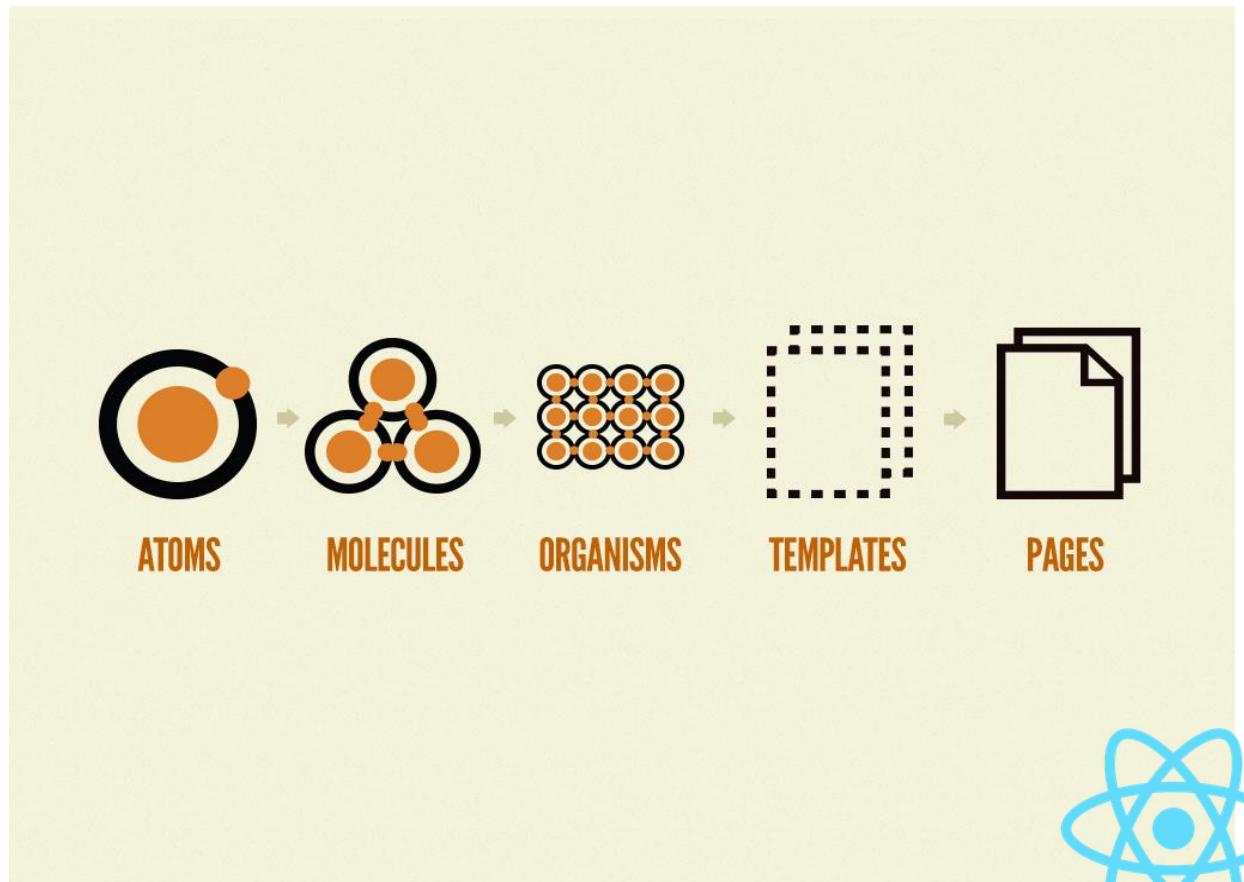


## 6.3 - Metodología atomic design

¿Qué es el diseño atómico?

El diseño atómico es una metodología para crear sistemas de diseño. Hay cinco niveles distintos en el diseño atómico:

- Átomos
- Moléculas
- Organismos
- Plantillas
- Páginas



# 6.3 - Atomic Design

## Periodic Table of the Elements

html																			col	table
head	span																			
title	a																			
meta	rt	dfn	em	i	small	ins	s	br	p	blockquote	legend	optgroup	address	h3	nav	menu	th			
base	rp	abbr	time	b	strong	del	kbd	hr	ol	dl	label	option	datalist	h4	article	command	tbody			
link	noscript	q	var	sub	mark	bdi	wbr	figcaption	ul	dt	input	output	keygen	h5	footer	summary	thead			
style	script	cite	samp	sup	ruby	bdo	code	figure	li	dd	textarea	button	progress	h6	hgroup	details	tfoot			

img	area	map	embed	object	param	source	iframe	canvas	track	audio	video
-----	------	-----	-------	--------	-------	--------	--------	--------	-------	-------	-------

Root element

Metadata and scripting

Embedding content

Text-level semantics

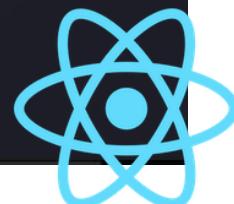
Grouping content

Forms

Document sections

Tabular data

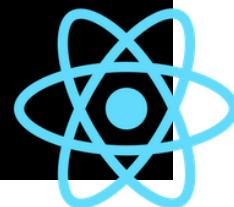
Interactive elements



## 6.3 - Atomic Design

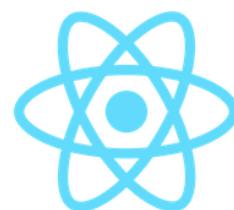
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App17.js, posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js.

```
1 import React, {Component} from "react";
2 import Login from './Login'
3
4 class App extends Component {
5   render() {
6     return (
7       <div className="App">
8         <Login/>
9       </div>
10    );
11  }
12}
13
14 export default App;
```



## 6.3 - Atomic Design

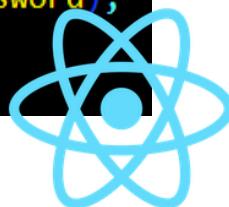
Para administrar la lógica detrás de nuestro formulario, podríamos guardar nuestros valores de entrada en el estado del componente y llamar a un método apropiado cuando se active el evento `onChange` , actualizando nuestros valores con estado:



# 6.3 - Atomic Design

Crear el archivo src/Login.js. Contenido:

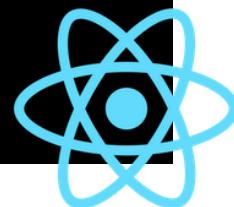
```
1 import React, { Component } from "react";
2
3 class Login extends Component {
4     constructor(props) {
5         super(props);
6         this.state = {
7             username: "",
8             password: ""
9         };
10    }
11
12    handleChangeUsername = (event) => {
13        this.setState({ username: event.target.value });
14    }
15
16    handleChangePassword = (event) => {
17        this.setState({ password: event.target.value });
18    }
19
20    handleSubmit = (event) => {
21        event.preventDefault(); // |
22        alert('Login efectuado: ' + this.state.username + ', ' + this.state.password);
23    }
24}
```



# 6.3 - Atomic Design

## Continuación contenido archivo src/Login.js :

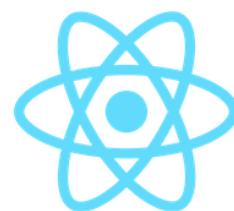
```
25 render() {
26     return (
27         <div className="container" style={{marginLeft:'30%'}}>
28             <form onSubmit={this.handleSubmit} className='jumbotron' style={{width: '40%'}}>
29                 <h1 className="display-3">Login</h1>
30                 <div className='form-items'>
31                     <div className='form-group'>
32                         <input
33                             name="email"
34                             type="text"
35                             placeholder="correo@dominio.com"
36                             onChange={this.handleChangeUsername}
37                             value={this.username}
38                             className='form-control'
39                             required/>
40                     </div>
41                     <div className='form-group'>
42                         <input
43                             name="password"
44                             type="password"
45                             placeholder="Ingrese su password"
46                             onChange={this.handleChangePassword}
47                             value={this.password}
48                             className='form-control'
49                             required/>
50                     </div>
51                 </div>
52                 <button type="submit" className="btn btn-primary btn-lg btn-block" >
53                     Ingresar
54                 </button>
55             </form>
56         </div>
57     );
58 }
59 }
60 export default Login;
```



# 6.3 - Atomic Design

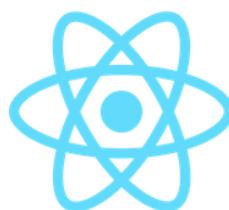
Prueba de src/App.js y src/Login.js :

A screenshot of a web browser window. The address bar shows 'localhost:3001'. The main content area displays a 'Login' form. The form has two input fields: the first contains 'henryaduquea@gmail.com' and the second contains '.....'. Below the inputs is a large blue button labeled 'Ingresar'. A modal dialog box is overlaid on the page, containing the text 'localhost:3001 dice' and 'Login efectuado: henryaduquea@gmail.com, 1234567' followed by a blue 'Aceptar' button.



