Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Vacaciones Diciembre 2025

PROYECTO 2 SIMULACIÓN DE SISTEMA DE PROCESAMIENTO DE PAQUETERÍA LOGITRACK S.A, CLIENTE SERVIDOR UTILIZANDO SPRING BOOT JAVA Y VITE REACT

201807484 - Marck Dilan Orantes García
202341770 - Carlos Eduardo Reyes Villela

Resumen

La implementación de estos dos campos cliente servidor se basa en que un sistema controla parte visible de la página web que es el front end y el otro la parte lógica llamado backend.

Existen varias formas de implementar un cliente servidor con varios tipos de lenguajes de programación en esta ocasión se utilizó React como frontend que utiliza lenguaje javascript, html y css , para la parte de back end se utilizó Spring boot que utiliza java 25.

En este análisis se evidencia la forma que ambos sistemas se enlazan para intercambiar información para poder realizar distintos cambios de datos o edición de los mismos.

Palabras clave

Frontend: Es la parte en donde el usuario va a interactuar con una interfaz creada anteriormente que pide información al backend.

Backend: Es la infraestructura que hace de servidor lógico en el sistema cliente servidor, maneja los datos delicados del sistema.

Abstract

The implementation of this client server architecture is based on one system controlling the visible part of the website, the front end, and the other controlling the logical part, the back end.

There are many ways to implement a client server architecture using various programming languages. In this case, React was used as the front end, utilizing JavaScript, HTML, and CSS. Spring Boot, and the back end uses Java 25.

This analysis demonstrates how both systems are linked to exchange information, enabling various data changes and edits between.

Keywords

Frontend: This is the part where the user interacts with a previously created interface that requests information from the backend.

Backend: This is the infrastructure that acts as the logical server in the client-server system, handling the system's sensitive data.

Introducción

Cuando se desarrolla aplicaciones web normalmente la combinación de utilización de Java Spring Boot y vite React es una opción muy usadas para crear muchas funciones y que sean escalables, Java Spring boot es un framework de desarrollo web que facilita los manejos de respuestas y recibimiento de preguntas hacia el servidor.

Y react es un framework de desarrollo de interfaces web que permite a que el programador cree interfaces rápidamente y darles estilo con sus implementaciones de CSS.

Para este proyectó se utilizaron los frameworks mencionados para crear un sistema cliente servidor donde se manejan varias listas nativas de java , donde el frontend muestra las listas de los objetos creados a partir de que el backend lea un archivo mandado desde la interfaz de la página web

Desarrollo del tema

Modelo cliente servidor:

El propósito es mantener la comunicación de un equipo con uno que es el servidor principal en donde se almacena toda la información y la lógica de la página o del programa en cuestión, entonces tiene como objetivo que varios usuarios se conecten al cliente que será la parte del frontend y estos usuarios no podrán ver nada de la lógica del sistema porque las respuestas que se muestran en el frontend página principal son manejadas por el servidor backend de la empresa. Para este ensayo se creó un cliente y un servidor como parte del mismo computador, pero cada sistema maneja sus rutas que serán manejadas por los endpoints del servidor.

Un sistema cliente servidor está conformado por:

Protocolo: Un protocolo es la agrupación de normas o reglas que establecen de manera clara el flujo de información de la red.

Servicios: El servicio es la información que buscar responder a las peticiones del cliente frontend pueden ser llamad endpoints

Los endpoints son las formas que tienen los sistemas clientes servidor para poder intercambiar información ya sea de cliente a servidor por medio de gets o cambiar algo en el servidor por medio de un endpoint put.

Base de datos: Son Conjunto de información ordenada y categorizada según la conveniencia del servidor, forma parte de la red cliente servidor , si el frontend se conecta con el backend este último se conecta con el servidor para hacer peticiones de datos específico, para este proyecto se manejó toda la información desde la memoria dinámica y no se requirió el uso de bases de datos, lo que quiere decir que no hay persistencia de datos una vez se termine de ejecutar el Servidor Backend

Spring boot JAVA:

Cuando se crea un proyecto Spring boot este crea un archivo de configuración en donde se definen las dependencias necesarias para poner a correr el proyecto , y cuando este arranca toma un archivo de configuración en donde se encuentra la dirección del servidor y el puerto que se utilizará para intercambiar información.

