DuocUC® INFORMÁTICA Y TELECOMUNICACIONES

# Integración RESTful

- Desarrollo Fullstack II
- DSY1104

FORMAMOS PERONAS PARA UNA SOCIEDAD MEJOR

01

# Repaso de la sesión anterior

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

# ¿Para qué sería útil crear una aplicación web integral en estos contextos?

1. *Desarrollo de Aplicaciones para Startups*

2. *Automatización de Procesos Empresariales*

3. **Desarrollo de Soluciones para el Cliente**

DuocUC®

02

# Integración y Comunicación REST

# • Crear una aplicación web completa con CRUD

Utilizando tecnologías: Spring Boot (Backend) y React.js (Frontend)

## Backend (Spring Boot):

**Configuración**: Inicia un proyecto con Spring Initializr y selecciona dependencias como Spring Web, Spring Data JPA, y H2/MySQL.

**Modelo**: Define entidades JPA para representar tablas de base de datos.

**Repositorio**: Crea interfaces que extiendan JpaRepository para operaciones CRUD.

**Servicio**: Implementa la lógica de negocio en servicios.

**Controlador REST**: Desarrolla controladores para manejar solicitudes HTTP (GET, POST, PUT, DELETE).

## Frontend (React.js):

**Configuración**: Crea un proyecto con Create React App.

**Componentes**: Diseña componentes para listar, crear, editar y eliminar registros.

**Llamadas a la API**: Usa axios o fetch para interactuar con la API REST de Spring Boot.

**Estado**: Maneja el estado con React hooks (useState, useEffect).

**Enrutamiento:** Implementa react-router para la navegación entre páginas.

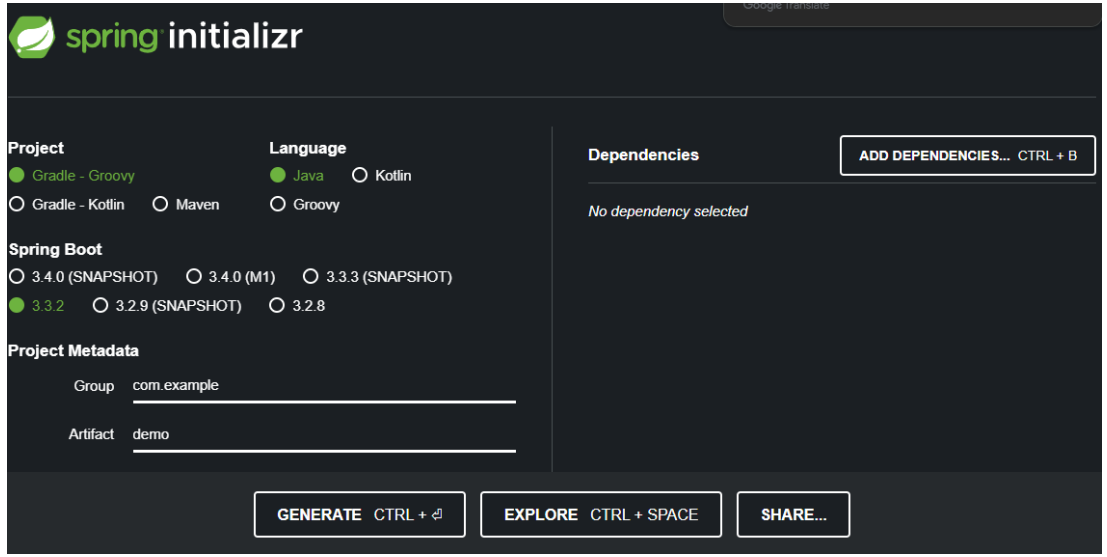
## Integración:

**Despliegue**: Configura el backend en un servidor (Heroku, AWS) y el frontend en Netlify o GitHub Pages.

**Pruebas:** Asegúrate de probar todas las funcionalidades CRUD y la conexión entre frontend y backend.

- # ¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?

## Crear el Backend con Spring Boot




### 1. Configurar el Proyecto

- Visita Spring Initializr.
- Selecciona:Project: Maven Project
- Language: Java
- Spring Boot: 3.0.0 o superior
- Dependencies: Spring Web, Spring Data JPA, H2 Database (o MySQL si prefieres)
- Haz clic en "Generate" para descargar el proyecto y descomprime el archivo.