## 6.3 - Atomic Design

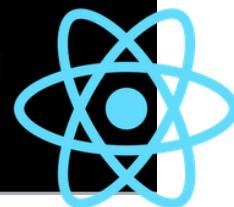
Siguiendo el procedimiento conocido, guarde el archivo `src/App.js`, como `src/App18.js`, posteriormente realice las siguientes actualizaciones sobre el archivo `src/App.js`.



## 6.3 - Atomic Design

Modifique el archivo src/App.js, con el siguiente contenido:

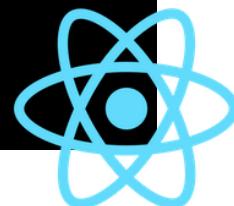
```
1 import React, { Component, useState } from "react";
2
3 class App extends Component {
4     render() {
5         return (
6             <div className="App">
7                 <LoginForm />
8             </div>
9         );
10    }
11 }
12
13 function LoginForm(props){
14     const [username, setUserName] = useState("");
15     const [password, setPassword] = useState("");
16     const handleSumit = (event) =>{
17         event.preventDefault(); //
18         alert('Login enviado:' +username+ '+' +password);
19         // Código Lógica de negocio
20     }
21     return (
22         <div className="container" style={{marginLeft: '30%'}}>
23             <form onSubmit={handleSumit} className='jumbotron' style={{width: '40%'}}>
24                 <h1 className="display-3">Login</h1>
25                 <div className='form-items'>
26                     <div className='form-group'>
27                         <input
28                             name="email"
29                             type="text"
30                             placeholder="correo@dominio.com"
31                             onChange={(event) => setUserName(event.target.value)}
32                             value={username}
33                             className='form-control'
34                             required/>
35                 </div>
36             </form>
37         </div>
38     );
39 }
40
41 export default App;
```



## 6.3 - Atomic Design

Continuación archivo src/App.js:

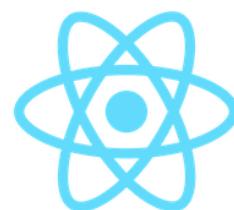
```
36          <div className='form-group'>
37              <input
38                  name="password"
39                  type="password"
40                  placeholder="Ingrese su password"
41                  onChange={(event) => setPassword(event.target.value)}
42                  value={password}
43                  className='form-control'
44                  required
45              />
46          </div>
47      </div>
48      <button type="submit" className="btn btn-primary btn-lg btn-block" >
49          Ingresar
50      </button>
51  </form>
52</div>
53);
54}
55
56export default App;
```



# 6.3 - Atomic Design

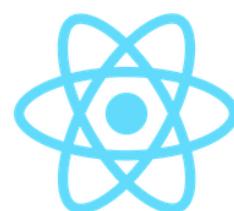
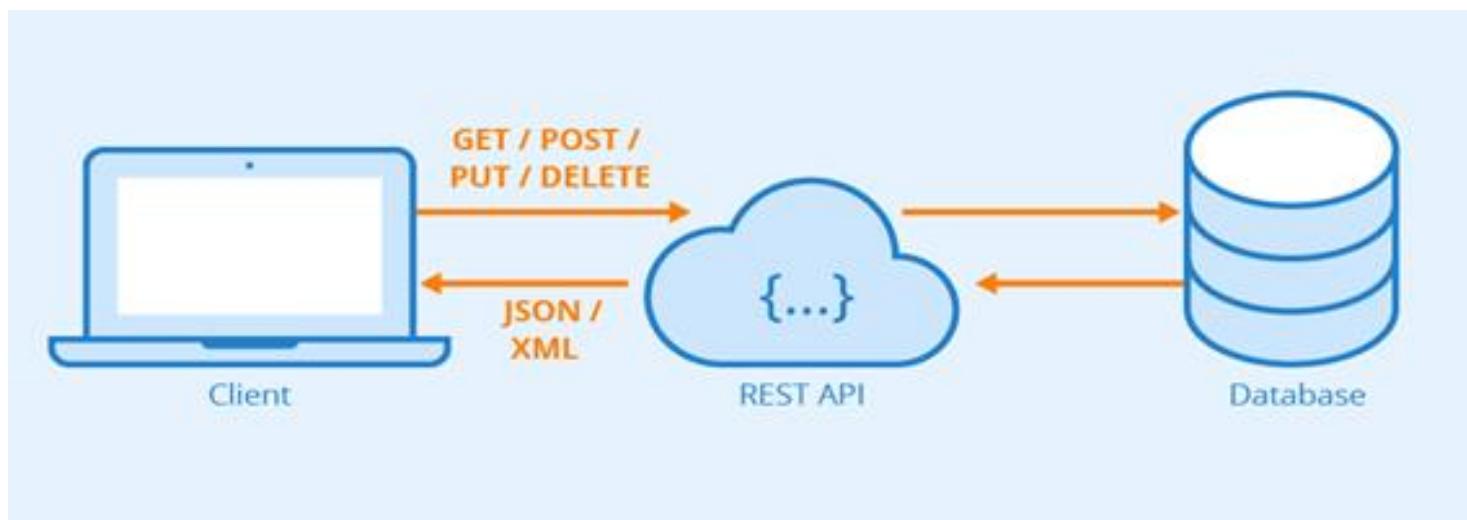
Prueba src/App.js:

A screenshot of a web browser window displaying a login form. The URL bar shows 'localhost:3001'. A modal dialog box is overlaid on the page, containing the text 'localhost:3001 dice' and 'Login enviado:henryaduquea@gmail.com+ ABCDEF' with a blue 'Aceptar' button. Below the modal, the word 'Login' is partially visible. The main form has two input fields: one with the email 'henryaduquea@gmail.com' and another with masked password '.....'. A large blue button labeled 'Ingresar' is at the bottom.



## 6.4 - Peticiones a una API REST

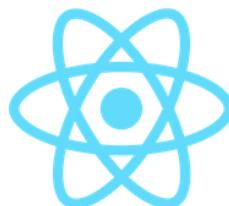
REST: La transferencia de estado representacional (en inglés *representational state transfer*) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.



## 6.4 - Peticiones a una API REST

REST: La transferencia de estado representacional (en inglés representational state transfer) o REST es un estilo de arquitectura software para sistemas hipermédia distribuidos como la World Wide Web.

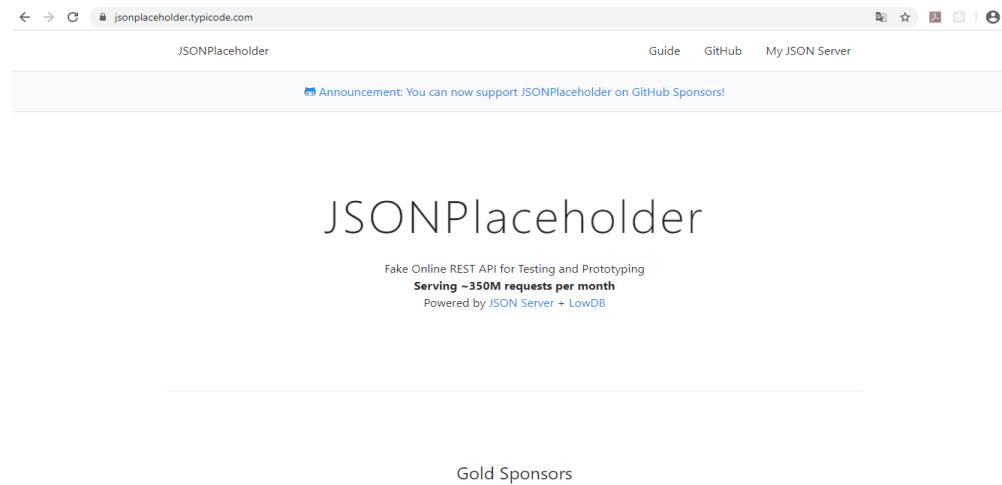
Siguiendo el procedimiento conocido, guarde el archivo src/App.js, como src/App19.js, posteriormente realice las siguientes actualizaciones sobre el archivo src/App.js.