Funcionalidad De React:

React está basado en concepto de componentes que representan bloques de código y estos a su vez se visualizan como interfaz para el usuario. Cada uno de estos componentes tiene estados y formas de renderizarse según el contenido.

React utiliza la api fetch para enviar solicitudes al backend por medio de enlaces http.

a. Utilización de la página para el usuario

Al iniciar la página se encontrará con una página principal en donde se podrá cargar un archivo xml.

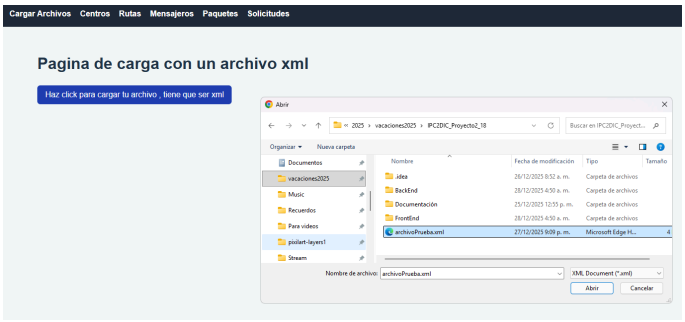


Figura 1. Menú principal Subida de archivo xml.

Fuente: elaboración propia.

Luego la pagina le dirá si se ha cargado correctamente el archivo

Figura 2. Menú principal Subida correcta

Fuente: elaboración propia.

El usuario ahora podrá dirigirse a las páginas de listar los objetos leidos del archivo xml con los botones superiores de la página principal

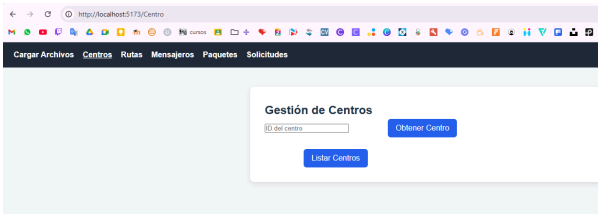


Figura 3. Página gestion de centros

Fuente: elaboración propia.

Para que la página muestre los centros que están cargados en el backend tiene que darle clik al botón de listar Centro y se desplegarán todos los centros con sus atributos

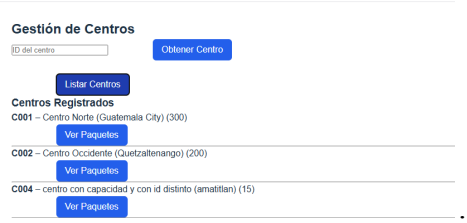


Figura 4. Pagina Centros mostrar los centros cargados

Fuente: elaboración propia.

Si el usuario necesita más información acerca del centro en cuestión puede colocar el id del centro a buscar en la parte Gestion de Centros y darle click en obtener centro para desplegar su información

Figura 5. Pagina Centros mostrar datos de un centro específico

Fuente: elaboración propia.

La sección de rutas contiene un botón para listar las rutas actuales y luego poder editar cada ruta individualmente , o crear una ruta nueva con los campos a escribir que tiene en la parte superior y luego hacer click en el botón Crear Ruta,

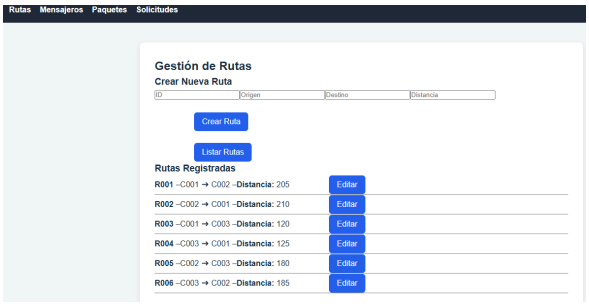


Figura 6. Página Rutas

Fuente: elaboración propia.

