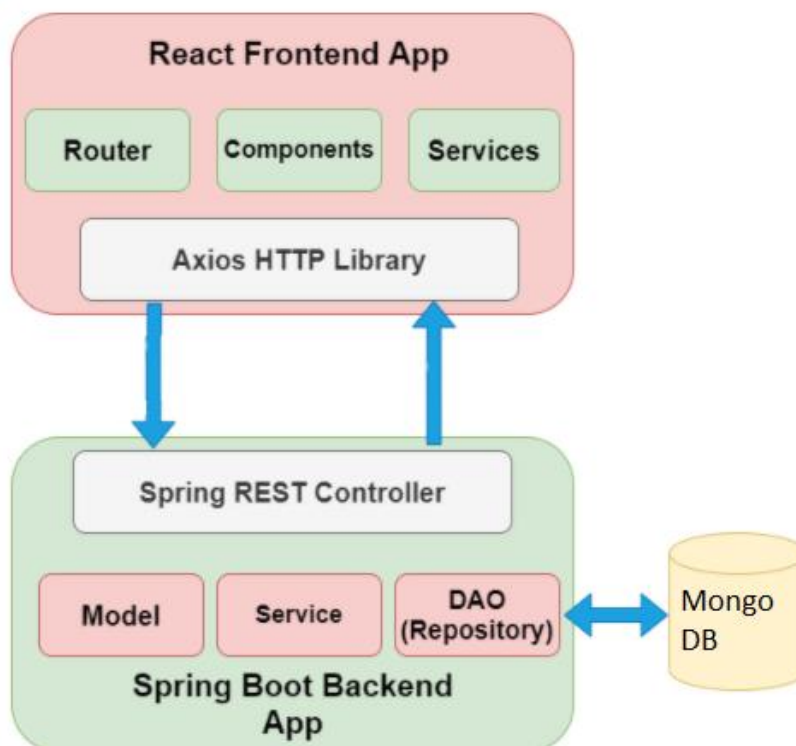


Aplicación de interfaz de usuario React JS.

Vamos a completar el microservicio de clientes, ya hicimos nuestro proyecto backend en el anterior ejercicio, hoy vamos a consumir el servicio Rest con React, en el siguiente gráfico mostramos la arquitectura del micorservicio.

Spring Boot React Full-Stack Architecture



Crearemos una aplicación React y consumiremos las API CRUD Restful desarrolladas y expuestas por el proyecto Spring boot, para ello seguiremos los siguientes pasos:

1. Crear React UI con "Create React App".
2. Agregar Bootstrap en React usando NPM.
3. Componente React de Servicio ClienteServicio – Llama a REST API.
4. Desarrollar el Componente React.

5. Ajuste de App.js.
6. Ejecutar React App

1. Crear React UI con "Create React App"

Podemos crear el esqueleto de una aplicación React de varias maneras, entre ellas es popular el uso de una aplicación llamada "Create React App" que es ideal para crear una aplicación de una sola página.

Como sabemos, React es una biblioteca basada en JavaScript que no tiene la capacidad de realizar solicitudes HTTP; por lo tanto, necesitamos utilizar bibliotecas de terceros para lograrlo.

Hay muchas bibliotecas disponibles para realizar llamadas HTTP en la aplicación React. Algunos de ellos se enumeran a continuación.

- Axios
- Fetch
- Superagent
- React-axios
- Use-http
- React-request

Usaremos la biblioteca HTTP de Axios para realizar una llamada HTTP Get REST API en este ejemplo.

Comencemos con la creación de una aplicación React usando la CLI create-react-app.

Create React App

Create React App es un ambiente cómodo para **aprender React**, y es la mejor manera de comenzar a construir **una nueva aplicación de página única** usando React.

Create React App configura tu ambiente de desarrollo de forma que puedas usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando tu aplicación para producción. Necesitarás tener Node $\geq 14.0.0$ y npm ≥ 5.6 instalados en tu máquina. Para crear un proyecto ejecuta:

```
npx create-react-app my-app  
cd my-app  
npm start
```

Para poder ejecutar esta instrucción se debe instalar previamente con la siguiente instrucción sobre una terminal de Visual Estudio:

npm install -g create-react-app

La aplicación "Create React App" es una forma compatible de crear aplicaciones React de una sola página. Ofrece una configuración de construcción moderna sin configuración.