## 6.4 - Peticiones a una API REST

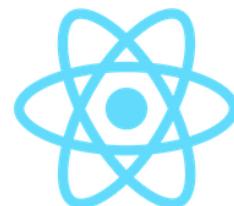
Se tomará como muestra, un origen de datos en formato json:

<https://jsonplaceholder.typicode.com/>



De esta ubicación se explorará un ejemplo de usuarios:

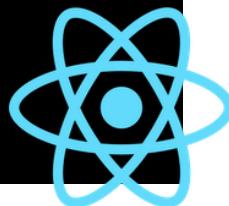
<https://jsonplaceholder.typicode.com/users/>



# 6.5 - Peticiones con fetch

Contenido desarrollo src/App.js:

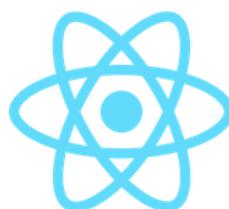
```
1 import React, {Component} from "react"
2 import Table from './UsuarioList'
3
4 class App extends Component {
5   constructor(props) {
6     super(props)
7     this.state = { usuarios: [] }
8   }
9   UNSAFE_componentWillMount() {
10     fetch('https://jsonplaceholder.typicode.com/users')
11       .then((response) => {
12         return response.json()
13       })
14       .then((extraerUsuarios) => {
15         this.setState({ usuarios: extraerUsuarios })
16       })
17   }
18   render() {
19     if (this.state.usuarios.length > 0) {
20       return (
21         <div className="container">
22           <Table tablaDatos={this.state.usuarios} />
23         </div>
24       )
25     } else {
26       return <p className="text-center">Cargando usuarios...</p>
27     }
28   }
29 }
30 export default App;
```



## 6.5 - Peticiones con fetch

La [API Fetch](#) proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global [fetch\(\)](#) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

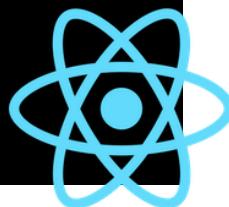
El método `fetch()` toma un argumento obligatorio, la ruta al recurso que desea obtener. Devuelve un [Promise](#) que resuelve a esa solicitud [Response](#), ya sea exitosa o no



# 6.6 - Actualizando el estado con la respuesta

Contenido desarrollo src/App.js:

```
1 import React, {Component} from "react"
2 import Table from './UsuarioList'
3
4 class App extends Component {
5   constructor(props) {
6     super(props)
7     this.state = { usuarios: [] }
8   }
9   UNSAFE_componentWillMount() {
10     fetch('https://jsonplaceholder.typicode.com/users')
11       .then((response) => {
12         return response.json()
13       })
14       .then((extraerUsuarios) => {
15         this.setState({ usuarios: extraerUsuarios })
16       })
17   }
18   render() {
19     if (this.state.usuarios.length > 0) {
20       return (
21         <div className="container">
22           <Table tablaDatos={this.state.usuarios} />
23         </div>
24       )
25     } else {
26       return <p className="text-center">Cargando usuarios...</p>
27     }
28   }
29 }
30 export default App;
```

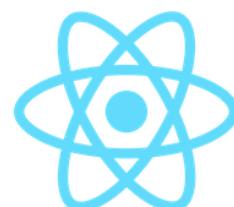


## 6.6 - Actualizando el estado con la respuesta

Prueba src/App.js:

← → ⌂ ⓘ localhost:3001

Nombre	Id Usuario	Email
Leanne Graham	Bret	Sincere@april.biz
Ervin Howell	Antonette	Shanna@melissa.tv
Clementine Bauch	Samantha	Nathan@yesenia.net
Patricia Lebsack	Karianne	Julianne.OConner@kory.org
Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca
Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info
Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz
Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me
Glenna Reichert	Delphine	Chaim_McDermott@dana.io
Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz



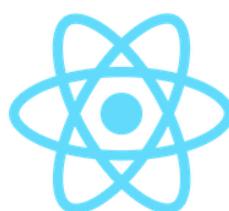
## 6.7 - Reescribiendo peticiones con axios

Primero Axios debe ser instalado:

```
C:\react_ej01>npm install axios --save
```

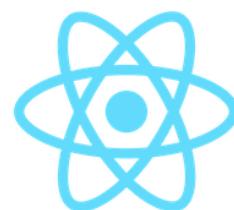
Axios es un librería Javascript capaz de ejecutarse tanto en el navegador como en NodeJS, que facilita todo tipo de operaciones como cliente HTTP.

La librería está basada en promesas, por lo que al ser usada se podrá generar código de programación asíncrono.



## 6.7 - Reescribiendo peticiones con axios

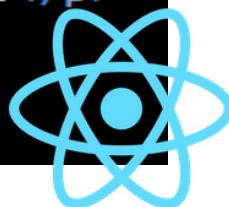
Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App20.js, efectue una copia de src/UsuarioList.js como src/UsuarioListAxio.js, posteriormente realice las siguientes actualizaciones sobre los archivo src/App.js y src/UsuarioListAxio.js:



# 6.7 - Reescribiendo peticiones con axios

## Actualización archivo src/App.js:

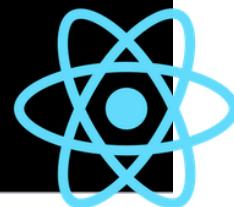
```
1 import React, {Component} from "react";
2 import axios from 'axios';
3 import Table from './UsuarioListAxio';|
4
5 class App extends Component {
6     state = {
7         usuarios: []
8     }
9
10    componentDidMount() {
11        axios.get('https://jsonplaceholder.typicode.com/users')
12            .then(res => {
13                const usuarios = res.data;
14                this.setState({ usuarios });
15            })
16    }
17
18    render() {
19        if (this.state.usuarios.length > 0) {
20            return (
21                <div className="container">
22                    <Table tablaDatos={this.state.usuarios} />
23                </div>
24            )
25        } else {
26            return <p className="text-center">Cargando usuarios...</p>
27        }
28    }
29}
30 export default App;
```



# 6.7 - Reescribiendo peticiones con axios

## Actualización src/UsuarioListAxio.js:

```
1 import React, {Component} from 'react';
2
3 const TableHeader = () =>{
4     return(
5         <thead>
6             <tr>
7                 <th>Nombre</th>
8                 <th>Id Usuario</th>
9                 <th>Email</th>
10            </tr>
11        </thead>
12    )
13}
14
15 const TableBody = props => {
16     const rows = props.tablaDatos.map((row,index) => {
17         return (
18             <tr key={index}>
19                 <td>{row.name}</td>
20                 <td>{row.username}</td>
21                 <td>{row.email}</td>
22             </tr>
23         )
24     })
25     return <tbody>{rows}</tbody>
26 }
27
28 class Table extends Component{
29     render(){
30         const {tablaDatos} = this.props
31         return (
32             <table className="table table-hover">
33                 <TableHeader />
34                 <TableBody tablaDatos={tablaDatos}/>
35             </table>
36         )
37     }
38 }
39 export default Table;
```

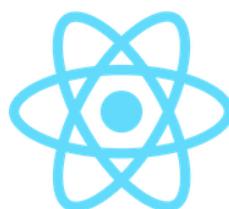


# 6.7 - Reescribiendo peticiones con axios

## Prueba src/App.js y src/UsuarioListAxio.js:

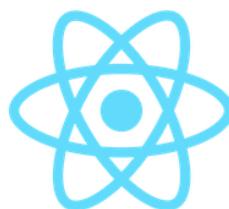
localhost:3001

Nombre	Id Usuario	Email
Leanne Graham	Bret	Sincere@april.biz
Ervin Howell	Antonette	Shanna@melissa.tv
Clementine Bauch	Samantha	Nathan@yesenia.net
Patricia Lebsack	Karianne	Julianne.OConner@kory.org
Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca
Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info
Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz
Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me
Glenna Reichert	Delphine	Chaim_McDermott@dana.io
Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz



## 6.8 - API Ejemplo

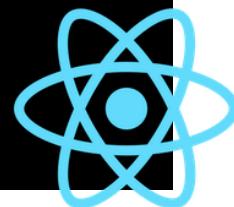
Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App21.js. Posteriormente efectue una copia del archivo src/UsuarioList.js como src/UsuarioListAPI.js



# 6.8 - API Ejemplo

Ejemplo src/UsuarioListAPI.js:

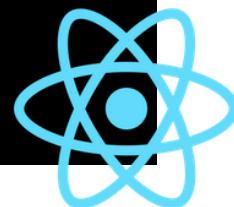
```
1 import React, {Component} from 'react';
2
3 const TableHeader = () =>{
4     return(
5         <thead>
6             <tr>
7                 <th>Id</th>
8                 <th>Nombre</th>
9                 <th>Id Usuario</th>
10                <th>Email</th>
11            </tr>
12        </thead>
13    )
14}
15
16 const TableBody = props => {
17     const rows = props.tablaDatos.map((row,index) => {
18         return (
19             <tr key={index}>
20                 <td>{row.id}</td>
21                 <td>{row.name}</td>
22                 <td>{row.username}</td>
23                 <td>{row.email}</td>
24             </tr>
25         )
26     })
27     return <tbody>{rows}</tbody>
28 }
29
30 class Table extends Component{
31     render(){
32         const {tablaDatos} = this.props
33         return (
34             <table className="table table-hover">
35                 <TableHeader />
36                 <TableBody tablaDatos={tablaDatos}/>
37             </table>
38         )
39     }
40 }
41 export default Table;
```



