

Ejercicios sesiones 1, 2 y 3

Para este primer reto de programación en React vamos a poner en practica lo aprendido hasta ahora para crear dos componentes.

Cada uno de estos componentes tendrá sus propias características.

Componente A: Este componente va a tener disponer un Contacto(crea una clase para ello), que va a contar con las siguientes características:

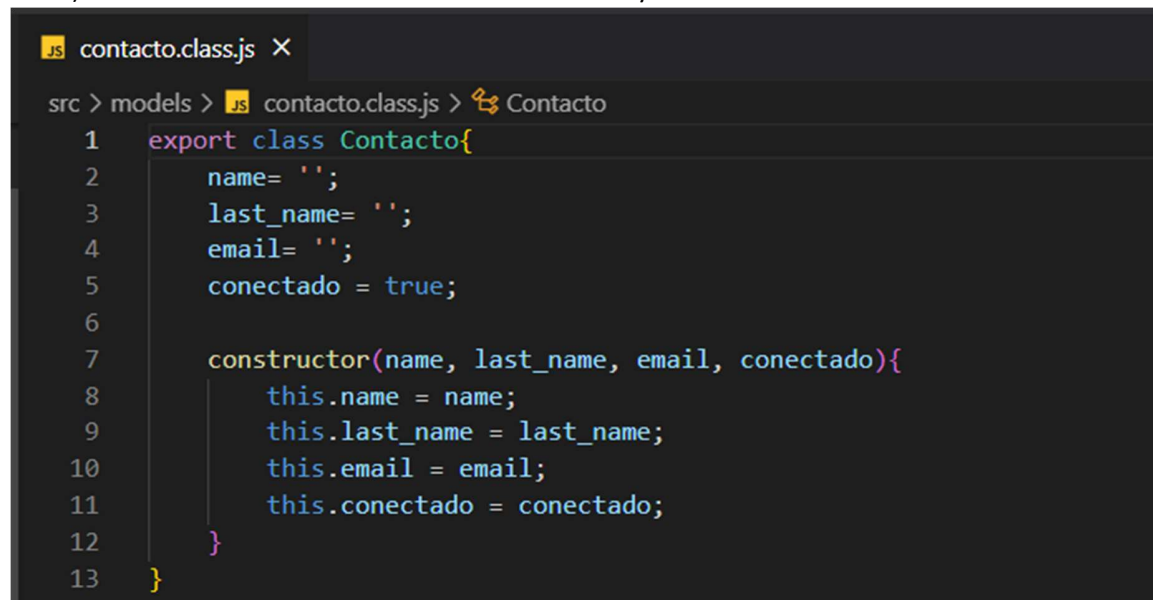
Nombre: que será un String.

Apellido: también un String.

Email: de nuevo un String.

Conectado: será un booleano que nos indicará si la persona está conectada o no.

a) Genero contacto en una clase con un modelo y su constructor



```
.JS contacto.class.js X
src > models > .JS contacto.class.js > Contacto
1  export class Contacto{
2      name= '';
3      last_name= '';
4      email= '';
5      conectado = true;
6
7      constructor(name, last_name, email, conectado){
8          this.name = name;
9          this.last_name = last_name;
10         this.email = email;
11         this.conectado = conectado;
12     }
13 }
```

Genero Componente A como un componente puro con los atributos definidos en el modelo

```
componenteA.jsx X
src > components > pure > componenteA.jsx > ...
1 import React from 'react';
2 import PropTypes from 'prop-types'
3 import {Contacto} from '../../models/contacto.class'
4
5 export const ComponenteA = ({contacto}) => {
6   return (
7     <div>
8       <h2>
9         Nombre: {contacto.name}
10      </h2>
11      <h2>
12        Apellido: {contacto.last_name}
13      </h2>
14      <h2>
15        Email: {contacto.email}
16      </h2>
17      <h3>
18        {contacto.name} is: { contacto.conectado? 'CONECTADO':'NO_CONECTADO'}
19      </h3>
20    </div>
21  );
22 };
23
24 ComponenteA.propTypes = {
25   contacto: PropTypes.instanceOf(Contacto)
26 };
```

Genero un componente para crear un nuevo contacto que será renderizado

```
componenteA_list.jsx X JS App.js
src > components > container > componenteA_list.jsx > ...
1 import React from 'react';
2 import {Contacto} from '../../models/contacto.class';
3 import {ComponenteA} from '../pure/componenteA';
4
5 const ComponenteAList = () => {
6   const componenteA = new Contacto('Pepe','Juarez', 'pj@gmail', true)
7   return (
8     <div>
9       <div>
10        <h1>Contactos:</h1>
11      </div>
12      <ComponenteA contacto={componenteA}></ComponenteA>
13    </div>
14  );
15 };
16 export default ComponenteAList;
17
```

Importo el componente a renderizar en App.js

```
JS App.js ×
src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3 import ComponenteAList from '../src/components/container/componenteA_list';
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        <ComponenteAList></ComponenteAList>
11      </header>
12    </div>
13  );
14 }
15
16 export default App;
```

You can now view **rejericiosiones123** in the browser.

