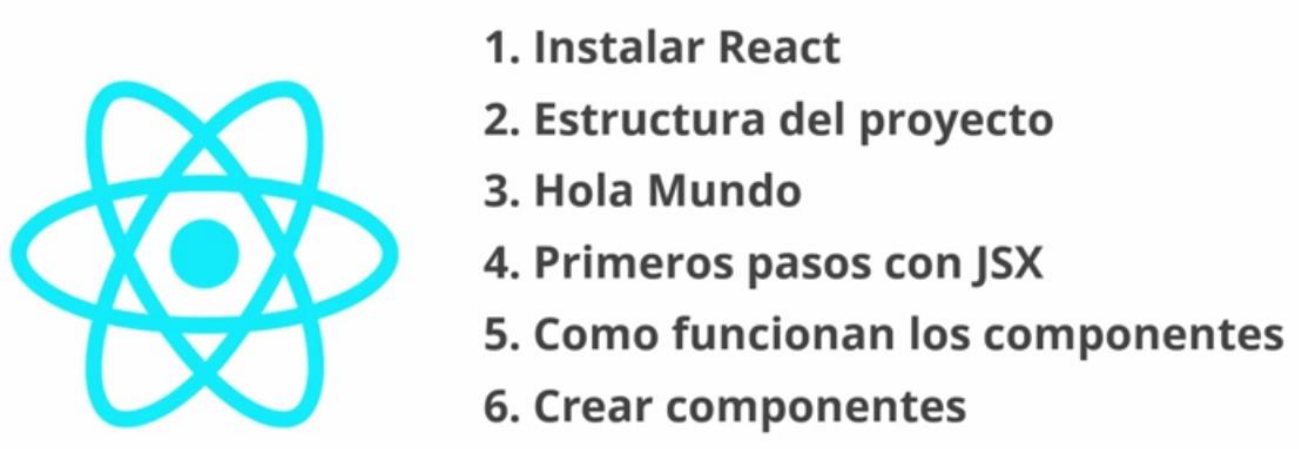
React

React es una biblioteca escrita en JavaScript, desarrollada en Facebook para facilitar la creación de componentes interactivos, reutilizables, para interfaces de usuario. Se utiliza en Facebook para la producción de componentes, e Instagram está escrito enteramente en **React**.



1- Instalar React

a- Instalar Node.js

https://nodejs.org/es/

b- Abrir la consola de Windows, Linux o Mac y ejecutar el comando

c- Ejecutar el comando

```
npm install -g npm@latest
```

para actualizar el gestor de paquetes de node npm y de esa forma posteriormente poder instalar la última versión de React

d- Limpiamos la cache con el comando
`npm cache clean --force`

e- Instalar paquete
`npm install -g create-react-app`
El cual contiene un intérprete de consola, web pack y live reload entre otras ventajas para el desarrollo en React

