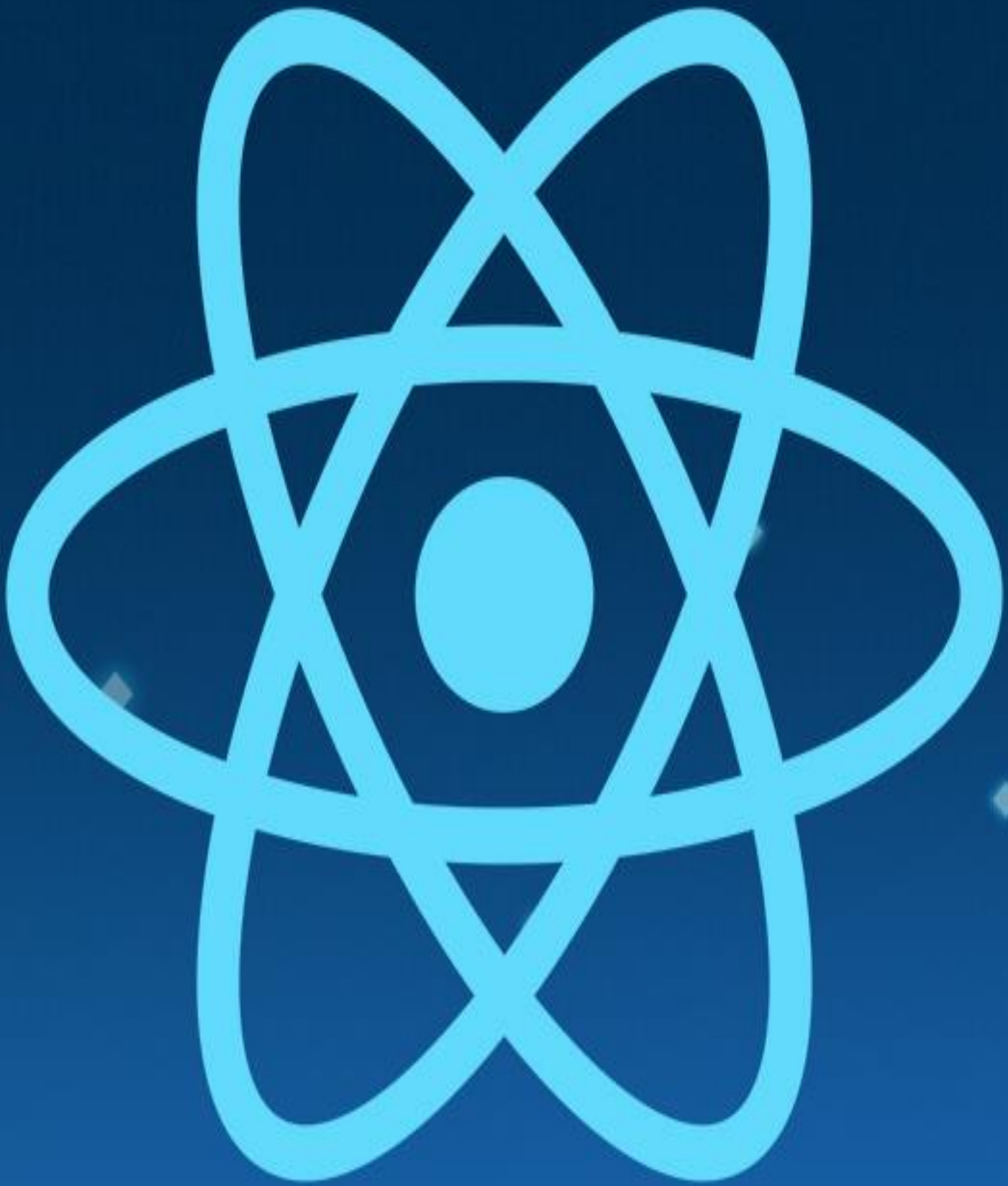


Taller React.js

Objetivo

El propósito de este taller es introducir el framework React.js en un proyecto pequeño pero compuesto por un Stack completo, en este caso será Mern-Stack comprendido por las tecnologías MongoDB, Express, Node y React todas basadas en JavaScript. El proyecto se compone de vistas frontend las cuales serán creadas con React.js, bibliotecas que facilitan la conexión con el backend y con la base de datos proporcionando un ambiente completo de desarrollo.