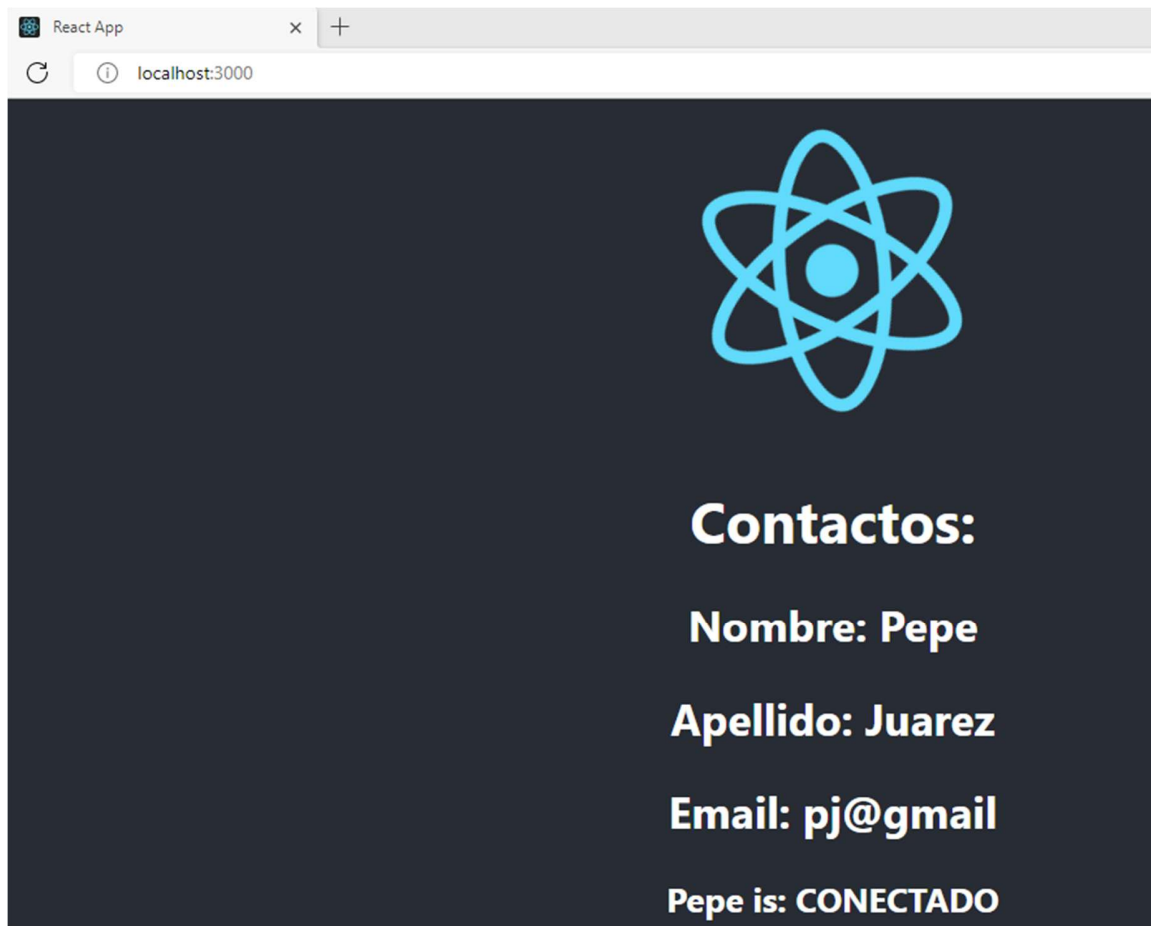
Local: http://localhost:3000

On Your Network: http://192.168.0.8:3000

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

webpack compiled **successfully**





Componente B: Por otro lado, tenemos el componente B que va a recibir por props un contacto y va a poder cambiar su estado de conectado a desconectado y viceversa.

Si el contacto está conectado, se debe mostrar: Contacto En Línea

Si el contacto no está conectado, se debe mostrar: Contacto No Disponible

Renderizado de componentes en la solución:

El Componente A debe ser renderizado dentro del componente App.js del proyecto. **OK**

El Componente B debe ser renderizado desde el componente A y recibir los props adecuadamente.

NO PUDE HACER

Recordatorio: Haz uso de PropTypes adecuadamente