# 6.8 - API Ejemplo

## Ejemplo src/App.js:

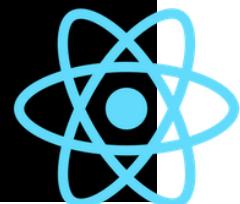
```
1 import React, {Component} from 'react';
2 import axios from 'axios';
3 import {
4     BrowserRouter as Router,
5     Switch,
6     Route,
7     Link
8 } from "react-router-dom";
9 import Table from './UsuarioListAPI';
10
11 class App extends Component{
12     render(){
13         return (
14             <div className="App">
15                 <h1>EJEMPLO API</h1>
16                 <Router>
17                     <div>
18                         <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
19                             <div className="collapse navbar-collapse" id="navbarColor02">
20                                 <ul className="navbar-nav mr-auto">
21                                     <li className="nav-item active">
22                                         <Link className="nav-link" to="/">Inicio</Link>
23                                     </li>
24                                     <li className="nav-item">
25                                         <Link className="nav-link" to="/agregar">Agregar</Link>
26                                     </li>
27                                     <li className="nav-item">
28                                         <Link className="nav-link" to="/borrar">Borrar</Link>
29                                     </li>
30                                     <li className="nav-item">
31                                         <Link className="nav-link" to="/reporte">Reporte</Link>
32                                     </li>
33                                 </ul>
34                             </div>
35                         </nav>
```



# 6.8 - API Ejemplo

Continuación src/App.js:

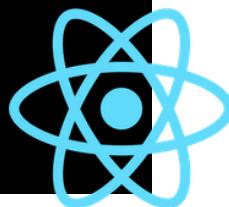
```
-- 36      /* Un <Switch> mira a través de sus elementos secundarios <Route>s y
37      representa el primero que coincide con la URL actual. */
38  <Switch>
39    <Route path="/agregar">
40      <PersonasAgregar />
41    </Route>
42    <Route path="/borrar">
43      <PersonasBorrar />
44    </Route>
45    <Route path="/reporte">
46      <PersonasReporte />
47    </Route>
48    <Route path="/">
49      <Inicio />
50    </Route>
51  </Switch>
52 </div>
53 </Router>
54 </div>
55 );
56 }
57 }
58
59 class PersonasAgregar extends Component {
60   state = {
61     name: '',
62   }
63
64   handleChange = event => {
65     this.setState({ name: event.target.value });
66   }
67
68   handleSubmit = event => {
69     event.preventDefault();
70
71     const usuario = {
72       name: this.state.name
73     };
74
75     axios.post('https://jsonplaceholder.typicode.com/users', { usuario })
76       .then(res => {
77         console.log(res);
78         console.log(res.data);
79       })
80   }
}
```



# 6.8 - API Ejemplo

Continuación src/App.js:

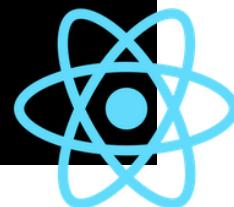
```
81
82
83     render() {
84         return (
85             <div>
86                 <form onSubmit={this.handleSubmit}>
87                     <label>
88                         Nombre:
89                         <input type="text" name="name" onChange={this.handleChange} />
90                     </label>
91                     <button type="submit">Agregar</button>
92                 </form>
93             </div>
94         )
95     }
96
97
98     class PersonasBorrar extends Component {
99         state = {
100             id: '',
101         }
102
103         handleChange = event => {
104             this.setState({ id: event.target.value });
105         }
106
107         handleSubmit = event => {
108             event.preventDefault();
109
110             axios.delete('https://jsonplaceholder.typicode.com/users/${this.state.id}')
111                 .then(res => {
112                     console.log(res);
113                     console.log(res.data);
114                 })
115         }
116
117         render() {
118             return (
119                 <div>
120                     <form onSubmit={this.handleSubmit}>
121                         <label>
122                             ID De la Persona:
123                             <input type="text" name="id" onChange={this.handleChange} />
124                         </label>
125                         <button type="submit">Borrar</button>
126                     </form>
127                 </div>
128             )
129         }
130     }
131
132     render() {
133         return (
134             <div>
135                 <h1>CRUD con API REST</h1>
136                 <hr/>
137                 <h2>Crear una Persona</h2>
138                 <hr/>
139                 <h2>Borrar una Persona</h2>
140             </div>
141         )
142     }
143 }
```



# 6.8 - API Ejemplo

Continuación src/App.js:

```
129     }
130 }
131
132
133 class PersonasReporte extends Component {
134     state = {
135         usuarios: []
136     }
137
138     componentDidMount() {
139         axios.get('https://jsonplaceholder.typicode.com/users')
140             .then(res => {
141                 const usuarios = res.data;
142                 this.setState({ usuarios });
143             })
144     }
145
146     render() {
147         if (this.state.usuarios.length > 0) {
148             return (
149                 <div className="container">
150                     <Table tablaDatos={this.state.usuarios} />
151                 </div>
152             )
153         } else {
154             return <p className="text-center">Cargando usuarios...</p>
155         }
156     }
157 }
158
159
160 class Inicio extends Component{
161     render(){
162         return(
163             <h2>Inicio de la API</h2>
164         )}
165     }
166
167     export default App;
168 }
```



# 6.8 - API Ejemplo

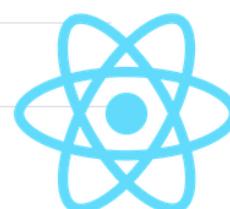
Prueba src/App.js:

localhost:3001/reporte

## EJEMPLO API

Inicio Agregar Borrar Reporte

Id	Nombre	Id Usuario	Email
1	Leanne Graham	Bret	Sincere@april.biz
2	Ervin Howell	Antonette	Shanna@melissa.tv
3	Clementine Bauch	Samantha	Nathan@yesenia.net
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca
6	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info
7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz
8	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me
9	Glenna Reichert	Delphine	Chaim_McDermott@dana.io



# **Componentes de orden superior**

7.1 - Crear servidor JSON

7.2 - Agregando loader a nuestras peticiones

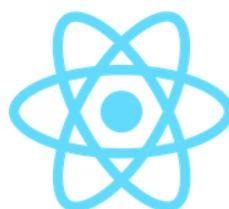
7.3 - Eliminando inconsistencias en el proyecto

7.4 - Escribiendo un componente de orden superior

7.5 - Agregando lógica reutilizable al HOC

7.6 - Bonus: Ejemplo de páginas privadas

7.7 - Bonus: Subir imágenes con un HOC



## 7.1 - Crear servidor JSON

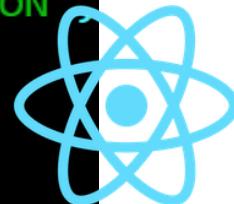
JSON Server lo ayuda a configurar una API REST de manera simple y rápida.

- Instalación de json-Server:

```
C:\react_ej01>npm install json-server
```

- Dentro de la carpeta del proyecto /react\_ej01 (raíz de esta carpeta) se creará el archivo "db.json" con los siguiente contenido:

```
1  {
2    "posts": [
3      {
4        "title": "Cuso de React JS, nivel 1",
5        "tags": "[Henry]"
6      },
7      {
8        "title": "Definiendo un servidor para formato JSON",
9        "tags": "[Duque]"
10     }
11   ]
12 }
```



## 7.1 - Crear servidor JSON

Nota: Si archivo “db.json” no existe, json-server, creará un archivo con un contenido por omisión.

