Certified Tech Developer
The Ultimate Degree

DigitalHouse>

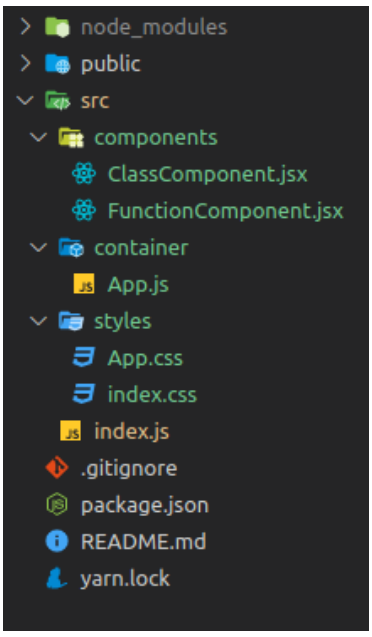
Front End III

Clase 6: Práctica integradora

Consigna

Es hora de repasar lo visto esta semana. Vamos a crear un proyecto de React desde cero. Para hacerlo, te recomendamos seguir los pasos a continuación e insertar las líneas de código donde corresponda. No te preocupes si no entendés algo, la idea es ver cómo normalmente se estructura un proyecto y las ventajas que tiene React.
LET'S GO!

El resultado final deberá presentar una estructura de proyecto así:



Y que tu página se vea de la siguiente manera:

Invitados:

- Nicolás está invitado a la fiesta
- Iván no está invitado a la fiesta
- Carolina está invitada a la fiesta

Tareas:

- Nicolás traerá papas fritas
- Iván traerá pizzas
- Carolina traerá bebidas

Ahora sí, ¡a trabajar!

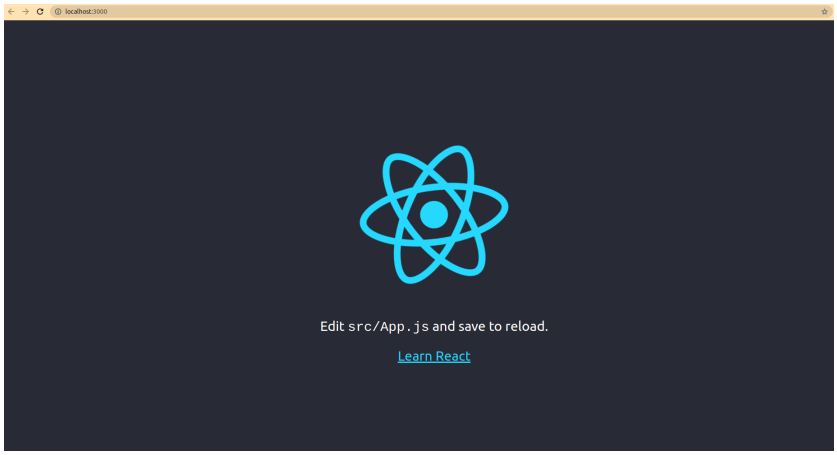
Paso I

Desde la terminal, deberás utilizar el comando para crear un proyecto. ¿Te acordás cómo se escribía?