Cuando se hace clic en editar una ruta en especifico se abre un submenú que ya rellena con los datos actuales de la ruta, ahí el usuario puede los parámetros de la ruta y si comete algún error se indica cual fue, o puede cancelar la edición y no se harán cambios. Si el usuario está conforme tiene que dar click en el botón de Guardar Cambios. y regresará automáticamente a la lista de centros.

Figura 7. Página Rutas editar ruta especifica

Fuente: elaboración propia

Cuando se hace clic en eliminar primero se debe de colocar el id de la ruta que se quiera eliminar y se eliminará automáticamente de la lista.

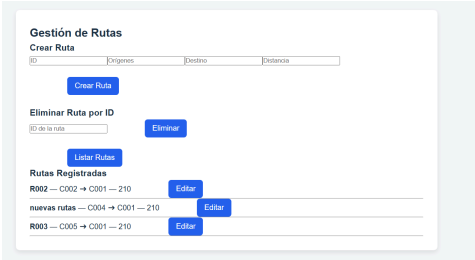


Figura 8. Eliminación de rutas

Fuente: elaboración propia

Si el usuario quisiera eliminar una solicitud debe de ingresar el ID de la solicitud que desea eliminar y hacer clic en el botón eliminar que está dentro del menú de solicitudes.

Figura 9. Página Solicitudes Eliminar solicitud

Fuente: elaboración propia

b. Documentación para el desarrollador

Backend:

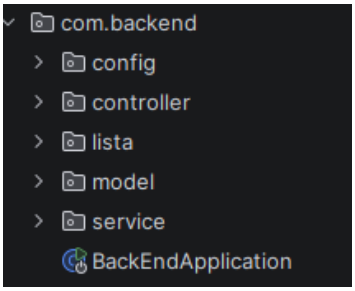


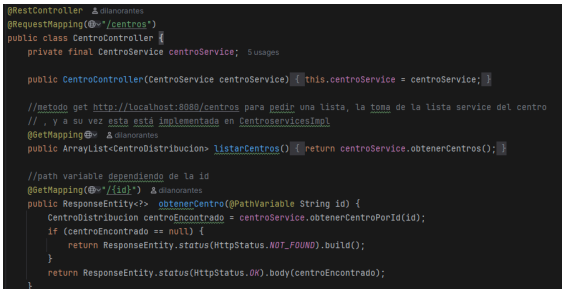
Figura 11. Clases y carpetas del programa.

Fuente: elaboración propia.

Las clases son importantes para un correcto orden de los objetos y el manejo de los controladores que se utilizan funciones Post, Get para intercambiar información con el frontend.

Para el backend donde se está utilizando Spring Boot java 25 , está separado por los modelos que son las clases de los objetos como los centros, mensajeros y paquetes , estos servirán para crear objetos correspondientes y luego guardarlos en una lista global en la carpeta lista.

La carpeta de Service manejara la mayoría de la lógica del backend, ya que en esta se encuentra funciones lógicas para crear o editar los objetos necesarios que luego se llamarán en los controllers correspondientes.



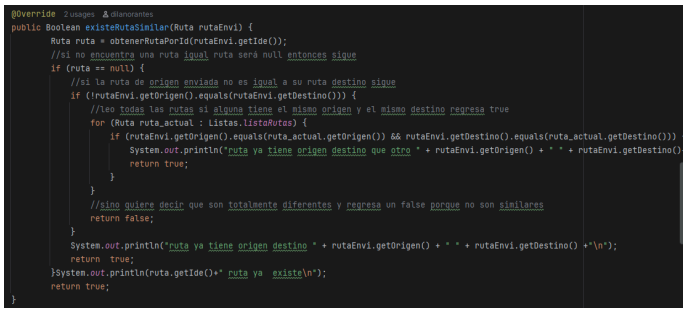
```
@RestController  & dianorantes
@RequestMapping(@v"/centros")
public class CentroController {
    private final CentroService centroService;  5 usages

    public CentroController(CentroService centroService) { this.centroService = centroService; }

    //metodo get http://localhost:8080/centros para pedir una lista, la toma de la lista service del centro
    // , y a su vez esta está implementada en centroserviceslmpl
    @GetMapping@v  & dianorantes
    public ArrayList<CentroDistribucion> listarCentros() { return centroService.obtenerCentros(); }

    //path variable dependiendo de la id
    @GetMapping(@v"/{id}")  & dianorantes
    public ResponseEntity<?>  obtenerCentro(@PathVariable String id) {
        CentroDistribucion centroEncontrado = centroService.obtenerCentroPorId(id);
        if (centroEncontrado == null) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
        return ResponseEntity.status(HttpStatus.OK).body(centroEncontrado);
    }
```

Figura 12. Controller de los centros

Fuente: elaboración propia.