- Proceda a levantar el servidor JSON, efectuando el comando:

```
>npx json-server --watch db.json --port 3011
```

```
C:\react_ej01>npx json-server --watch db.json --port 3011

\{^_^}/ hi!

Loading db.json
Done

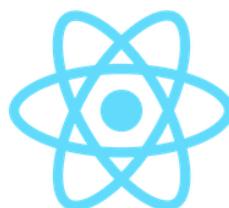
Resources
http://localhost:3011/posts

Home
http://localhost:3011

Type s + enter at any time to create a snapshot of the database
Watching...
```

- Proceda a levantar la aplicación:

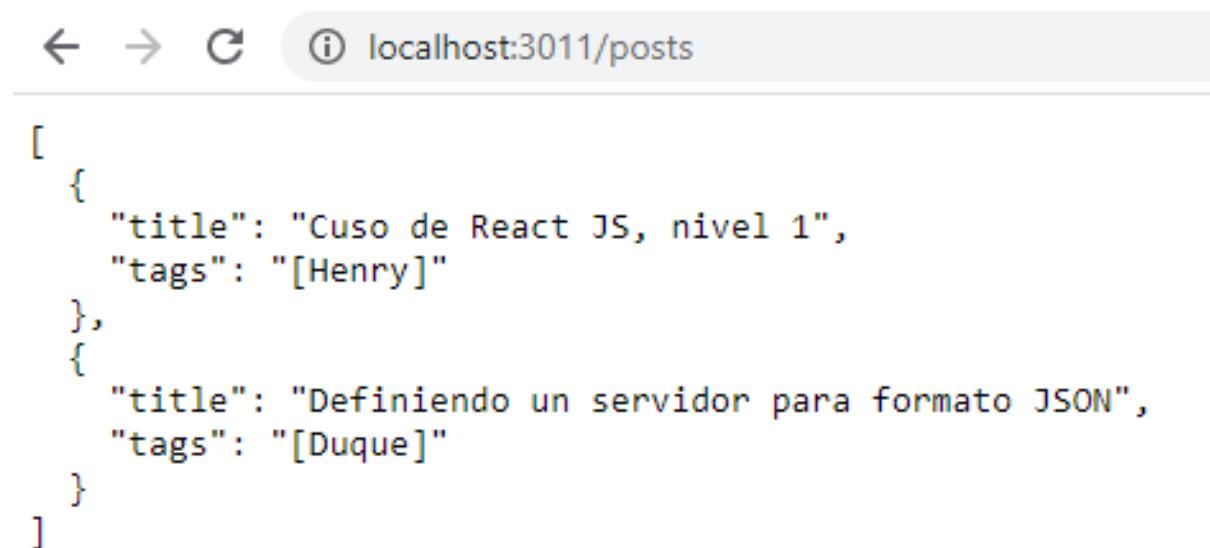
```
>npm start
```



## 7.1 - Crear servidor JSON

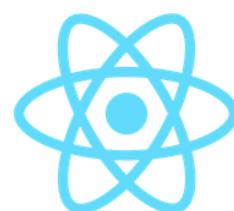
- Prueba: Consiste en el ejecutar la siguiente URL, desde el navegador. Observe el resultado:

<http://localhost:3011/posts>



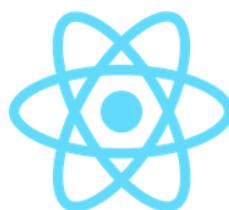
A screenshot of a web browser window. The address bar shows the URL "localhost:3011/posts". The main content area displays a JSON array of two posts. Each post is an object with "title" and "tags" properties.

```
[  
  {  
    "title": "Cuso de React JS, nivel 1",  
    "tags": "[Henry]"  
  },  
  {  
    "title": "Definiendo un servidor para formato JSON",  
    "tags": "[Duque]"  
  }]
```



## 7.1 - Crear servidor JSON

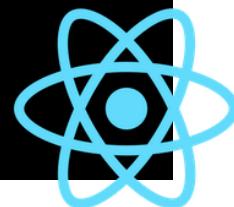
Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App22.js. Realice los siguiente cambios sobre src/App.js



# 7.1 - Crear servidor JSON

src/App.js:

```
1 import React, {Component} from 'react';
2
3 class App extends Component{
4     constructor(props){
5         super(props);
6         this.state = {
7             posts:[]
8         };
9     }
10    componentDidMount(){
11        let url = 'http://localhost:3011/posts'
12        fetch(url)
13            .then(resp => resp.json())
14            .then(data => {
15                let posts = data.map((post, index) =>{
16                    return (
17                        <div className="alert alert-dismissible alert-warning" key={index}>
18                            <h3 className="alert-heading">{post.title}</h3>
19                            <p className="mb-0">Tags: {post.tags}</p>
20                        </div>
21                    )
22                })
23                this.setState({posts: posts});
24            })
25        }
26    render(){
27        return (
28            <div className="jumbotron">
29                {this.state.posts}
30            </div>
31        );
32    }
33}
34
35 export default App;
```



# 7.1 - Crear servidor JSON

Prueba: src/App.js:

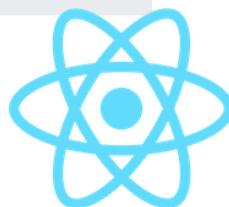
A screenshot of a web browser window displaying two cards on localhost:3001. The browser interface includes standard navigation buttons, a search bar with 'localhost:3001', and a toolbar with icons for refresh, search, and other functions.

The first card has a light orange background and contains the following text:

**Cuso de React JS, nivel 1**  
Tags: [Henry]

The second card has a light orange background and contains the following text:

**Definiendo un servidor para formato JSON**  
Tags: [Duque]



## 7.2 - Agregando loader a nuestras peticiones

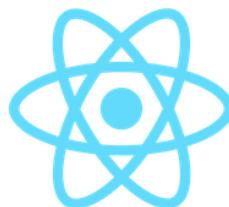
Consiste en incluir la siguiente API:

[https://upload.wikimedia.org/wikipedia/commons/b/b1>Loading\\_icon.gif](https://upload.wikimedia.org/wikipedia/commons/b/b1>Loading_icon.gif)

En las peticiones de la aplicación.

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App23.js. Realice los siguiente cambios sobre src/App.js

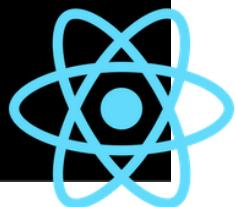
Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App23.js. Realice los siguiente cambios sobre src/App.js



# 7.2 - Agregando loader a nuestras peticiones

## Ejemplo src/App.js:

```
1 import React, {Component} from "react"
2 import Table from './UsuarioList'
3
4 class App extends Component {
5   constructor(props) {
6     super(props)
7     this.state = { usuarios: [] }
8   }
9   UNSAFE_componentWillMount() {
10     fetch('https://jsonplaceholder.typicode.com/users')
11       .then((response) => {
12         return response.json()
13       })
14       .then((extraerUsuarios) => {
15         this.setState({ usuarios: extraerUsuarios })
16       })
17   }
18   render() {
19     if (this.state.usuarios.length > 0) {
20       return (
21         <div className="container">
22           <Table tablaDatos={this.state.usuarios} />
23         </div>
24       )
25     }
26     return(
27       <div style = {{align:'center'}}>
28         <img alt = "Por favor espere" src = "https://upload.wikimedia.org/wikipedia/commons/b/b1>Loading_icon.gif"/>
29       </div>
30     )
31   }
32 }
33
34 export default App;
```



## 7.2 - Agregando loader a nuestras peticiones

Prueba ejemplo src/App.js:

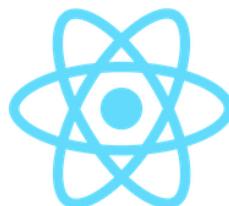
Se mostrará la siguiente imagen:



Antes de generar el listado:

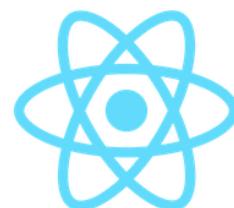
localhost:3001

Nombre	Id Usuario	Email
Leanne Graham	Bret	Sincere@april.biz
Ervin Howell	Antonette	Shanna@melissa.tv
Clementine Bauch	Samantha	Nathan@yesenia.net
Patricia Lebsack	Karianne	Julianne.OConner@kory.org
Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca
Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info
Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz
Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me
Glenna Reichert	Delphine	Chaim_McDermott@dana.io
Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz



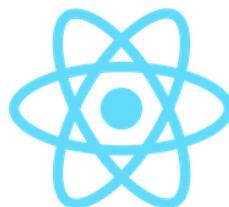
## 7.3 - Eliminando incosistencias en el proyecto

Una manera de reducir inconsistencias en el proyecto es reutilizar la lógica de un componente a lo largo de un proyecto y esto se logra concretamente con una función común y corriente que recibe como parámetro un componente y retorna un nuevo componente, cabe resaltar que esta técnica no hace parte del API de React, solo es un patrón utilizado para la reutilización de lógica. Esta técnica se conoce como componente de orden superior.



## 7.4 - Escribiendo un componente de orden superior

Un componente de orden superior (HOC por las siglas en inglés de higher-order component) es una técnica avanzada en React para el reuso de la lógica de componentes. Los HOCs no son parte de la API de React. Son un patrón que surge de la naturaleza compositacional de React.