Para crear una nueva aplicación, puede elegir uno de los siguientes métodos:

Using npx

```
npx create-react-app react-frontend
```

Using npm

```
npm init react-app react-frontend
```

npm init is available in npm 6+

Using Yarn

```
yarn create react-app react-frontend
```

La ejecución de cualquiera de estos comandos creará un directorio llamado react-frontend dentro de la carpeta actual. Dentro de ese directorio, generará la estructura inicial del proyecto e instalará las dependencias transitivas:

```
react-frontend
├─ README.md
├─ node_modules
├─ package.json
├─ .gitignore
├─ 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
   └─ serviceWorker.js
```

Exploremos archivos y carpetas importantes del proyecto react.

Para que el proyecto se compile, estos archivos deben existir con nombres de archivo exactos:

public/index.html: es la plantilla de página;

src/index.js: es el punto de entrada de JavaScript.

Puede eliminar o cambiar el nombre de los otros archivos. Revisemos rápidamente la estructura del proyecto.

package.json: el archivo package.json contiene todas las dependencias necesarias para nuestro proyecto React JS. Lo más importante es que

	<p style="text-align: center;">MINTIC 2022 Desarrollo de Aplicaciones Web Aplicación de interfaz de usuario React JS</p>	
---	---	---

puede verificar la versión actual de React que está utilizando. Tiene todos los scripts para iniciar, construir y expulsar nuestra aplicación React.

carpeta public: la carpeta public contiene index.html . Como se usa react para construir una aplicación de una sola página, tenemos este único archivo HTML para renderizar todos nuestros componentes. Básicamente, es una plantilla HTML. Tiene un elemento div con id como root y todos nuestros componentes se representan en este div con index.html como una sola página para la aplicación completa de reacción.

Carpeta src: en esta carpeta, tenemos todos los archivos javascript y CSS globales. Todos los diferentes componentes que estaremos construyendo, siéntense aquí.

index.js: este es el procesador superior de su aplicación de reacción.

node_modules: todos los paquetes instalados por NPM o Yarn residirán dentro de la carpeta node_modules.

App.js: el archivo App.js contiene la definición de nuestro componente de aplicación que en realidad se representa en el navegador y este es el componente raíz.

Después de la ejecución de la aplicación se generará el proyecto, mostrando un mensaje como el que sigue:

```
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 frontend
npm start

Happy hacking!
PS C:\Users\Carlos Beltrán\frontend> |
```

Después de cambiar de directorio y ejecutar npm start y aparece un mensaje como el siguiente:

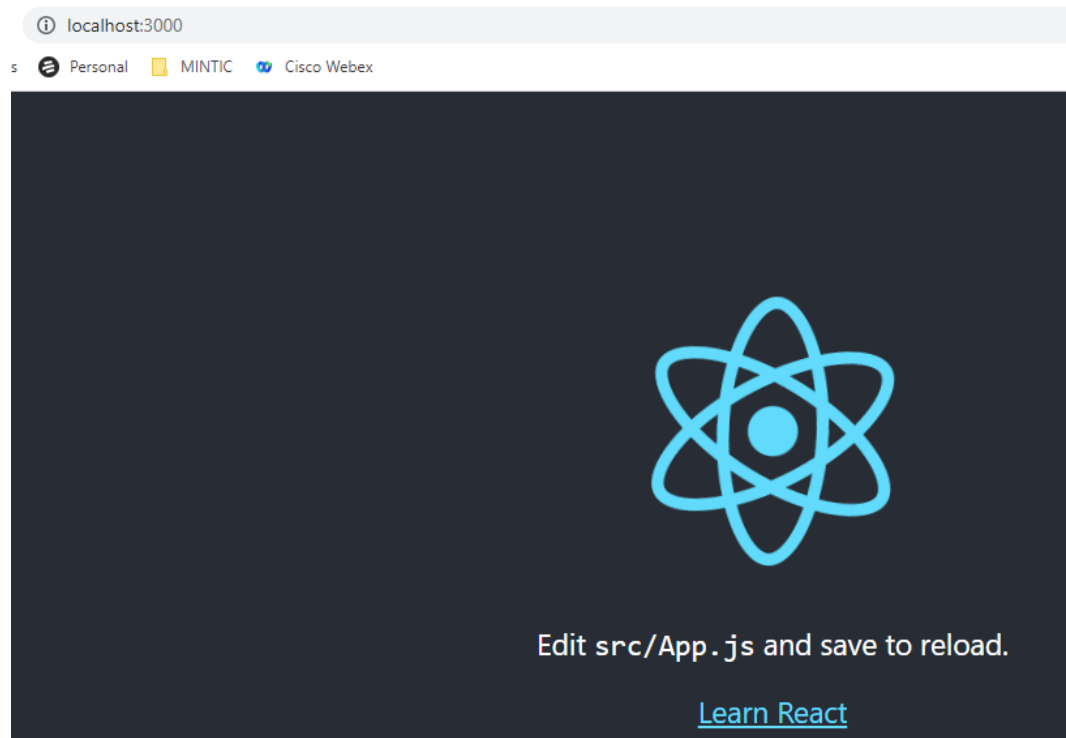
```
Compiled successfully!

You can now view frontend in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.1.5:3000

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

En el navegador se puede apreciar la siguiente presentación:



2. Agregar Bootstrap en React usando NPM.

- En una terminal de Windows, escriba el siguiente comando que permite instalar Bootstrap:

```
$ npm install bootstrap --save
```

- Ahora para que Bootstrap funcione correctamente es necesario instalar jQuery, ejecuto el siguiente comando.

npm install jquery --save

- Y también necesitamos instalar la librería Popper JS, ejecutamos el siguiente comando en la consola de comandos para instalarla.

npm install popper.js --save

Después de instalar el paquete de arranque, deberá importarlo en su archivo de entrada de la aplicación React.

Abra el archivo **src/index.js** y agregue el siguiente código:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

src/index.js

Aquí está el código completo para el archivo index.js :

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
import 'bootstrap/dist/css/bootstrap.min.css';

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

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

3. Componente React de Servicio ClienteServicio – Llama a REST API.

Para nuestras llamadas a la API, usaremos Axios. A continuación se muestra el comando "npm" para instalar Axios.


```
npm add axios
```

A continuación, se muestra la implementación del servicio ClienteService.js para realizar nuestra llamada HTTP REST a través de Axios.

Nuestro endpoint de clientes del backend está disponible en "http://localhost:8082/api/clientes".

Dentro de la carpeta src creamos otra llamada servicio y allí creamos el archivo o componente llamado ClienteServicio, su contenido se puede apreciar en la siguiente gráfica:

```
src > servicios > JS ClienteServicio.js > [⌘] CLIENTE_REST_API_URL
1  import axios from "axios";
2
3  const CLIENTE_REST_API_URL = 'http://localhost:8082/api/clientes/clientes';
4
5  class ClienteServicio{
6    getClientes(){
7      return axios.get(CLIENTE_REST_API_URL);
8    }
9  }
10
11  export default new ClienteServicio();
```

4. Desarrollar el Componente React.

Los componentes permiten separar la interfaz de usuario en partes independientes, reutilizables y pensar en cada parte 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.

Creamos un archivo ClienteComponente dentro de la carpeta componentes, el contenido es como se puede apreciar en la gráfica:

```
src > componentes > JS ClienteComponente.js > ClienteComponente > render
1  import React from 'react';
2  import ClienteServicio from '../servicios/ClienteServicio';
3
4  class ClienteComponente extends React.Component{
5
6      constructor(props){
7          super(props)
8          this.state = {
9              clientes:[]
10         }
11     }
12
13     componentDidMount(){
14         ClienteServicio.getClientes().then((response) => {
15             this.setState({clientes: response.data})
16         })
17     }
18
```

```
19     render(){
20         return(
21             <div>
22                 <h1 className = "text-center"> Lista de Clientes</h1>
23                 <table className = "table table-striped">
24                     <thead>
25                         <tr>
26
27                             <td> Cliente Id</td>
28                             <td> Cédula</td>
29                             <td> Nombre</td>
30                             <td> Dirección</td>
31                             <td> Correo</td>
32                             <td> Teléfono</td>
33                         </tr>
34                     </thead>
35
```