Los controllers son base importante para intercambiar información con el frontend, estos intercambian respuestas u objetos json dependiendo si se necesitan listas de objetos o respuestas por cambios no correctos, para manejar estas respuestas se utilizó Response Entity que es una importación de una clase de java Spring boot y funciona para devolver una respuesta http junto con un body o cuerpo o un estado.

También se utilizan las anotaciones de request o gets para indicar que tipo de intercambio de información es y cual es la dirección o ruta que necesita el frontend para intercambiar información con el backend.



```
@Override  2 usages  & dianorantes
public Boolean existeRutaSimilar(Ruta rutaEnvi) {
        Ruta ruta = obtenerRutaPorId(rutaEnvi.getIde());
        //si no encuentra una ruta igual ruta será null entonces sigue
        if (ruta == null) {
            //si la ruta de origen enviada no es igual a su ruta destino sigue
            if (!rutaEnvi.getOrigen().equals(rutaEnvi.getDestino())) {
                //leo todas las rutas si alguna tiene el mismo origen y el mismo destino regresa true
                for (Ruta ruta_actual : Listas.listaRutas) {
                    if (rutaEnvi.getOrigen().equals(ruta_actual.getOrigen()) && rutaEnvi.getDestino().equals(ruta_actual.getDestino())
                        System.out.println("ruta ya tiene origen destino que otro " + rutaEnvi.getOrigen() + " " + rutaEnvi.getDestino()
                        return true;
                    }
                }
                //sino quiere decir que son totalmente diferentes y regresa un false porque no son similares
                return false;
            }
            System.out.println("ruta ya tiene origen destino " + rutaEnvi.getOrigen() + " " + rutaEnvi.getDestino() +"\n");
            return  true;
        }System.out.println(ruta.getIde()+" ruta ya  existe\n");
        return true;
}
```

Figura 13. archivo RutaServiceImpl función para verificar rutas

Fuente: elaboración propia.

Los archivo Service como ya se indicó manejan la lógica del programa, en este caso la función verifica si existe una ruta similar en la lista de rutas si es asi devuelve un verdadero.

Si se crea un objeto tipo RutaService se puede utilizar para que llame a las funciones que tiene RutaServiceImpl y verificar errores en otros modelos Service o en los mismos controllers para manejar y enviar errores correspondientes al frontend.

Frontend:

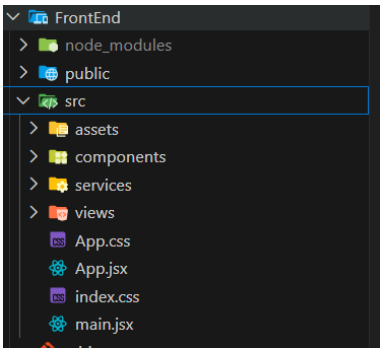


Figura 14. Clases y carpetas del programa.

Fuente: elaboración propia.

De parte de frontend se trabajó de forma ordenada separando las views de los services y los componentes que serán utilizados para trabajar el estilo de la página web.

Figura 15. código de programa.

Fuente: elaboración propia.

Dentro de cada view se encontraran iniciadas de primer lugar las constantes que serán necesarias para la utilización de nuestro backend en nuestro frontend cada una es llamada para su utilización en el código posteriormente.

```
//se llama el metodo de listar los centros del backend
const listarCentros = async () => {
  try {
    setLoading(true)
    setError(null)

    const response = await fetch('http://localhost:8080/centros')
    const data = await response.json()

    setCentros(data)
    setMostrarLista(true)
  } catch (err) {
    setError('No se pudieron cargar los centros')
  } finally {
    setLoading(false)
  }
}
```

Figura 16. código de programa.

