Manual Técnico Didáctico – Sistema RRHH (Spring Boot + React + MySQL)

---

## Parte 1: Backend (Spring Boot + Java 24 + MySQL)

[...Contenido ya incluido anteriormente...]

---

## Parte 2: Frontend – Instalación moderna con Vite + React

### 1. Crear carpeta del frontend

Ubícate en tu carpeta de trabajo, por ejemplo:

```bash

mkdir Recursos-Humanos-App

cd Recursos-Humanos-App

```

### 2. Crear proyecto con Vite

Ejecuta:

```bash

npm create vite@latest . -- --template react

```

> El punto `.` indica que se instalará en la carpeta actual. Si no tienes Vite instalado, te pedirá confirmación.

### 3. Instalar dependencias

```bash

npm install

```

### 4. Ejecutar la aplicación

```bash

npm run dev

```

La app se ejecutará en:

```

http://localhost:5173

```

### 5. Configurar el backend Spring Boot para permitir peticiones desde React (puerto 5173)

En tu controlador REST, añade:

```java

@CrossOrigin(origins = "http://localhost:5173")

```

### 6. Resultado esperado

- React se instala con Vite.

- El frontend es más ligero y rápido que con create-react-app.

- Puedes usar App.jsx para empezar a crear tus componentes.

### 7. Próximos pasos

- Crear componentes (lista, formulario, etc.).

- Instalar `axios` para conectar con la API REST.

- Diseñar las operaciones CRUD.

---

## Parte 3: CRUD completo con React + Axios + Backend

### 1. Instalar Axios

Desde la raíz del proyecto React:

```bash

npm install axios

```

### 2. Crear estructura básica en `src/`

```

src/

├── componentes/

│ ├── EmpleadoLista.js

│ └── EmpleadoFormulario.js

└── servicios/

└── EmpleadoServicio.js

```

### 3. Crear servicio API – `EmpleadoServicio.js`

```javascript

import axios from 'axios';

const API\_URL = 'http://localhost:8080/rh-app/empleados';

class EmpleadoServicio {

obtenerEmpleados() {

return axios.get(API\_URL);

}

crearEmpleado(empleado) {

return axios.post(API\_URL, empleado);

}

actualizarEmpleado(empleado) {

return axios.put(API\_URL, empleado);

}

eliminarEmpleado(empleado) {

return axios.delete(API\_URL, { data: empleado });

}

}

export default new EmpleadoServicio();

```

### 4. Lista de empleados – `EmpleadoLista.js`

```javascript

import React, { useEffect, useState } from 'react';

import EmpleadoServicio from '../servicios/EmpleadoServicio';

function EmpleadoLista() {

const [empleados, setEmpleados] = useState([]);

useEffect(() => {

cargarEmpleados();

}, []);

const cargarEmpleados = () => {

EmpleadoServicio.obtenerEmpleados()

.then(res => setEmpleados(res.data))

.catch(err => console.error(err));

};

const eliminarEmpleado = (empleado) => {

EmpleadoServicio.eliminarEmpleado(empleado)

.then(() => cargarEmpleados())

.catch(err => console.error(err));

};

return (

<div>

<h2>Lista de Empleados</h2>

<table>

<thead>

<tr>

<th>ID</th><th>Nombre</th><th>Departamento</th><th>Sueldo</th><th>Acciones</th>

</tr>

</thead>

<tbody>

{empleados.map(e => (

<tr key={e.idEmpleado}>

<td>{e.idEmpleado}</td>

<td>{e.nombre}</td>

<td>{e.departamento}</td>

<td>{e.sueldo}</td>

<td>

<button onClick={() => eliminarEmpleado(e)}>Eliminar</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

export default EmpleadoLista;

```

### 5. Componente principal – App.js (básico)

```javascript

import React from 'react';

import EmpleadoLista from './componentes/EmpleadoLista';

function App() {

return (

<div className="App">

<h1>Gestión de Empleados</h1>

<EmpleadoLista />

</div>

);

}

export default App;

```

### 6. Resultado esperado

- Verás un listado con empleados cargados desde la base de datos.

- Botón para eliminar funciona directamente contra el backend Spring Boot.

### 7. Próximos pasos

- Añadir formulario para crear y editar empleados.

- Añadir validaciones y diseño con CSS o librerías como Bootstrap.

- Manejar errores de red y mostrar mensajes al usuario.

---

## Parte 4: Formulario React para alta y edición de empleados

### 1. Crear componente: `EmpleadoFormulario.js`

```javascript

import React, { useState } from 'react';

import EmpleadoServicio from '../servicios/EmpleadoServicio';

function EmpleadoFormulario({ onEmpleadoAgregado }) {

const [empleado, setEmpleado] = useState({ nombre: '', departamento: '', sueldo: '' });

const manejarCambio = (e) => {

const { name, value } = e.target;

setEmpleado({ ...empleado, [name]: value });

};

const manejarEnvio = (e) => {

e.preventDefault();

EmpleadoServicio.crearEmpleado(empleado)

.then(() => {

setEmpleado({ nombre: '', departamento: '', sueldo: '' });

onEmpleadoAgregado();

})

.catch(err => console.error(err));

};

return (

<form onSubmit={manejarEnvio}>

<h3>Agregar Empleado</h3>

<input type="text" name="nombre" placeholder="Nombre" value={empleado.nombre} onChange={manejarCambio} required />

<input type="text" name="departamento" placeholder="Departamento" value={empleado.departamento} onChange={manejarCambio} required />

<input type="number" name="sueldo" placeholder="Sueldo" value={empleado.sueldo} onChange={manejarCambio} required />

<button type="submit">Guardar</button>

</form>

);

}

export default EmpleadoFormulario;

```

### 2. Actualizar `EmpleadoLista.js` para incluir el formulario

```javascript

import React, { useEffect, useState } from 'react';

import EmpleadoServicio from '../servicios/EmpleadoServicio';

import EmpleadoFormulario from './EmpleadoFormulario';

function EmpleadoLista() {

const [empleados, setEmpleados] = useState([]);

useEffect(() => {

cargarEmpleados();

}, []);

const cargarEmpleados = () => {

EmpleadoServicio.obtenerEmpleados()

.then(res => setEmpleados(res.data))

.catch(err => console.error(err));

};

const eliminarEmpleado = (empleado) => {

EmpleadoServicio.eliminarEmpleado(empleado)

.then(() => cargarEmpleados())

.catch(err => console.error(err));

};

return (

<div>

<EmpleadoFormulario onEmpleadoAgregado={cargarEmpleados} />

<h2>Lista de Empleados</h2>

<table>

<thead>

<tr>

<th>ID</th><th>Nombre</th><th>Departamento</th><th>Sueldo</th><th>Acciones</th>

</tr>

</thead>

<tbody>

{empleados.map(e => (

<tr key={e.idEmpleado}>

<td>{e.idEmpleado}</td>

<td>{e.nombre}</td>

<td>{e.departamento}</td>

<td>{e.sueldo}</td>

<td>

<button onClick={() => eliminarEmpleado(e)}>Eliminar</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

export default EmpleadoLista;

```

### 3. Resultado esperado

- Se muestra un formulario encima de la tabla.

- Al rellenarlo y enviarlo, se guarda un nuevo empleado.

- La tabla se actualiza automáticamente.

### 4. Mejora futura

- Reutilizar el formulario para editar registros (rellenando los datos actuales).

- Agregar validaciones visuales.

- Usar librerías de diseño (Bootstrap, Material UI).

---

Fin de la Parte 4 – Formulario para alta de empleados conectado al backend.