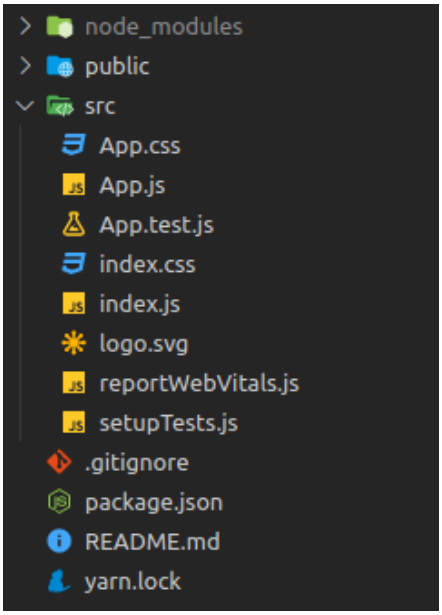
SOLUCIÓN ---> *Clic aquí*

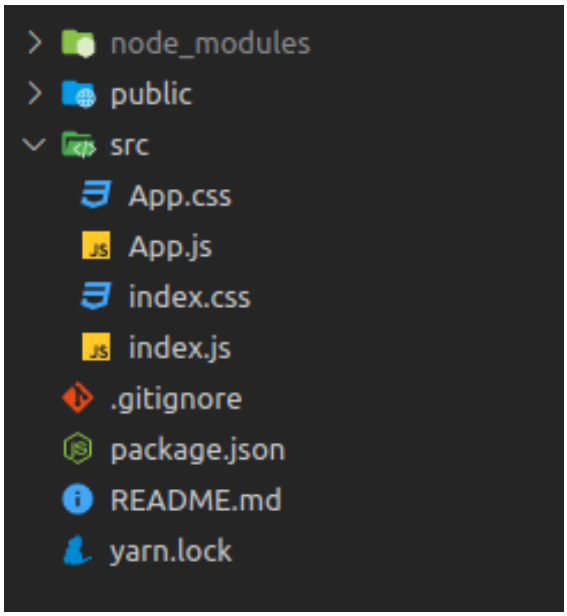
Paso II

Una vez creado, accedés al mismo y escribís npm start. De esta manera se te abrirá tu explorador con la siguiente vista:



En el proyecto, verás muchas carpetas y archivos:





Vas a necesitar hacer un poco de limpieza para que quede de la siguiente manera:

ELIMINAR LOS SIGUIENTES ARCHIVOS

- App.test.js
- logo.svg
- reportWebVitals.js
- setupTests.js

Paso III

Seguramente te diste cuenta de que el proyecto te está dando un error. Revisá los archivos index.js para eliminar lo que ya no existe. Te brindaremos un código para el archivo App.js.

PISTA ;) → index.js deben eliminar una función y un import.

```
import './App.css';


function App() {
 return (
   <div className="App">
     <p>Hello React</p>
   </div>
 );
}


export default App;
```

Vas a ver en el navegador el texto “Hello React”.

Paso IV

Ahora pasamos a la parte estructural del proyecto. Lo mejor siempre es tener un estilo de estructura para guardar los archivos. Esto siempre dependerá del equipo de trabajo donde te encuentres.

Para el día de hoy, vas a utilizar tres carpetas que se llamarán “components”, “container” y “styles” .

- En la carpeta “container” vas a colocar el archivo App.js.
- En la carpeta “components” crearás dos archivos vacíos llamados “ClassComponent.jsx” y “FunctionComponent.jsx”.
- En la carpeta “styles” moverás index.css, App.css y crearás un archivo nuevo llamado Component.css.

Paso V

Nuevamente, el proyecto nos indicará un error relacionado con que no encuentra los archivos. En este punto, deberás arreglar el import en App.js e index.js. Recordá que con dos puntos (../) salís de la carpeta y volvés a la anterior, mientras que con un punto (./) buscás archivos dentro de la carpeta.

Paso VI

En esta parte, el código debería estar funcionando sin problemas, caso contrario seguir verificando las rutas de los imports. Ahora, agregá código a tus dos componentes. Copiá el siguiente código:

ClassComponent.jsx

```
import React, { Component } from 'react'

export default class ClassComponent extends Component {

   render() {

       return (

           <li>TuNombre está invitado a la fiesta</li>

       )

   }

}
```

FunctionComponent.jsx

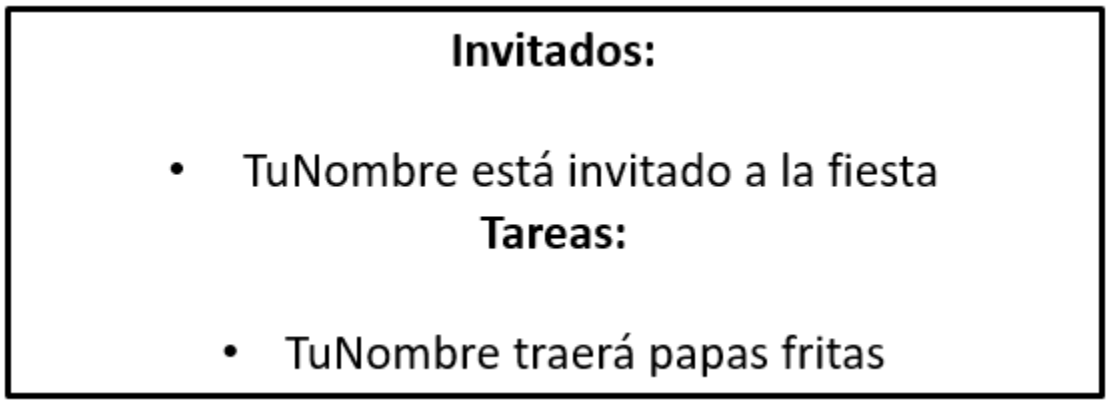
```
import React from 'react'



const functionComponent = () => {

   return (

       <li>tuNombre traerá papas fritas</li>

   )

}



export default functionComponent
```