Sobre el problema planteado

Se presenta una aplicación sencilla de vista, creación, edición y eliminación de notas por usuario, también permite la creación y eliminación de usuarios. Todos los cambios en la aplicación se verán reflejados en la base de datos MongoDB. El proyecto incluirá librerías propias de React y librerías externas para frontend, routing y actualización automática de cambios.

Requisitos previos

- Se requiere instalación de Node.js y MongoDB.
- Se requiere haber completado la guía de instalación de herramientas del Instructivo de Instalación React.

NOTA: El backend no se desarrollará durante el taller, solo se explicarán ciertos elementos de este.

Por hacer

BACKEND

1. Descomprima el archivo adjunto “backendFiles.rar”, ahí se encuentra una carpeta “src” con toda la lógica del backend y un archivo “.env”, agregue ambos a su proyecto de la carpeta backend.
2. Dentro del archivo package.json en backend, edite la sección scripts y agregue lo siguiente

```
3. "dev":"nodemon src/index.js"
```

   Este paso sirve para indicar una nueva forma de levantar el backend junto con la base de datos.

FRONTEND

1. Dentro de la carpeta src en frontend borre los siguientes archivos
   a. App.test.js
   b. serviceWorker.js

c. logo.svg

2. Abra el archivo index.js y borre las líneas:

```
a. serviceWorker.unregister();
```

```
b. import * as serviceWorker from './serviceWorker';
```

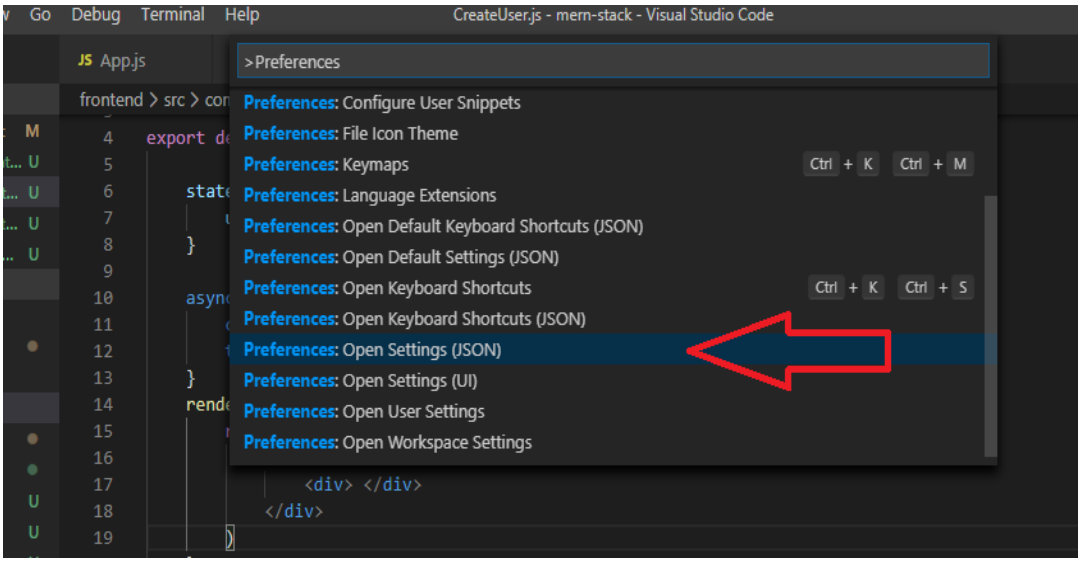
3. Dirijase al archivo “App.js”, borre el contenido de la etiqueta div en function App() y cámbielo por un mensaje cualquiera de momento.

```
function App() {
  return (
    <div>
      Hello Web Development
    </div>
  );
}
```

4. Elimine también de App.js el import de logo.svg

```
import logo from './logo.svg';
```

5. Preferencias de desarrollo(**Vs Code**, no requerido pero eficiente):
   a. Busque el archivo settings.json. F1 para abrir el buscador.



