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

PROYECTO 2

202401198 – José Bernardo Méndez Juárez
202405532 – Marcos Manuel Chip Castillo

Resumen

El presente ensayo analiza el diseño y la implementación de un sistema de gestión de envíos. El sistema modela la operación logística de centros de distribución, rutas, mensajeros, paquetes y solicitudes, integrando estructuras de datos, reglas de negocio y servicios REST. Su relevancia radica en la simulación de problemas reales de logística y transporte, ampliamente presentes en contextos nacionales e internacionales, donde la optimización de recursos y el control del estado de los envíos resultan críticos. Se abordan posturas técnicas relacionadas con la arquitectura por capas, el uso de colas de prioridad para la gestión de solicitudes y la validación estricta de estados sin el uso de enumeraciones, privilegiando la comprensión de la lógica interna. El impacto del sistema se refleja a nivel técnico, al fomentar buenas prácticas de diseño, y a nivel académico, al fortalecer habilidades de análisis, modelado y desarrollo backend. Finalmente, se concluye que una correcta separación de responsabilidades y validación de reglas de negocio es clave para construir sistemas robustos y escalables.

Palabras clave

Frontend, BackEnd, SpringBoot, React, Distribucion.

Abstract

This essay analyzes the design and implementation of a shipment management. The system models the logistics operation of distribution centers, routes, couriers, packages, and requests, integrating data structures, business rules, and REST services. Its relevance lies in simulating real-world logistics problems commonly found in national and international contexts, where resource optimization and shipment state control are critical. Technical approaches related to layered architecture, priority queues for request management, and strict state validation without using enumerations are discussed, emphasizing the understanding of internal logic. The system's impact is reflected at a technical level, by promoting good design practices, and at an academic level, by strengthening analysis, modeling, and backend development skills. It is concluded that proper separation of responsibilities and validation of business rules are essential to build robust and scalable systems.

Keywords

Frontend, BackEnd, SpringBoot, React, Distribution.

Introducción

En la actualidad, el desarrollo de software moderno se basa en arquitecturas distribuidas que separan claramente la lógica de negocio de la interfaz de usuario. En este contexto, las aplicaciones web suelen dividirse en dos componentes fundamentales: frontend y backend, los cuales se comunican a través de servicios web.

El presente proyecto consiste en el desarrollo de un sistema de gestión logística compuesto por un backend desarrollado en Spring Boot y un frontend desarrollado en React. El sistema permite administrar varios centros a la vez.

El backend expone una API REST, encargada de procesar la lógica del sistema, validar las reglas del negocio y manejar el estado interno de las entidades. Por su parte, el frontend actúa como cliente de esta API, proporcionando una interfaz gráfica que permite al usuario interactuar con el sistema de forma intuitiva y dinámica.

Desarrollo del tema

a. Arquitectura del sistema

El sistema fue diseñado siguiendo una arquitectura por capas, un enfoque ampliamente utilizado en aplicaciones empresariales y sistemas web modernos. Esta arquitectura permite separar responsabilidades, mejorar la mantenibilidad del código y facilitar la evolución del sistema a largo plazo. En el proyecto, la arquitectura se compone principalmente de tres capas: controladores, servicios y modelos, cada una con una función claramente definida.

La capa de controladores es la encargada de exponer los endpoints de la API REST. Estos controladores reciben las peticiones HTTP provenientes del frontend desarrollado en React, procesan los datos de entrada en formato JSON y delegan la lógica de negocio a la capa de servicios. De esta forma, los controladores actúan como intermediarios entre el cliente y el núcleo del sistema, evitando que el frontend tenga acceso directo a los datos internos.

La capa de servicios concentra la lógica de negocio del sistema. En esta capa se implementan las reglas que gobiernan el comportamiento de las entidades, como la validación de estados, la asignación de mensajeros, el procesamiento de solicitudes y la verificación de rutas entre centros. Esta separación es fundamental, ya que permite que las reglas del negocio se mantengan independientes de la forma en que se presentan o consumen los datos. Además, facilita la reutilización del backend con diferentes interfaces, ya sea una aplicación web, móvil o incluso pruebas automatizadas.

La capa de modelos define las estructuras de datos principales del sistema. Clases como Centro, Mensajero, Paquete, Ruta y Solicitud representan los objetos del dominio del problema. Cada modelo encapsula atributos y métodos básicos, permitiendo trabajar con los datos de forma estructurada y coherente. Este enfoque refuerza el uso de la programación orientada a objetos, promoviendo principios como encapsulación y abstracción.

Desde el punto de vista de integración, esta arquitectura facilita la comunicación entre el frontend y el backend mediante el uso de servicios REST, permitiendo una clara separación entre la interfaz de usuario y la lógica del sistema. Gracias a esto, el desarrollo del frontend puede realizarse de forma paralela al backend, optimizando el trabajo colaborativo y reduciendo dependencias entre equipos.

---

b. Gestión de solicitudes mediante cola de prioridad

Uno de los componentes más importantes del sistema es la gestión de solicitudes de envío, la cual se implementa utilizando una cola de prioridad. Cada solicitud posee un nivel de prioridad numérico que determina el orden en el que será atendida. Este mecanismo permite simular escenarios reales en los que ciertos envíos requieren atención preferente, como entregas urgentes o paquetes con mayor importancia operativa.