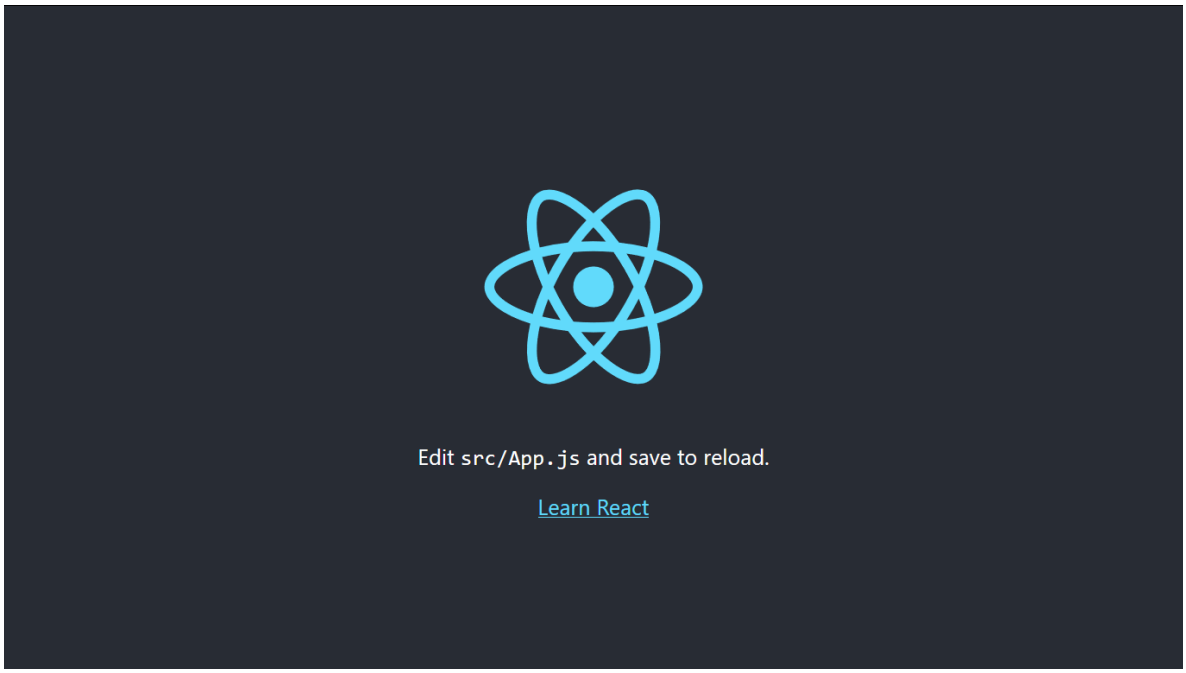
f- Ir mediante la consola a la carpeta donde tenemos nuestros proyectos de React Ejemplo: d:/ProyectosReact

g- Ejecutar el comando
`create-react-app NOMBRE_DEL_PROYECTO`
Ejemplo: `create-react-app primer-app-react` (todo en minusculas)
Si les tira error ejecuten el comando
`npm cache clean --force` y vuelvan a intentar la creación
si el comando finaliza con éxito por consola debemos ver

```
removed 1 package and audited 931401 packages in 10.214s

58 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


Created git commit.

Success! Created primerappreact at D:\ProyectosReact\primerappreact
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

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

  npm test
    Starts the test runner.

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

We suggest that you begin by typing:

  cd primerappreact
  npm start

Happy hacking!

D:\ProyectosReact>
```

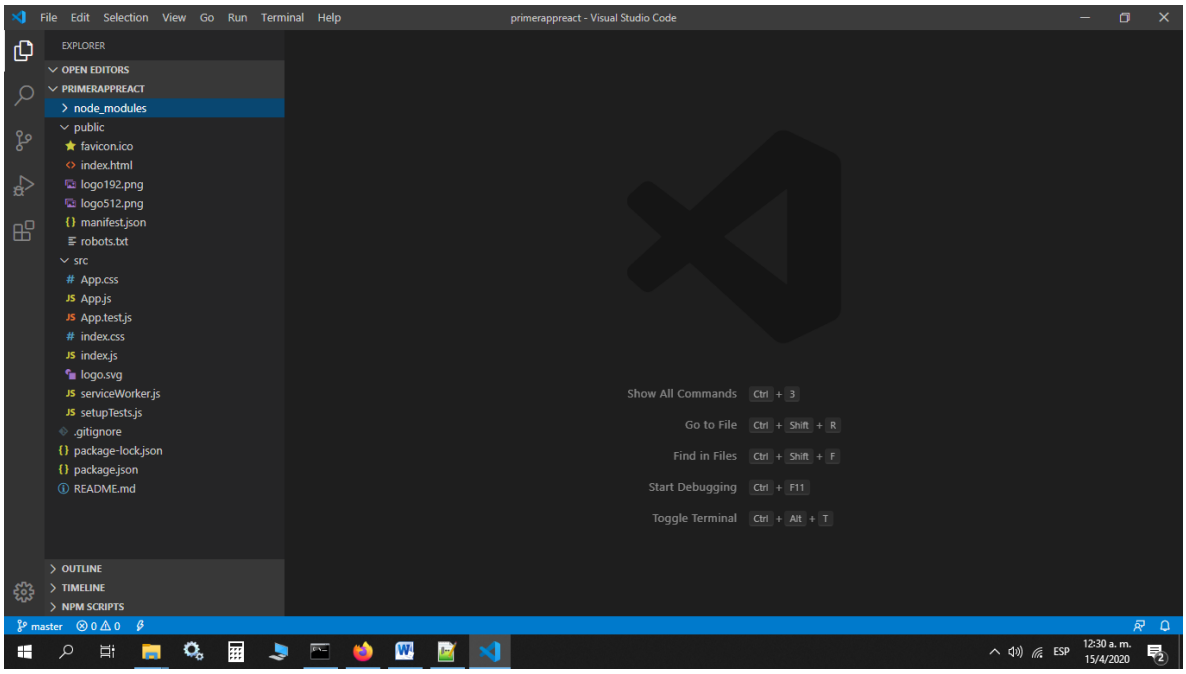
h- Hemos finalizado la instalación de React y la creación de nuestra primera aplicación