   b. En el archivo settings.json incluya javascriptreact en la sección emmet.

```
"emmet.includeLanguages":{
          "javascript":"javascriptreact"
```

```
    }
```

6. Puede incluir también la extensión de React para facilitar la creación de componentes.

--- A partir de este punto comienza la creación de elementos propios de nuestro proyecto ---

7. Cree una carpeta dentro de "src" llamada "components" y en ella cree 4 archivos js:
    a. CreateNote.js
    b. CreateUser.js
    c. Navigation.js
    d. NotesList.js
8. Borre el contenido del archivo index.css e inserte el siguiente:

```css
body{
    background: #E0EAFC;
    background: -webkit-linear-gradient(to right, #CFDEF3, #E0EAFC);
    background: linear-gradient(to right, #CFDEF3, #E0EAFC);

}
```

NAVIGATION.js

Esta será la vista principal de nuestra aplicación, mostrará una barra de navegación con links a las otras 3 vistas que se mostrarán dentro de esta.

9. Dentro del archivo Navigation.js cree la clase Navigation, si instaló la extensión de React solo escriba rcc y presione enter, la clase se creará automáticamente.

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

export default class Navigation extends Component {
    render() {
        return (
            <div>

            </div>
```

```
        )
    }
}
```

10. Agregue el import de “Link” que permite dirigir a las otras vistas

```
import {Link} from 'react-router-dom'
```

11. Dentro del return en el método render borre el div actual y agregue el JSX de Bootstrap con la barra de navegación.

```
<nav className="navbar navbar-expand-lg navbar-dark bg-dark">
                <div className="container">
                    <Link className="navbar-brand" to="/">
                        NotesApp
                    </Link>
                    <button className="navbar-
toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
                        <span className="navbar-toggler-icon"></span>
                    </button>

                    <div className="collapse navbar-
collapse" id="navbarSupportedContent">
                        <ul className="navbar-nav ml-auto">
                            <li className="nav-item active">
                                <Link className ="nav-
link" to="/">Notes</Link>
                            </li>
                            <li className="nav-item">
                                <Link className ="nav-
link" to="/create">Create Note</Link>
                            </li>
                            <li className="nav-item">
                                <Link className ="nav-
link" to="/user">Create User</Link>
                            </li>
                        </ul>
                    </div>
                </div>

            </nav>
```

NOTA: Si al copiar el JSX no trae el formato adecuado puede utilizar el comando **F1 format document**, esto arreglará las tabulaciones del JSX para que sea visualmente mas legible.

NOTESLIST.js

Este componente se encargará de mostrar todas las notas registradas en la base de datos, cada nota será mostrada mediante un card de Bootstrap que contendrá un header con un titulo y un botón de edición, un body con el contenido de la nota, el usuario que la creó y una fecha en formato "timeago".

12. Escriba la clase NotesList con rcc o bien copie el código:

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

export default class NotesList extends Component {
    render() {
        return (
            <div>

            </div>
        )
    }
}
```

13. Importe las siguientes librerias dentro de NotesList.js
    a. Axios: se utiliza para comunicar el frontend con el backend por medio de Http requests.

```
import axios from 'axios'
```

    b. Format: se utiliza para cambiar fechas por un formato en términos de tiempo pasado o futuro. Ejm: "18-01-20" -> "2 days ago".

```
import {format} from 'timeago.js'
```

    c. Link: para dirigir a otra vistas.

```
import {Link} from 'react-router-dom'
```

14. Dentro del return en el método render borre el div y copie el siguiente código:

```
<div className="row">
                {
                    this.state.notes.map(note =>
                        <div className="col-md-4 p-2" key={note._id}>
                            <div className="card">
```

```
                                <div className="card-header d-
flex justify-content-between">
                                    <h5>{note.title}</h5>
                                    <Link classname="btn btn-
secondary" to={"/edit/"+note._id}>
                                        Edit
                                    </Link>
                                </div>
                                <div className="card-body">
                                    <p>{note.content}</p>
                                    <p>{note.author}</p>
                                    <p>{format(note.date)}</p>
                                </div>
                                <div className="card-footer">
                                    <button className="btn btn-
danger" onClick={()=>this.deleteNote(note._id)}>
                                        Delete
                                    </button>
                                </div>
                            </div>
                        </div>
                    )
                    }
            </div>
```

15. Agregue dentro de la clase NotesList los siguientes métodos:
    a. componentDidMount: se utiliza para cargar todas las notas existentes en el momento que carga la vista.