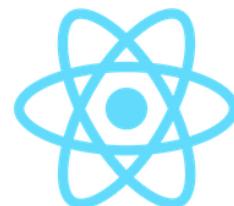
## 7.4 - Escribiendo un componente de orden superior

En concreto, un componente de orden superior es una función que recibe un componente y devuelve un nuevo componente.

```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

Mientras que un componente transforma props en interfaz de usuario, un componente de orden superior transforma un componente en otro.

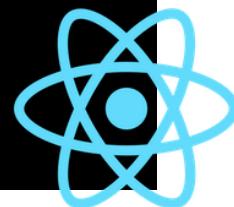
Los HOCs son comunes en bibliotecas de terceros usadas en React, tales como connect en Redux y createFragmentContainer en Relay.



## 7.4 - Escribiendo un componente de orden superior

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App24.js. Realice los siguiente cambios sobre src/App.js

```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   constructor(...props) {
5     super(...props)
6     this.state = { data: [] }
7   }
8   componentDidMount() {
9     fetch('https://jsonplaceholder.typicode.com/posts')
10    .then(response => response.json())
11    .then(data => {
12      this.setState({ data })
13    })
14    .catch(err => console.log(err.message))
15  }
16
17  render() {
18    const { data } = this.state
19    return (
20      <ul>
21        {data.map(post => <li key={post.id}>{post.title}</li> )}
22        </ul>
23    )
24  }
25}
26
27 export default App
```

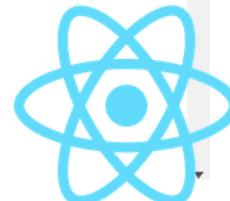


# 7.4 - Escribiendo un componente de orden superior

## Prueba src/App.js

localhost:3001

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem
- in quibusdam tempore odit est dolorem
- dolorum ut in voluptas mollitia et saepe quo animi
- voluptatem eligendi optio
- eveniet quod temporibus
- sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
- fugit voluptas sed molestias voluptatem provident
- voluptate et itaque vero tempora molestiae
- adipisci placeat illum aut reiciendis qui
- doloribus ad provident suscipit at
- asperiores ea ipsam voluptatibus modi minima quia sint
- dolor sint quo a velit explicabo quia nam
- maxime id vitae nihil numquam
- autem hic labore sunt dolores incident
- rem alias distinctio quo quis
- est et quae odit qui non
- quasi id et eos tenetur aut quo autem
- delectus ullam et corporis nulla voluptas sequi
- iusto eius quod necessitatibus culpa ea
- a quo magni similique preferendis
- ullam ut quidem id aut vel consequuntur

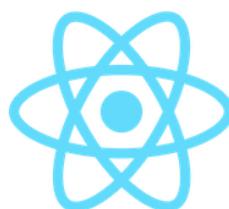


## 7.5 - Agregando lógica reutilizable al HOC

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App25.js.

Desarrolle el componente HOC src/extraerData.js

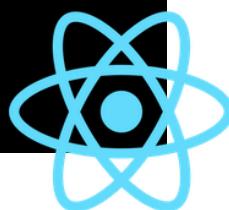
Realice los siguiente cambios sobre src/App.js



## 7.5 - Agregando lógica reutilizable al HOC

### src/extradata.js

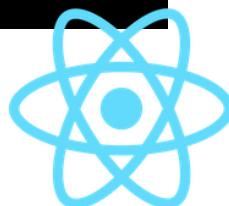
```
1 import React,{ Component } from 'react'
2
3 export default function extraerData(endpoint, WrappedComponent) {
4     return class extends Component {
5         constructor(props) {
6             super(props)
7             this.state = { data: [] }
8         }
9
10        componentDidMount() {
11            fetch(endpoint)
12                .then(response => response.json())
13                .then(data => {
14                    this.setState({ data })
15                })
16                .catch(err => console.log(err.message))
17        }
18
19        render() {
20            return <WrappedComponent data={this.state.data} {...this.props} />
21        }
22    }
23 }
```



## 7.5 - Agregando lógica reutilizable al HOC

src/App.js

```
1 import React,{ Component } from 'react';
2 import extraerData from './extraerData'
3
4 class App extends Component {
5   render() {
6     return (
7       <ul>
8         {this.props.data.map(post => <li key={post.id}>{post.title}</li> )}
9       </ul>
10    )
11  }
12 }
13
14 export default extraerData('https://jsonplaceholder.typicode.com/posts', App)
```

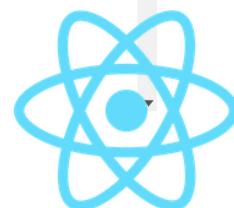


# 7.5 - Agregando lógica reutilizable al HOC

## Pruebas src/extraerData.js y src/App.js

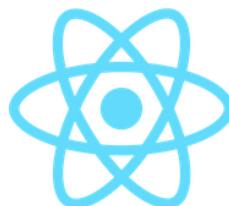
localhost:3001

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem
- in quibusdam tempore odit est dolorem
- dolorum ut in voluptas mollitia et saepe quo animi
- voluptatem eligendi optio
- eveniet quod temporibus
- sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
- fugit voluptas sed molestias voluptatem provident
- voluptate et itaque vero tempora molestiae
- adipisci placeat illum aut reiciendis qui
- doloribus ad provident suscipit at
- asperiores ea ipsam voluptatibus modi minima quia sint
- dolor sint quo a velit explicabo quia nam
- maxime id vitae nihil numquam
- autem hic labore sunt dolores incident
- rem alias distinctio quo quis
- est et quae odit qui non
- quasi id et eos tenetur aut quo autem
- delectus ullam et corporis nulla voluptas sequi
- iusto eius quod necessitatibus culpa ea
- a quo magni similique perferendis
- ullam ut quidem id aut vel consequuntur



# Hooks

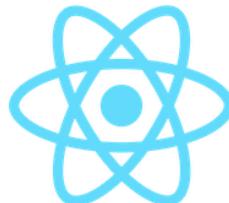
- 8.1 - Reescribiendo componentes para utilizar el estado
- 8.2 - Utilizando el hook useState
- 8.3 - Actualizando el estado con hooks
- 8.4 - Utilizando el hook useEffect
- 8.5 - Componentes con múltiples estados
- 8.6 - Escribir un custom hook
- 8.7 - Bonus: Custom hook useFetch
- 8.8 - Bonus: Custom hook useCounter



## 8.1 - Reescribiendo componentes para utilizar el estado

Los *Hooks* son una nueva incorporación en React 16.8. Le permiten usar estados y otras características de React sin escribir una clase.

Los hooks son una nueva forma de usar características propias de React como el estado o el ciclo de vida sin la necesidad de escribir una clase que es como se hacia hasta ahora, lo cual no significa que el equipo detrás de React piense dejar de usar clases en un futuro cercano ya que hay una gran multitud de proyectos que usan componentes creados de esta manera.

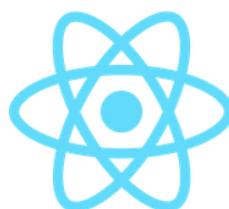


## 8.2 - Utilizando el hook useState

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App26.js.

Desarrolle el componente src/Puerta.js

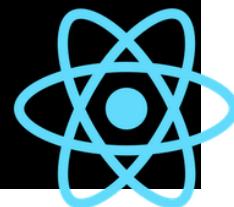
Realice los siguiente cambios sobre src/App.js



## 8.2 - Utilizando el hook useState

Contenido src/Puerta.js:

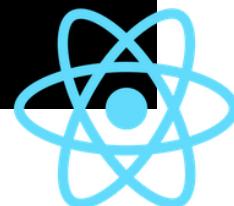
```
1 import React, { Component } from 'react'
2
3 class Puerta extends Component {
4     constructor(...props) {
5         super(...props)
6         this.state = {open: false}
7     }
8
9     render() {
10        const { open } = this.state
11        return (
12            <div className="alert alert-dismissible alert-primary">
13                <h1>La puerta esta {open ? 'abierta' : 'cerrada'}</h1>
14                <button className="btn btn-primary"
15                    onClick={() => this.setState({ open: !this.state.open })}>
16                    {open ? 'Cerrar' : 'Abrir'}
17                </button>
18            </div>
19        )
20    }
21 }
22
23 export default Puerta
```



## 8.2 - Utilizando el hook useState

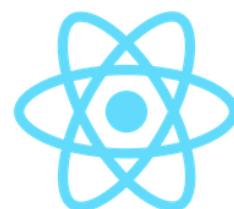
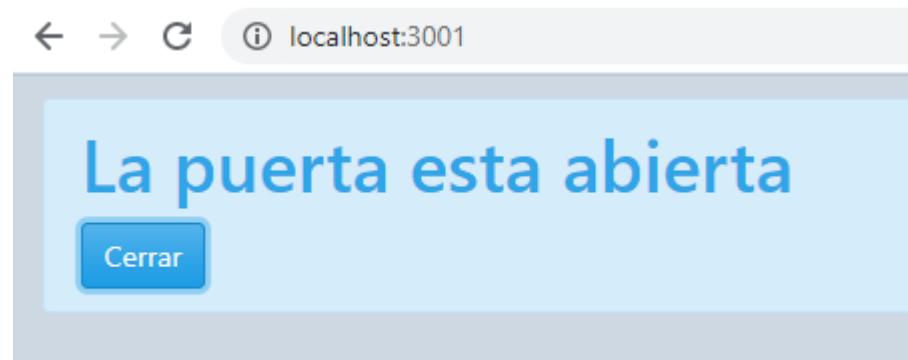
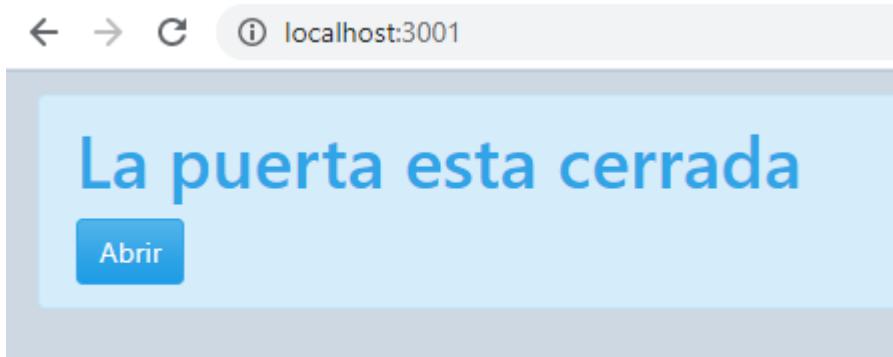
Contenido src/App.js