i- Para ejecutar el proyecto nos paramos dentro del proyecto
`cd primerappreact`
y ejecutamos el comando
`npm start`



2- Estructura del Proyecto

a- Abrimos el proyecto con **Visual Studio Code**



Archivos Importantes:

package.json archivo de configuración del proyecto, versiones, dependencias, paquetes, etc.

.gitignore nos permite agregar los archivos y directorios que deseamos que no sean tenidos en cuenta por git

Carpeta src->index.js archivo de inicio de nuestra aplicación.

Carpeta src->index.css hoja de estilo general de la aplicación.

Carpeta src->App.js es un componente de React. El componente inicial de nuestra Aplicación.

Carpeta src->App.css hoja de estilo para el componenteApp.js.

Carpeta public->index.html archivo html inicial de la aplicación dentro del

```
<div id="root"></div>
```

Se carga toda la aplicación.

3- HOLA MUNDO

Modificamos App.js

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        Hola Mundo
      </header>
    </div>
  );
}
export default App;
```

4- Primeros pasos con JSX

Trabajaremos con JSX lenguaje que mezcla HTML y JavaScript y nos permite poder trabajar con React de una forma amigable, si no lo usamos se torna muy complicado el desarrollo con React.

Modificamos nuestro archivo App.js

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function mensajeHolaMundo(mensaje){
  return mensaje.toUpperCase();
```

```
}

function App() {
  var mensaje = "hola mundo";
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        {mensajeHolaMundo(mensaje)}
      </header>
    </div>
  );
}

export default App;
```

como puede verse JSX vincula y genera el código resultante de la mezcla de HTML con JavaScript.

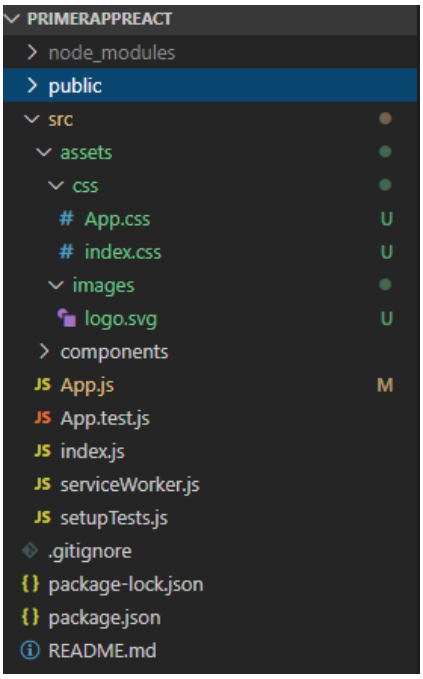
5- Cómo funcionan los componentes

Que es un componente? Simplemente es una sección de la interfaz de usuario que podemos reutilizar en N interfaces diferentes. Puede ser un botón, una grilla, una caja de texto personalizada, un template, etc. Sirve como un mediador recibiendo datos por ejemplo desde el backend y luego los emite hacia la vista y viceversa.