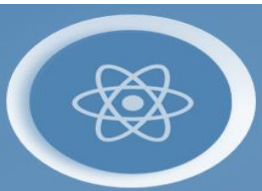
```
async componentDidMount(){
        this.getNotes()
    }
```

    b. getNotes: hace la petición de las notas al backend

```
async getNotes(){
        const res = await axios.get('http://localhost:4000/api/notes')
        this.setState({notes: res.data})
    }
```

    c. deleteNote: Borra la nota seleccionada y refresca la vista

```
deleteNote = async (id) =>{
        await axios.delete('http://localhost:4000/api/notes/'+id)
        this.getNotes();
    }
```



d. state: variable de estado para almacenar las notas

```
state={
        notes:[]
    }
```

CREATENOTE.js

Esta es la vista que contendrá el formulario donde se crea una nueva nota. Este mismo componente maneja 2 funciones, crear y editar por medio de 2 estados diferentes.

16. Escriba la clase CreateNote con rcc o bien copie el código:

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

export default class CreateNote extends Component {
    render() {
        return (
            <div>

            </div>
        )
    }
}
```

17. Importe las siguientes librerías dentro CreateNote.js
    a. Axios

```
import axios from 'axios'
```

    b. DatePicker: Es un componente de React que muestra un calendario interactivo para seleccionar fechas.

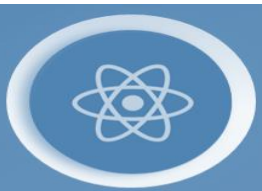
```
import DatePicker from 'react-datepicker'
import 'react-datepicker/dist/react-datepicker.css'
```

18. Dentro del return en el método render borre el div y copie el siguiente JSX:

```
<div className="col-md-6 offset-md-3">
                <div className="cad card-body">
                    <h4>Create a Note</h4>

                    {/*SELECT USER*/}

                    <div className="form-group">
```

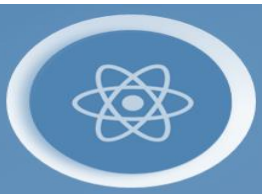


```
                    <select
                        className="form-control"
                        name="userSelected"
                        onChange={this.OnInputChange}
                        value={this.state.userSelected}
                    >
                        {
                            this.state.users.map(user =>
                            <option key={user} value={user}>
                                {user}
                            </option>)
                        }
                    </select>
                </div>
                <div className="form-group">
                    <input
                        type="text"
                        className="form-control"
                        placeholder="Title"
                        name="title"
                        onChange={this.OnInputChange}
                        value={this.state.title}
                        required/>
                </div>

                <div className="form-group">
                    <textarea
                        name="content"
                        className="form-control"
                        placeholder="Content"
                        onChange={this.OnInputChange}
                        value={this.state.content}
                        required>

                    </textarea>
                </div>
                <div className="from-group">
                    <DatePicker
                        className="form-control"
                        selected={this.state.date}
                        onChange={this.onChangeDate}
                        />
                </div>

                <form onSubmit={this.onSubmit}>
```



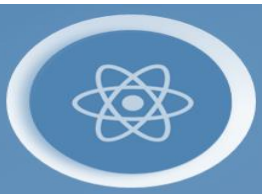
```
                        <button type="submit" className ="btn btn-
primary">
                            Save
                        </button>
                    </form>
                </div>
            </div>
```

19. Dentro de la clase CreateNote incluya los siguientes métodos:
    a. componentDidMount: hace una petición de los usuarios y en caso de que sea una nota para editar, hace una petición de esa nota.