Paso VII

El proyecto debería seguir funcionando. Vas a importar estos dos componentes al archivo App.js. Copiá estas líneas de código luego del return. Si comprobás, el proyecto debería fallar, ya que te están faltando los import. Mirá la manera en que está en index.js para ver cómo debería importarse.

```
<div className="App">

    <h3>Invitados:</h3>

    <ClassComponent />

    <h3>Tareas: </h3>

    <FunctionComponent />

  </div>
```

Si realizaste bien los imports, deberás ver lo siguiente en el navegador:



Paso VIII

Ya estás empezando a darle un sentido al uso de React y en esta parte la vas a explotar al máximo. Vas a hacer dos pequeñas cosas para poder reutilizar estos componentes: enviar props a los componentes en App.js y utilizarlos en tu Components files.

```
<div className="App">

    <h3>Invitados:</h3>

    <ClassComponent nombre="Nicolas" estaEnLista={true} />

    <ClassComponent nombre="Ivan" estaEnLista={false} />

    <ClassComponent nombre="Carolina" estaEnLista={true} />

    <h3>Tareas: </h3>

    <FunctionComponent nombre="Nicolas" tarea="papas fritas" />

    <FunctionComponent nombre="Ivan" tarea="pizzas" />

    <FunctionComponent nombre="Carolina" tarea="bebidas" />

  </div>
```

Ahora vamos a modificar nuestros componentes para que utilicen estas props.

ClassComponent.jsx (línea 6, luego del return)

```
                <li>{this.props.nombre} {this.props.estaEnLista ? "esta"
: "no está"} invitado a la fiesta</li>
```

FunctionComponent.jsx

```
import React from 'react'


const FunctionComponent = (props) => {
   return (
       <li>{props.nombre} traerá {props.tarea}</li>
   )
}


export default FunctionComponent
```

Paso IX

Si todo salió bien, tu página debería mostrar lo siguiente:

Invitados:

- Nicolás está invitado a la fiesta
- Iván no está invitado a la fiesta
- Carolina está invitada a la fiesta

Tareas:

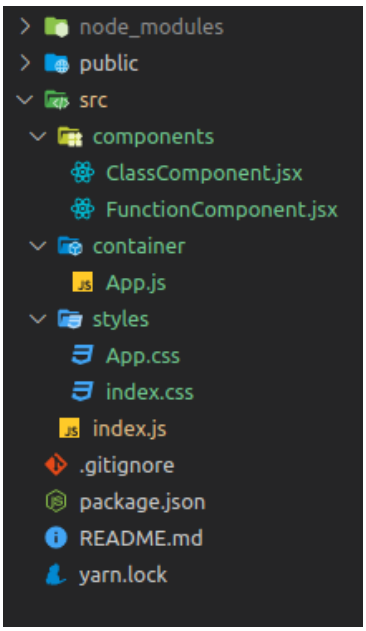
- Nicolás traerá papas fritas
- Iván traerá pizzas
- Carolina traerá bebidas

¡HEMOS REUTILIZADO CÓDIGO!

Imaginá proyectos con miles de líneas de código donde algunas quizás hacen lo mismo… Es ahí donde React nos ayuda, además de muchas otras cosas que iremos viendo en las siguientes clases.

Paso X

Para finalizar, la estructura del proyecto te debería quedar de esta manera.



EXTRA → Te invitamos a refactorizar ciertas líneas de código y realizar algo llamado *desestructuración.*

También te dejamos la resolución del ejercicio de la clase anterior para que puedas revisarlo y compararlo con tu resolución. ¡Contanos cómo te fue!

Chequeá el link:
https://github.com/lvanszs/dh-frontend3-clases/tree/master/clase-5/practica-clase-5-res

¡Hasta la próxima!