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

**Definir el Modelo (Entidad JPA)**

Crea una entidad JPA para representar una tabla en la base de datos. Por ejemplo, una entidad Producto:

```java
@Entity
public class Producto {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private double precio;
    private int cantidad;

    // Getters y Setters
}
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

### Crear el Repositorio

Crea una interfaz que extienda JpaRepository para realizar las operaciones CRUD:

```
public interface ProductoRepository extends JpaRepository<Producto, Long> {
}
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

Implementa la lógica de negocio en un servicio:

```java
@Service
public class ProductoService {
    @Autowired
    private ProductoRepository productoRepository;

    public List<Producto> obtenerTodos() {
        return productoRepository.findAll();
    }
    public Producto guardarProducto(Producto producto) {
        return productoRepository.save(producto);
    }
    public Producto obtenerPorId(Long id) {
        return productoRepository.findById(id).orElse(null);
    }
    public void eliminarProducto(Long id) {
        productoRepository.deleteById(id);
    }
}
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

Crear Controladores REST

Desarrolla controladores REST para manejar solicitudes HTTP:

```java
@RestController
@RequestMapping("/api/productos")
public class ProductoController {
    @Autowired
    private ProductoService productoService;
    @GetMapping
    public List<Producto> listarProductos() {
        return productoService.obtenerTodos();
    }
    @PostMapping
    public Producto crearProducto(@RequestBody Producto producto) {
        return productoService.guardarProducto(producto);
    }
    @GetMapping("/{id}")
    public Producto obtenerProducto(@PathVariable Long id) {
        return productoService.obtenerPorId(id);
    }
    @DeleteMapping("/{id}")
    public void eliminarProducto(@PathVariable Long id) {
        productoService.eliminarProducto(id);
    }
}
```

- # ¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?

## Configurar la Base de Datos

Configura la base de datos en application.properties o application.yml:

´´´*properties*

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

**Si usamos MySQL**:

´´´*properties*

```
spring.datasource.url=jdbc:mysql://localhost:3306/nombre_bd
spring.datasource.username=root
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

´´´

**Paso 2: Crear el Frontend con React.js**

1. Configurar el Proyecto
   - Usa Create React App para iniciar un proyecto:

```bash
npx create-react-app producto-crud
cd producto-crud
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

Instala Axios para las solicitudes HTTP:

```bash
npm install axios
```

## Crear Componentes

Crea componentes para listar, crear, editar y eliminar productos. Componente ProductoList.js:

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

ProductoList.js

```jsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';
const ProductoList = () => {
    const [productos, setProductos] = useState([]);
    useEffect(() => {
        axios.get('/api/productos')
            .then(response => setProductos(response.data))
            .catch(error => console.log(error));
    }, []);
    return (
        <div>
            <h2>Lista de Productos</h2>
            <ul>
                {productos.map(producto => (
                    <li key={producto.id}>{producto.nombre} - ${producto.precio}</li>
                ))}
            </ul>
        </div>
    );
};
export default ProductoList;
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

Componente Crear Producto.js:

```jsx
import React, { useState } from 'react';
import axios from 'axios';
const CrearProducto = () => {
    const [nombre, setNombre] = useState('');
    const [precio, setPrecio] = useState('');
    const [cantidad, setCantidad] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        const producto = { nombre, precio, cantidad };
        axios.post('/api/productos', producto)
            .then(response => console.log(response.data))
            .catch(error => console.log(error));
    };
    return (
        <form onSubmit={handleSubmit}>
            <input type="text" placeholder="Nombre" value={nombre} onChange={(e) => setNombre(e.target.value)} />
            <input type="number" placeholder="Precio" value={precio} onChange={(e) => setPrecio(e.target.value)} />
            <input type="number" placeholder="Cantidad" value={cantidad} onChange={(e) => setCantidad(e.target.value)} />
            <button type="submit">Crear Producto</button>
        </form>
    );
};
export default CrearProducto;
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

## Enrutamiento

Instala react-router-dom:

```bash
npm install react-router-dom
```

```
export default App;

import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import ProductoList from './ProductoList';
import CrearProducto from './CrearProducto';

const App = () => {
    return (
        <Router>
            <Switch>
                <Route path="/" exact component={ProductoList} />
                <Route path="/crear" component={CrearProducto} />
            </Switch>
        </Router>
    );
};

export default App;
```

Configura el enrutamiento en App.js:

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

**Paso 3: Integración y Despliegue**

1. CORS
    - Configura CORS en el backend para permitir solicitudes desde React:

```java
@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("http://localhost:3000")
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS");
    }
}
```

- **¿Cómo crear una aplicación web completa con CRUD utilizando Spring Boot (backend) y React.js (frontend)?**

### Empaquetado y Despliegue

- Empaqueta y despliega el backend en un servidor (Heroku, AWS).
- Despliega el frontend en Netlify o GitHub Pages.

### 3. Pruebas

Verificar que el frontend pueda realizar todas las operaciones CRUD con el backend.

03

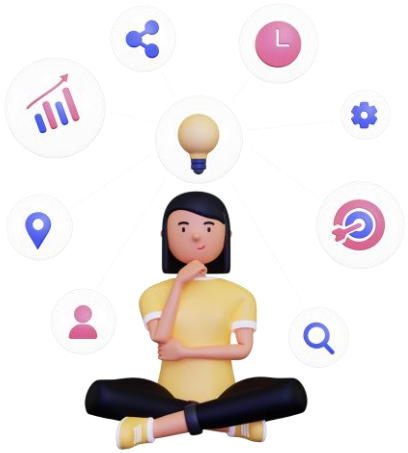
# Actividad 3.2.2

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

# Actividad 3.2.2

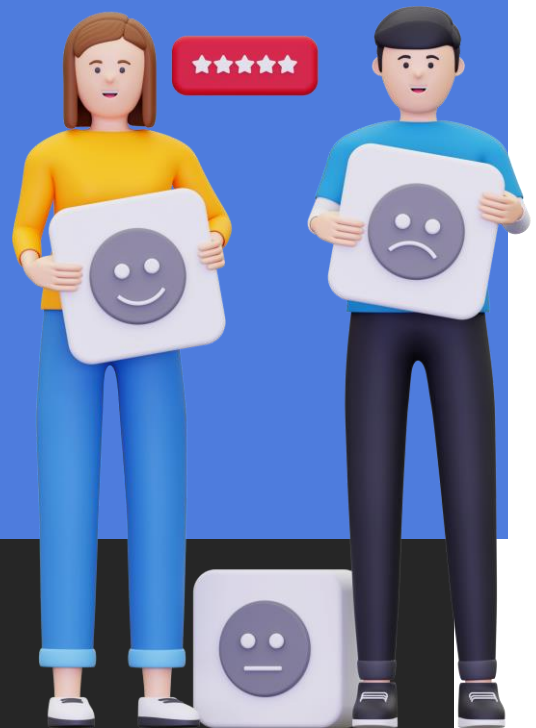
Ingresa al AVA de esta Actividad y desarrolla la actividad descrita en la guía 3.2.2 Actividad Individual Implementación de API RESTful

Consulta con tu docente las dudas que tengas al desarrollar la actividad.



DuocUC®

# ¿Estás de acuerdo con estas analogías?



## Sistema de correo postal

El frontend es como el remitente que escribe una carta (solicitud HTTP).

El backend es como la oficina de correos que procesa y entrega la carta (servidor que maneja la solicitud).

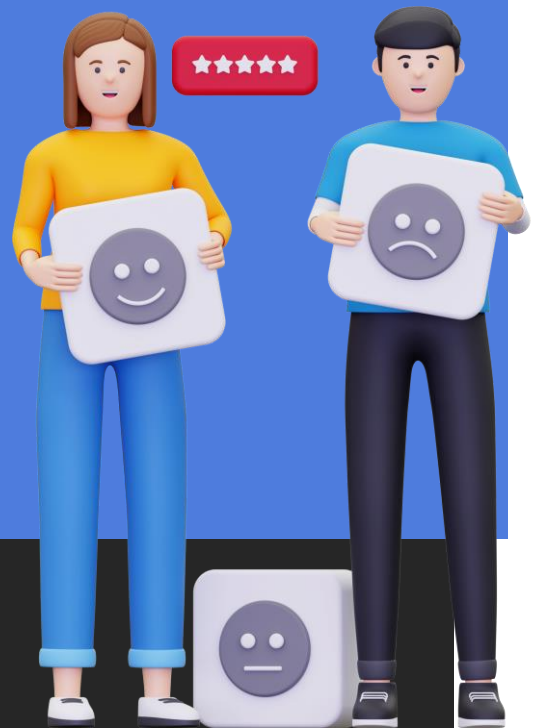
Los endpoints REST son como las direcciones de los destinatarios.

Los métodos HTTP (GET, POST, PUT, DELETE) son como los diferentes tipos de servicios postales (carta regular, paquete, correo certificado).

La respuesta del servidor es como la carta de respuesta que recibe el remitente.

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

# ¿Estás de acuerdo con estas analogías?



## Restaurante

El frontend es como el cliente que hace un pedido usando el menú.

El backend es como la cocina que prepara los platos.

Los endpoints REST son como los diferentes platos del menú.

Los métodos HTTP son como las acciones que puede realizar el cliente (pedir, modificar, cancelar).

La respuesta del servidor es como el plato que se sirve al cliente.

# • Bibliografía


**Libros Digitales Biblioteca Duoc. (Con tu cuenta de Duoc puedes consultar)**

- Larsson, M. (2023). Microservices with Spring Boot 3 and Spring Cloud: Build resilient and scalable microservices using Spring Cloud, Istio, and Kubernetes (3a ed.). Packt Publishing.

**Recursos de información.**

- Spring Boot: https://docs.spring.io/spring-boot/index.html
- Enabling Cross Origin Requests for a RESTful Web Service: https://spring.io/guides/gs/rest-service-cors
- Netlify Documentation: https://docs.netlify.com/

# Duoc UC®

CERCANÍA. LIDERAZGO. FUTURO.

duoc.cl

7 AÑOS ACREDITADO

Comisión Nacional de Acreditación CNA-Chile

DESDE AGOSTO 2017 HASTA AGOSTO 2024.
DOCENCIA DE PREGRADO. GESTIÓN INSTITUCIONAL. VINCULACIÓN CON EL MEDIO.