```
async componentDidMount(){
        const res = await axios.get('http://localhost:4000/api/users')
;
        if (res.data.length > 0) {
            this.setState({
                users:res.data.map(user=>user.username),
                userSelected: res.data[0].username
        })
        }
        if(this.props.match.params.id){
            const res = await axios.get('http://localhost:4000/api/not
es/'+this.props.match.params.id);
            this.setState({
                title:res.data.title,
                content:res.data.content,
                date:new Date(res.data.date),
                userSelected:res.data.author,
                editing:true,
                _id: res.data._id
            })
        }
    }
```

    b. onSubmit: maneja el evento de crear o editar una nota.

```
onSubmit = async (e) =>{
        e.preventDefault();
        const newNote = {
            title: this.state.title,
            content: this.state.content,
            date: this.state.date,
            author: this.state.userSelected
        };
```



```
        if(this.state.editing){
            await axios.put('http://localhost:4000/api/notes/'+this.st
ate._id,newNote);
        }else{
            await axios.post('http://localhost:4000/api/notes',newNote
);
        }
        window.location.href = "/";
    }
```

c. OnInputChange: Cada vez que se introduce texto o se hace un cambio en el formulario, este método actualiza el estado actual con los datos introducidos.

```
OnInputChange = e => {
        this.setState({
            [e.target.name]: e.target.value
        })
    }
```

d. onChangeDate: maneja el evento de cambio de fecha con el date-picker

```
onChangeDate = date => {
        this.setState({date});
    }
```

e. state: contiene la lista de usuarios actuales y los elementos de la nota.

```
state={
        users:[],
        userSelected:'',
        title:'',
        content:'',
        date: new Date(),
        editing:false,
        _id: ''
    }
```

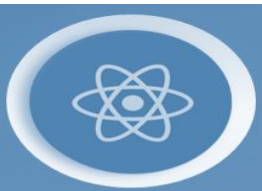
CREATEUSER.js

Este componente se encarga de crear usuarios y visualizar una lista completa de los existentes. Los usuarios no tienen atributos más que un username.

20. Escriba la clase CreateUser con rcc o bien copie el código:

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

export default class CreateUser extends Component {
```



```
    render() {
        return (
            <div>

            </div>
        )
    }
}
```

21. Importe la librería Axios en CreateUser.js

```
import axios from 'axios'
```

22. Dentro del return en el método render borre el div e incluya el siguiente código JSX:

```
<div className="row">
                <div className="col-md-4">
                    <div className="card card-body">
                        <h3>Create New User</h3>
                            <form onSubmit={this.onSubmit}>
                                <div className="form-group">
                                    <input
                                        type="text"
                                        className="form-control"
                                        value={this.state.username}
                                        onChange={this.onChangeUserNam
e} />
                                </div>
                                <button type="submit" className="btn b
tn-primary">

                                    Agregar usuario
                                </button>
                            </form>
                    </div>
                </div>
                <div className="col-md-8">
                    <ul className="list-group">
                        {
                            this.state.users.map(user => (
                            <li className="list-group-item list-group-
item-action"

                                key={user._id}
                                onDoubleClick={()=>this.deleteUser(use
r._id)}

                                >
```

```
                                {user.username}
                            </li>))
                        }
                    </ul>
                </div>
            </div>
```

23. Agregue los siguientes métodos dentro de la clase CreateUser

    a. componentDidMount: Para obtener la lista de usuarios al abrir la vista.

```
async componentDidMount(){
        this.getUsers();
    }
```

    b. getUsers: hace la petición Http para obtener los usuarios.

```
getUsers = async () => {
        const res = await axios.get('http://localhost:4000/api/users')
;
        this.setState({users: res.data});
    }
```

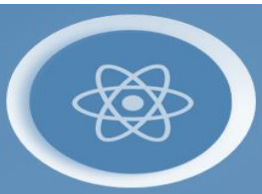
    c. onChangeUserName: Para manejar el evento cada vez que hay un cambio al introducir el nombre.

```
onChangeUserName = (e) => {
        this.setState({
            username: e.target.value
        })
    }
```

    d. onSubmit: Para hacer el registro del usuario por medio de un Http post. También actualiza la lista de usuarios dentro de la misma vista

```
onSubmit = async e => {
        e.preventDefault();//Evitar resetear form y pagina
        const res = await axios.post('http://localhost:4000/api/users'
,{
            username:this.state.username
        })
        console.log(res)
        this.setState({username:''})
        this.getUsers();

    }
```



e. deleteUser: Hace el Http delete al usuario seleccionado para borrarlo.

```
deleteUser = async (id) => {
        await axios.delete('http://localhost:4000/api/users/' + id);
        this.getUsers();
    }
```

f. State: Variable que maneja el estado actual.

```
state={
        users: [],
        username: ''
    }
```