```
1 import React,{ Component } from 'react';
2 import Puerta from './Puerta'
3
4 class App extends Component {
5   render() {
6     return (
7       <div className = "alert alert-dismissible alert-info">
8         <Puerta />
9       </div>
10    )
11  }
12}
13
14 export default App;
```



## 8.2 - Utilizando el hook useState

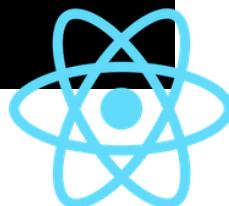
Prueba src/App.js y src/Puerta.js



## 8.2 - Utilizando el hook useState

Ahora proceda a guardar src/Puerta.js como src/Pruerta1.js y realice los siguientes cambios:

```
1 import React, { useState } from 'react'
2
3 const Puerta = () => {
4     const [open, setOpen] = useState(false)
5
6     return (
7         <div>
8             <h1>La puerta esta {open ? 'abierta' : 'cerrada'}</h1>
9             <button onClick={() => setOpen(!open)}>
10                 {open ? 'Cerrar' : 'Abrir'}
11             </button>
12         </div>
13     )
14 }
15
16 export default Puerta
```



## 8.3 - Actualizando el estado con hooks

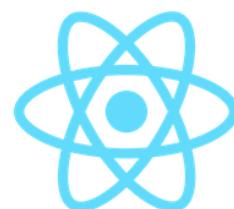
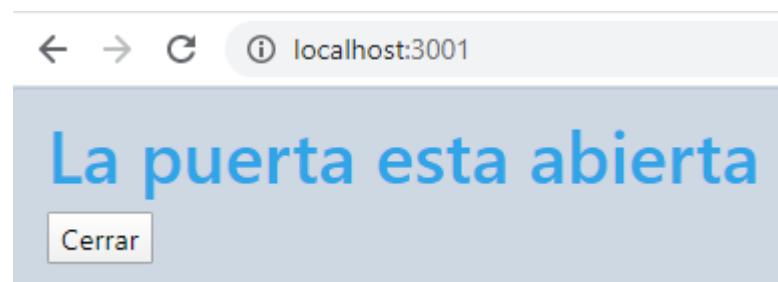
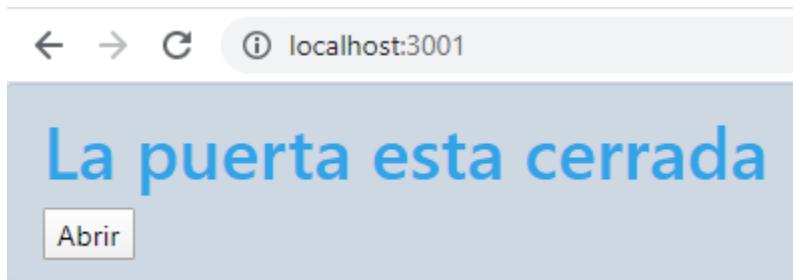
Análisis src/Puerta.js:

En este código primero que todo estamos usando el hook `useState()` el cual nos permite usar la característica de estado en este componente a pesar de no ser una clase, este hook nos recibe como parámetro el valor por defecto que queremos que tenga nuestro estado y nos devuelve un array con dos posiciones, la primera posición es el valor que le asignaremos por defecto en este caso `false`, y como segunda posición nos devuelve la función que usaremos para actualizar este valor en nuestro componente, después retornamos nuestro `jsx` y aqui como se hizo anteriormente usamos un botón para actualizar el valor de `open` y dependiendo del valor que vaya tomando se renderiza que la puerta esta abierta o cerrada.



## 8.3 - Actualizando el estado con hooks

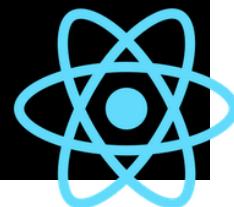
Prueba src/Puerta.js:



## 8.4 - Utilizando el hook useEffect

Desarrollar el programa src/ListaData.js:

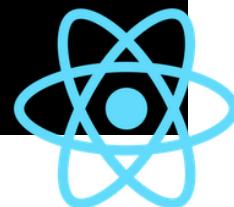
```
1 import React, { Component } from 'react';
2
3 class ListaData extends Component {
4   constructor(...props) {
5     super(...props)
6
7     this.state = { posts: [] }
8   }
9
10  componentDidMount() {
11    this.getPosts()
12  }
13
14  getPosts() {
15    fetch('https://jsonplaceholder.typicode.com/posts')
16      .then(response => response.json())
17      .then(posts => this.setState({ posts }))
18      .catch(err => console.log(err.message))
19  }
20
21  render() {
22    const { posts } = this.state
23    return (
24      <ul>
25        {posts.map(post => <li key={post.id}>{post.title}</li>)}
26      </ul>
27    );
28  }
29}
30
31 export default ListaData;
```



## 8.4 - Utilizando el hook useEffect

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App27.js. Desarrolle el siguiente contenido:

```
1 import React,{ Component } from 'react';
2 import ListaData from './ListaData'
3
4 class App extends Component {
5   render() {
6     return (
7       <div>
8         <ListaData />
9       </div>
10    )
11  }
12}
13
14 export default App;
```

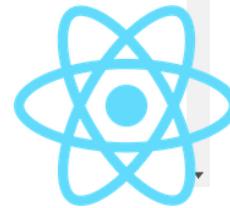


## 8.4 - Utilizando el hook useEffect

### Prueba:

localhost:3001

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem
- in quibusdam tempore odit est dolorem
- dolorum ut in voluptas mollitia et saepe quo animi
- voluptatem eligendi optio
- eveniet quod temporibus
- sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
- fugit voluptas sed molestias voluptatem provident
- voluptate et itaque vero tempora molestiae
- adipisci placeat illum aut reiciendis qui
- doloribus ad provident suscipit at
- asperiores ea ipsam voluptatibus modi minima quia sint
- dolor sint quo a velit explicabo quia nam
- maxime id vitae nihil numquam
- autem hic labore sunt dolores incident
- rem alias distinctio quo quis
- est et quae odit qui non
- quasi id et eos tenetur aut quo autem
- delectus ullam et corporis nulla voluptas sequi
- iusto eius quod necessitatibus culpa ea
- a quo magni similique preferendis
- ullam ut quidem id aut vel consequuntur

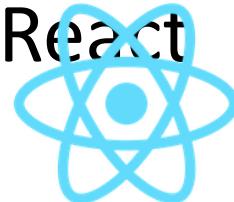


## 8.4 - Utilizando el hook useEffect

Análisis:

En este sencillo ejemplo solo estamos declarando un estado posts como un array, después en el componentDidMount() que es un método del ciclo de vida que se ejecuta cuando el componente se renderizo, estamos haciendo una petición a la api de jsonplaceholder, para traernos una lista de posts usando el método getPosts() que nosotros mismos declaramos, actualizamos el estado de los posts y los renderizamos como una lista en el método render().

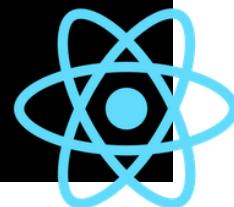
Ahora hagamos el mismo ejemplo con los React hooks



## 8.4 - Utilizando el hook useEffect

Guarde src/ListaData.js como src/ListaData1.js y realice las siguientes actualizaciones sobre src/ListaData.js:

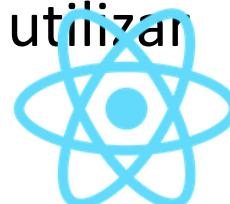
```
1 import React, { useEffect, useState } from 'react';
2
3 const ListaData = () => {
4     const [posts, setPosts] = useState( [] )
5
6     function getPosts() {
7         fetch('https://jsonplaceholder.typicode.com/posts')
8             .then(response => response.json())
9             .then(posts => setPosts(posts))
10            .catch(err => console.log(err.message))
11    }
12
13    useEffect(() => {
14        getPosts()
15    }, [])
16
17    return (
18        <ul>
19            {posts.map(post => <li key={post.id}>{post.title}</li>)}
20        </ul>
21    )
22}
23
24 export default ListaData;
```



## 8.4 - Utilizando el hook useEffect

Análisis src/ListaData.js:

- En este ejemplo primero que todo estamos definiendo un nuevo estado con useState(), el cual será por defecto un array vacío, después estamos definiendo la función getPosts() que será la encargada de hacer la petición a la API y usando el **hook useEffect()** es que ejecutamos la función que nos traerá los posts, **este hook useEffect()** nos permitirá reemplazar los métodos del ciclo de vida **componentDidMount()**, **componentDidUpdate()** y **componentWillUnmount()**, en este ejemplo estamos usando el **useEffect()** como si se tratara del **componentDidMount()**, esto debido al array vacío que le estamos pasando como segundo parámetro, con esto le estamos indicando que solo se va a ejecutar en una ocasión, después simplemente renderizamos estos nuevos artículos cuando retornemos.
- Como pueden ver usar los React hooks nos permiten tener componentes muchos más limpios, sin la necesidad de utilizar una clase.

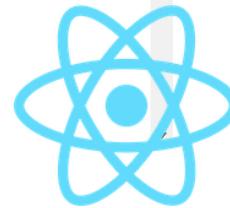


# 8.4 - Utilizando el hook useEffect

## Prueba:

localhost:3001

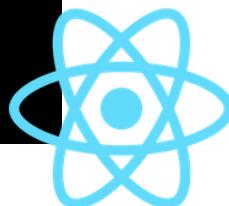
- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem
- in quibusdam tempore odit est dolorem
- dolorum ut in voluptas mollitia et saepe quo animi
- voluptatem eligendi optio
- eveniet quod temporibus
- sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
- fugit voluptas sed molestias voluptatem provident
- voluptate et itaque vero tempora molestiae
- adipisci placeat illum aut reiciendis qui
- doloribus ad provident suscipit at
- asperiores ea ipsam voluptatibus modi minima quia sint
- dolor sint quo a velit explicabo quia nam
- maxime id vitae nihil numquam
- autem hic labore sunt dolores incident
- rem alias distinctio quo quis
- est et quae odit qui non
- quasi id et eos tenetur aut quo autem
- delectus ullam et corporis nulla voluptas sequi
- iusto eius quod necessitatibus culpa ea
- a quo magni similique preferendis
- ullam ut quidem id aut vel consequuntur



## 8.6 - Escribir un custom hook

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App28.js. Desarrolle el siguiente contenido en src/App.js :

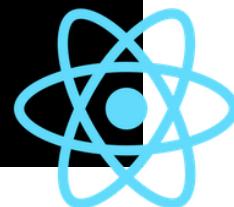
```
1 import React,{ Component } from 'react';
2 import DobleListaData from './DobleListaData'
3
4 class App extends Component {
5   render() {
6     return (
7       <div>
8         <DobleListaData />
9       </div>
10    )
11  }
12}
13
14 export default App;
```



## 8.6 - Escribir un custom hook

Ahora debe crear el archivo src/DobleListaData.js , con el siguiente contenido:

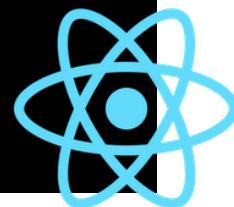
```
1 import React, { useEffect, useState, Fragment } from 'react';
2
3 const DobleListaData = () => {
4   const [posts, setPosts] = useState( [] )
5   const [users, setUsers] = useState( [] )
6
7   function getPosts() {
8     fetch('https://jsonplaceholder.typicode.com/posts')
9       .then(response => response.json())
10      .then(posts => setPosts(posts))
11      .catch(err => console.log(err.message))
12   }
13
14   function getUsers() {
15     fetch('https://jsonplaceholder.typicode.com/users')
16       .then(response => response.json())
17      .then(users => setUsers(users))
18      .catch(err => console.log(err.message))
19 }
```



## 8.6 - Escribir un custom hook

Continuación archivo src/DobleListaData.js:

```
20
21     useEffect(() => {
22         getPosts()
23     }, [])
24
25     useEffect(() => {
26         getUsers()
27     }, [])
28
29     return (
30         <Fragment>
31             <h2>Posts</h2>
32             <ul>
33                 {posts.map(post => <li key={post.id}>{post.title}</li>)}
34             </ul>
35             <h2>Usuarios</h2>
36             <ul>
37                 {users.map(user => <li key={user.id}>{user.name}</li>)}
38             </ul>
39         </Fragment>
40     )
41 }
42 export default DobleListaData;
```



## 8.6 - Escribir un custom hook

Prueba src/DobleListaData.js:

← → ⌂ ⓘ localhost:3001

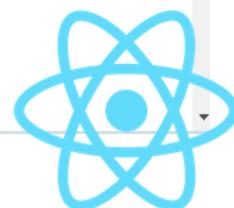


### Usuarios

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

### Posts

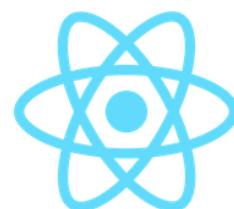
- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem



## 8.6 - Escribir un custom hook

Análisis src/DobleListaData.js:

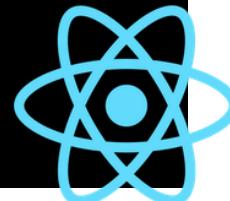
El problema con este código es que estamos haciendo dos funciones prácticamente idénticas para traer los usuarios y los artículos, en este tipo de casos es que nos sirven los custom hooks, el cual podemos hacerlo de la siguiente manera:



## 8.6 - Escribir un custom hook

Se creará el archivo TraerDatos.js, con el siguiente contenido:

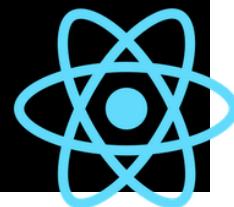
```
1 import { useState, useEffect } from 'react'
2
3 ▼ function TraerDatos(url, defaultValue) {
4     const [data, setData] = useState(defaultValue)
5     const [error, setError] = useState(null)
6
7 ▼     useEffect(() => {
8         fetch(url)
9             .then(response => response.json())
10            .then(data => {
11                setData(data)
12            })
13            .catch(err => {
14                setError(err)
15            })
16        }, [])
17
18        return { data, error }
19    }
20
21 export default TraerDatos|
```



## 8.6 - Escribir un custom hook

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App29.js. Desarrolle el siguiente contenido en src/App.js :

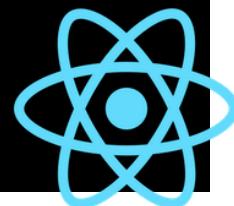
```
1 import React, { useEffect, useState } from 'react'
2 import TraerDatos from './TraerDatos'
3
4 const App = () => {
5   const users = TraerDatos('https://jsonplaceholder.typicode.com/users', [])
6   const posts = TraerDatos('https://jsonplaceholder.typicode.com/posts', [])
7
8   return (
9     <div>
10       <h2>Usuarios</h2>
11       <ul>
12         {users.data.map(user => (
13           <li key={user.id}>{user.name}</li>
14         ))}
15       </ul>
16       <h2>Posts</h2>
17       <ul>
18         {posts.data.map(posts => (
19           <li key={posts.id}>{posts.title}</li>
20         ))}
21       </ul>
22     </div>
23   )
24 }
```



## 8.6 - Escribir un custom hook

Siguiendo el procedimiento conocido, guarde el archivo: src/App.js, como src/App29.js. Desarrolle el siguiente contenido en src/App.js :

```
1 import React, { useEffect, useState } from 'react'
2 import TraerDatos from './TraerDatos'
3
4 const App = () => {
5   const users = TraerDatos('https://jsonplaceholder.typicode.com/users', [])
6   const posts = TraerDatos('https://jsonplaceholder.typicode.com/posts', [])
7
8   return (
9     <div>
10       <h2>Usuarios</h2>
11       <ul>
12         {users.data.map(user => (
13           <li key={user.id}>{user.name}</li>
14         ))}
15       </ul>
16       <h2>Posts</h2>
17       <ul>
18         {posts.data.map(posts => (
19           <li key={posts.id}>{posts.title}</li>
20         ))}
21       </ul>
22     </div>
23   )
24 }
```



# 8.6 - Escribir un custom hook

## Prueba src/App.js :

localhost:3001



### Usuarios

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

### Posts

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam
- nesciunt iure omnis dolorem tempora et accusantium
- optio molestias id quia eum
- et ea vero quia laudantium autem