```

36         <tbody>
37         {
38             this.state.clientes.map(
39                 cliente =>
40                 <tr key = {cliente._id}>
41                     <td> {cliente._id}</td>
42                     <td> {cliente.cedula}</td>
43                     <td> {cliente.nombre}</td>
44                     <td> {cliente.direccion}</td>
45                     <td> {cliente.email}</td>
46                     <td> {cliente.telefono}</td>
47                 </tr>
48             )
49         }
50     </tbody>
51 </table>
52 </div>
53 )
54 }
55 }
56 }
57 }
58
59 export default ClienteComponente

```

- **constructor()** - The **constructor()** es invocado antes de que sea montado. En el constructor, hemos declarado nuestras variables de estado y vinculamos los diferentes métodos para que sean accesibles desde el estado dentro del método render ().
- **componentDidMount()** - Se invoca inmediatamente después de que un componente se monte (se inserte en el árbol). La inicialización que requiere nodos DOM debería ir aquí. Si necesita cargar datos desde un punto final remoto, este es un buen lugar para instanciar la solicitud de red.
- **render()** - El **render()** es el método más usado en el ciclo de vida de React. El método render () en realidad genera HTML en el DOM..

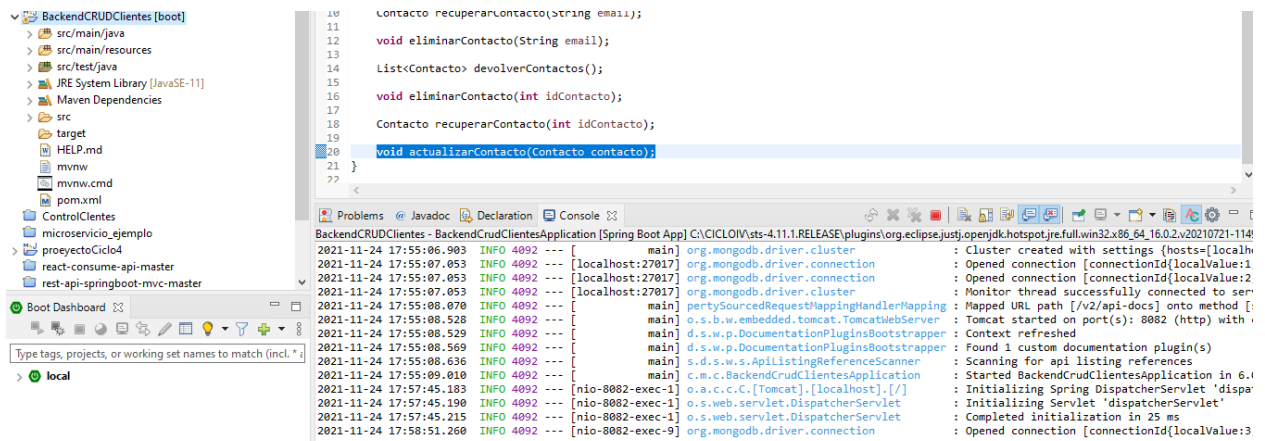
5. Ajuste de App.js.

Ajustamos el archivo App.js para poder mostrar la lista de los clientes.

```
src > JS App.js > ...
1  //import logo from './logo.svg';
2  import './App.css';
3  import ClienteComponente from './componentes/ClienteComponente';
4
5  function App() {
6    return (
7      <div className="App">
8        <ClienteComponente />
9      </div>
10   );
11 }
12
13 export default App;
14
```

6. Ejecutar React App.

Para poder visualizar el resultado de nuestro front con React, debemos iniciar primero el RestApi, lo abrimos desde STS o ejecutamos el archivo jar generado en el backend.



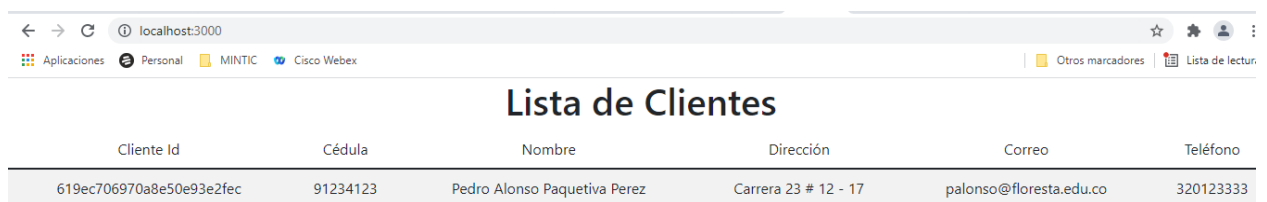
A continuación, ejecutamos el frontend con cualquiera de los siguientes comandos, con npm sería de la siguiente forma:

```
npm start
```

Con yarn se ejecuta de la siguiente forma:

```
yarn start
```

El resultado del front sería como se aprecia:



Lista de Clientes					
Cliente Id	Cédula	Nombre	Dirección	Correo	Teléfono
619ec706970a8e50e93e2fec	91234123	Pedro Alonso Paquetiva Perez	Carrera 23 # 12 - 17	palonso@floresta.edu.co	320123333