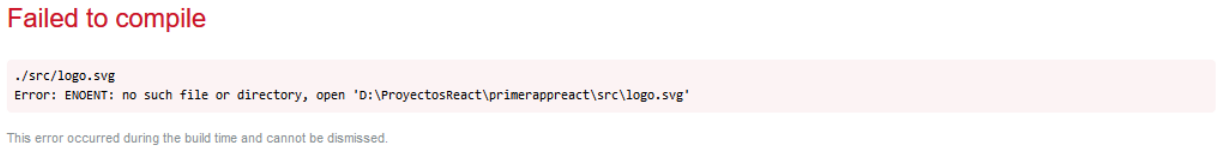
6- Creando componentes

Vamos a crear dentro de src una carpeta components donde iremos creando nuestros nuevos componentes, además vamos a crear una carpeta assets también dentro de src y dentro de esta una carpeta css para almacenar nuestras hojas de estilo y otra carpeta images para las imágenes.

Movemos los archivos css a la carpeta css y las imágenes a la carpeta imágenes el proyecto nos debe quedar similar a:



Lógicamente al mover los archivos la aplicación va a fallar



Debemos simplemente corregir las rutas que corresponden, y de esta forma tenemos un proyecto más ordenado y limpio para trabajar.

Dentro de la carpeta components vamos a crear nuestro primer componente con nombre MiComponente.js

La estructura básica de cualquier componente es la siguiente:

```
import React from 'react';

class MiComponente extends React.Component{

}
```

Siempre debo importar el módulo de React, declarar la clase y hacer que la clase herede de React.Component

Dentro de la clase vamos a tener un método render() que será el encargado de mostrar información la vista del componente (HTML y JavaScript) por pantalla.

Ejemplo

```
 render(){
        return (
            <h1>Titulo Aplicando Etiqueta H1</h1>
        );
```

Finalmente si deseo usar el componente en mi aplicación debo exportarlo

```
export default MiComponente;
```

El componente terminado quedaría:

```
import React from 'react';

class MiComponente extends React.Component{

    render(){
        return (
            <h1>Titulo Aplicando Etiqueta H1</h1>
        );

    }
}

export default MiComponente;
```

Si quiero usar este componente por ejemplo dentro de App.js debo en primer lugar importarlo

```
import MiComponente from './components/MiComponente';
```

y luego simplemente donde quiero usarlo declaro el componente como si fuese una etiqueta de HTML

```
 <MiComponente></MiComponente>
```

Un detalle de los componentes con JSX es que si deseo retornar un componente compuesto por multiples líneas siempre el componente debe estar contenido dentro de una única etiqueta raíz por ejemplo:

```
<div>

/*Multiples Lineas*/

</div>
```

O haciendo uso de un Fragmento de React (React.Fragment) por ejemplo:

```
     <React.Fragment>
          <h1>Titulo Aplicando Etiqueta H1</h1>
          <h2>Titulo Aplicando Etiqueta H2</h2>
     </React.Fragment>
```

Propiedades en Componentes React

Para terminar haremos uso del objeto this.props el cual contiene todas las propiedades definidas como atributos.

Ejemplo

```
<React.Fragment>
     <h1>{this.props.textoH1}</h1>
     <h2>{this.props.textoH2}</h2>
</React.Fragment>
```

Y para usarlo simplemente

```
<MiComponente textoH1="Propiedad textoH1" textoH2="Propiedad textoH2"></MiComponente>
```

Resultado:

Para siguiente aplicación deben instalar:

```
npm install --save react-router-dom
npm install react-bootstrap bootstrap
```