Fuente: elaboración propia.

Conclusiones

El uso combinado de Java spring boot y vite React es una opción recomendable para desarrollar aplicaciones web escalables, porque Spring boot facilita el manejo de respuestas y recibimiento de solicitudes para el servidor y el Cliente React permite crear interfaces de usuario de una manera asequible.

Para este proyecto se utilizó la arquitectura cliente servidor para crear un sistema que maneja listas nativas de java y el frontend puede mandar un archivo xml al servidor por medio de solicitudes http para que la lógica del backend maneje qué objetos crear y cuáles no.

Referencias bibliográficas

Meric, A. (2024). Mastering Spring Boot 3.0: A comprehensive guide to building scalable and efficient backend systems with Java and Spring. Packt Publishing Ltd.

Meza, R. E. M. (s.f.). *Modelo Cliente-Servidor.*

Oliveira, C. E. d., Turnquist, G. L., Antonov, A. (2018). *Developing Java Applications with Spring and Spring Boot: Practical Spring and Spring Boot Solutions for Building Effective Applications*. Reino Unido: Packt Publishing.

Schiaffarino (2019). *Modelo cliente servidor. Infranetworking.*

Apéndice

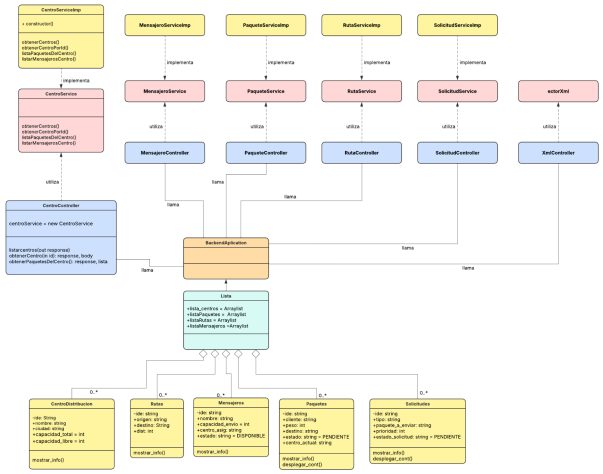


Figura 11. Diagrama de clases Backend.

Fuente: elaboración propia.

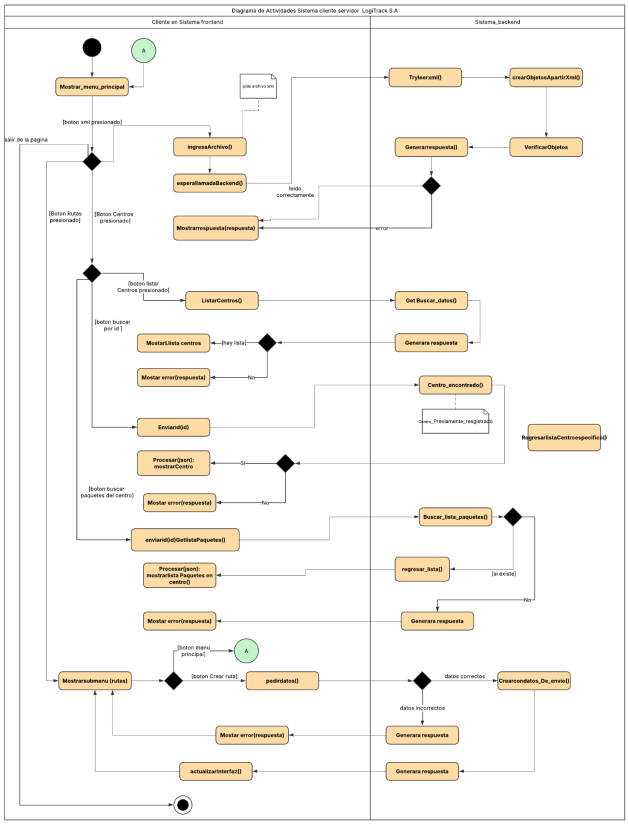


Figura 12. Diagrama de actividades Flujo del sistema cliente servidor

Fuente: elaboración propia.