APP.js

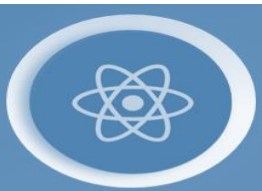
Este es archivo javascript que inicia la ejecución de la aplicación por medio de function App(), por lo que aquí se incluyen todos los componentes creados hasta el momento.

24. Dentro del archivo App.js incluya los siguientes imports:

```
import {BrowserRouter as Router, Route} from 'react-router-dom'
import 'bootstrap/dist/css/bootstrap.min.css';
import Navigation from './components/Navigation'
import NotesList from './components/NotesList'
import CreateNote from './components/CreateNote'
import CreateUser from './components/CreateUser'
```

25. Cambie el método function App() de esta forma:

```
function App() {
  return (
    <Router>
      <Navigation/>
      <div className="container p-4">
        <Route path="/" exact component={NotesList}/>
        <Route path="/edit/:id" component={CreateNote}/>
        <Route path="/create" component={CreateNote}/>
        <Route path="/user" component={CreateUser}/>
      </div>
    </Router>
  );
}
```



Hasta este punto ya nuestra aplicación web debe funcionar sin problemas. Para ejecutar el proyecto deben estar levantados 2 servers y la base de datos MongoDB. Para ello se van a ejecutar 2 comandos.

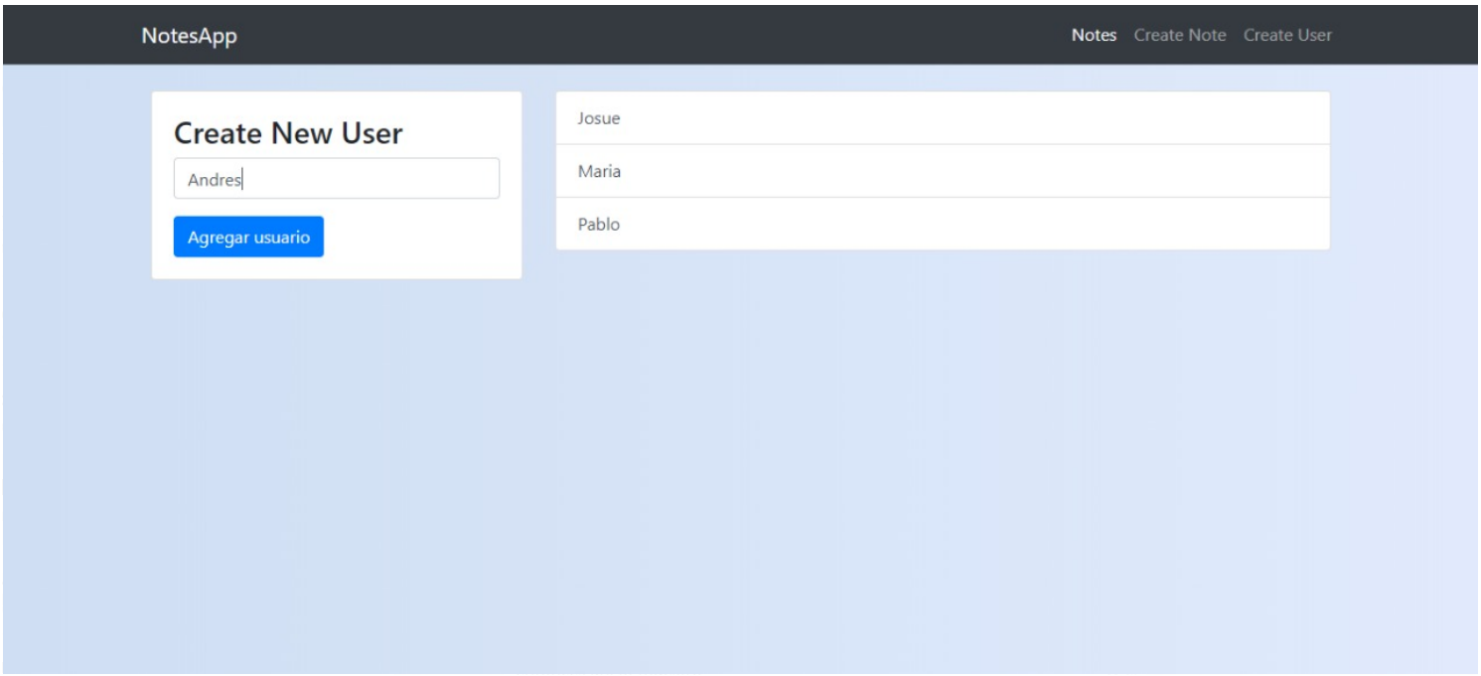
- Abra 2 líneas de comando, puede abrirlas dentro de Visual Studio Code o externas.
- En la primera ubique la carpeta backend y ejecute el comando:

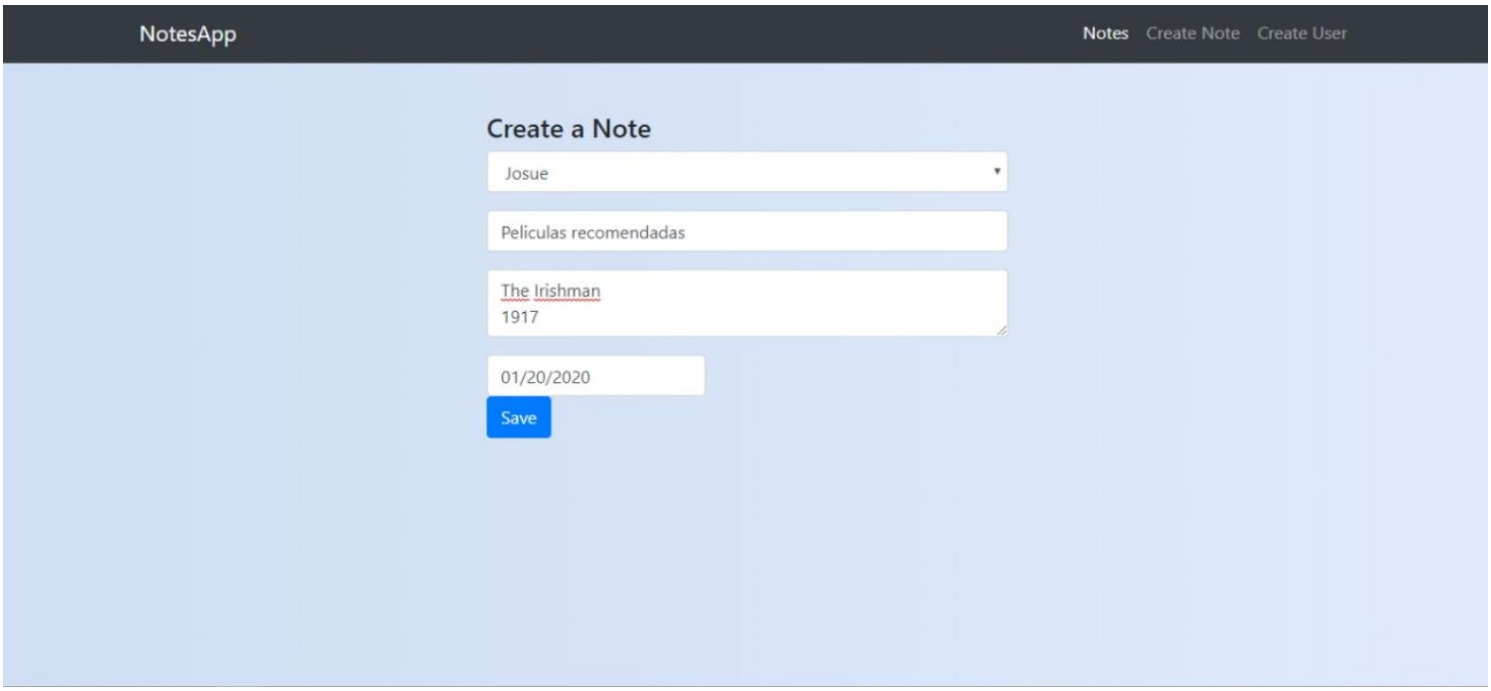
```
backend>npm run dev
```

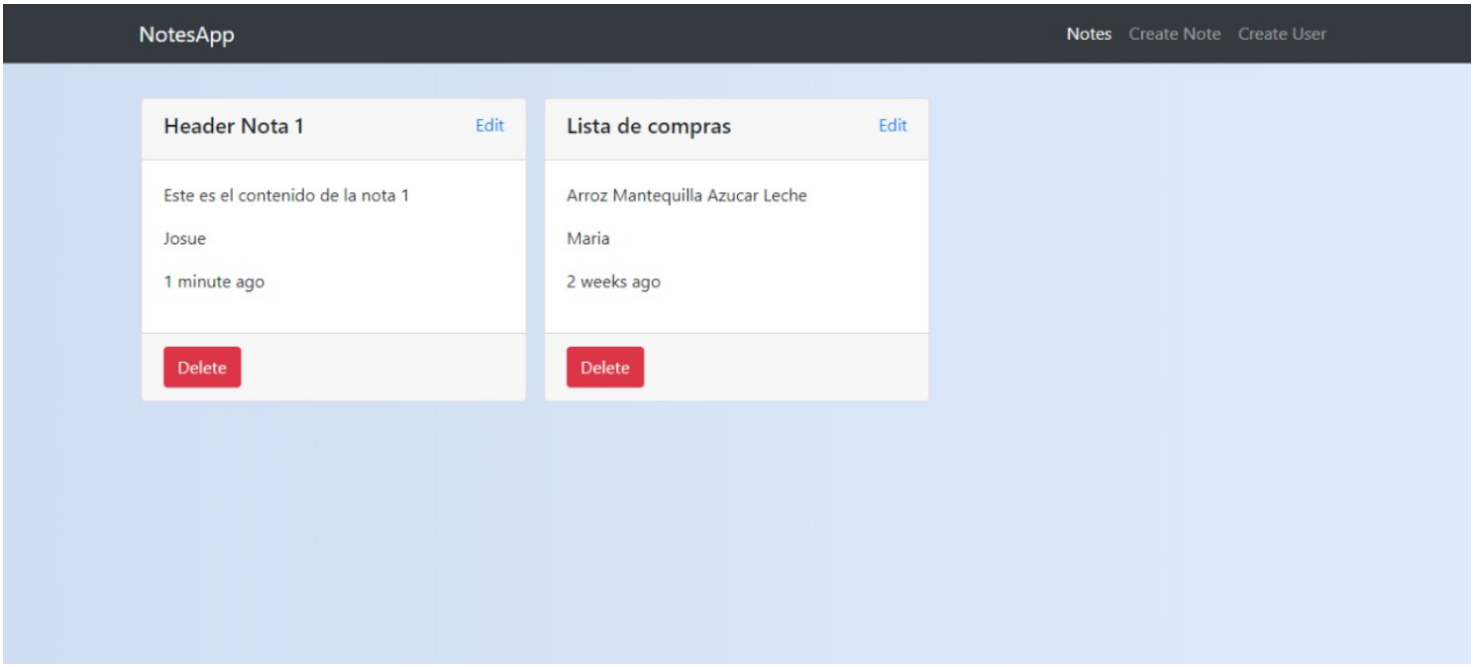
- Espere a que se muestren los mensajes: Server on port 4000 y DB is connected.
- En la segunda, diríjase a la carpeta frontend y ejecute el comando:

```
frontend>n pm start
```

Resultado final




NotesApp
Notes
Create Note
Create User
Create a Note
Josue
Peliculas recomendadas
The Irishman
1917
01/20/2020
Save


NotesApp
Notes
Create Note
Create User
Header Nota 1
Edit
Este es el contenido de la nota 1
Josue
1 minute ago
Delete
Lista de compras
Edit
Arroz Mantequilla Azucar Leche
Maria
2 weeks ago
Delete