La cola de prioridad se mantiene en memoria utilizando una estructura de datos que ordena las solicitudes de forma descendente según su prioridad. Cada vez que se agrega una nueva solicitud al sistema, esta se inserta en la cola y se reordena automáticamente para garantizar que la solicitud con mayor prioridad se procese primero. Este comportamiento es clave para asegurar una asignación eficiente de los recursos disponibles.

El procesamiento de solicitudes se realiza mediante endpoints específicos que permiten atender una solicitud o múltiples solicitudes en una sola operación. Durante este proceso, el sistema evalúa diversas condiciones antes de completar la operación, como la existencia del paquete, su estado actual, la disponibilidad de mensajeros y la capacidad de carga de los mismos.

Este enfoque no solo demuestra la aplicación práctica de estructuras de datos vistas en el curso, sino que también evidencia cómo estas estructuras pueden integrarse en sistemas reales para resolver problemas complejos. La cola de prioridad permite optimizar la toma de decisiones del sistema y refleja cómo la teoría aprendida en clase se traduce en soluciones funcionales dentro de un contexto realista.

---

c. Manejo de estados y validaciones

El manejo de estados y la aplicación de validaciones estrictas constituyen uno de los pilares fundamentales del sistema. Cada entidad relevante, como los paquetes y los mensajeros, posee un estado que define su situación actual dentro del flujo del sistema. En el caso de los paquetes, los estados permitidos son PENDIENTE, EN_TRANSITO y ENTREGADO, mientras que los mensajeros pueden encontrarse DISPONIBLES o EN_TRANSITO.

El sistema define claramente las transiciones válidas entre estados, evitando situaciones inconsistentes. Por ejemplo, un paquete solo puede pasar de PENDIENTE a EN_TRANSITO, y posteriormente a ENTREGADO. No se permite revertir un estado una vez que ha avanzado, lo cual garantiza la integridad del proceso de envío. De forma similar, un mensajero no puede ser reasignado a otro centro si se encuentra en tránsito, ya que esto representaría una inconsistencia lógica.

Todas estas validaciones se implementan en la capa de servicios, lo que asegura que ningún cambio de estado pueda realizarse sin pasar por las reglas del negocio, incluso si el frontend intenta enviar datos incorrectos. Este enfoque refuerza la seguridad y consistencia del sistema, ya que el backend actúa como la única fuente de verdad.

Adicionalmente, el sistema registra internamente los cambios de estado relevantes, como el momento en que un paquete es marcado como entregado. Este registro permite llevar un control del historial de operaciones y facilita futuras extensiones del sistema, como la generación de reportes o auditorías.

---

d. Impacto académico y técnico

Desde el punto de vista académico, este proyecto representa una aplicación integral de los conceptos fundamentales abordados en el curso de IPC2. Se refuerzan principios de programación orientada a objetos, como el uso de clases, encapsulación y responsabilidad única, así como la implementación de estructuras de datos para resolver problemas reales.

Asimismo, el proyecto permite comprender la importancia del diseño de software y la correcta organización del código. La separación en capas, el uso de servicios y la validación de reglas de negocio demuestran cómo un diseño adecuado facilita el mantenimiento, la escalabilidad y la comprensión del sistema.

Desde una perspectiva técnica, el sistema evidencia cómo una arquitectura bien definida y reglas de negocio claras contribuyen a la robustez de una aplicación backend. La integración con un frontend desarrollado en React muestra cómo los sistemas modernos dependen de una comunicación eficiente entre cliente y servidor, utilizando estándares como HTTP y JSON.

En conjunto, el proyecto no solo cumple con los objetivos académicos del curso, sino que también simula un entorno de desarrollo real, preparando al

estudiante para enfrentar desafíos propios del desarrollo de software profesional.

Conclusiones

El desarrollo del sistema de gestión de envíos permitió aplicar de manera práctica los conocimientos adquiridos durante el curso, integrando conceptos teóricos con una implementación funcional de un sistema backend y su consumo desde un frontend. A lo largo del proyecto, se evidenció la importancia de una correcta planificación y diseño de la arquitectura para garantizar la coherencia y escalabilidad del software.

La utilización de una arquitectura por capas facilitó la separación de responsabilidades entre controladores, servicios y modelos, lo cual permitió un código más organizado, legible y fácil de mantener. Esta estructura resultó fundamental para implementar reglas de negocio complejas sin afectar la comunicación con el frontend, demostrando la relevancia de un diseño adecuado en aplicaciones reales.

La implementación de una cola de prioridad para la gestión de solicitudes destacó el valor de las estructuras de datos dentro de un contexto práctico. El uso de prioridades permitió optimizar el procesamiento de envíos y simular escenarios reales donde ciertos paquetes requieren atención preferente, fortaleciendo la comprensión del impacto de las decisiones algorítmicas en el comportamiento del sistema.

El manejo estricto de estados y validaciones garantizó la consistencia lógica del sistema. Al definir transiciones válidas y restringir operaciones según el estado de paquetes y mensajeros, se evitó la generación de datos incoherentes. Este enfoque reforzó la idea de que el backend debe actuar como la autoridad principal para proteger la integridad del sistema, independientemente de las acciones realizadas desde el frontend.

Desde una perspectiva técnica, la integración entre el backend desarrollado en Spring Boot y el frontend en React evidenció la importancia de una comunicación clara mediante servicios REST. Este intercambio permitió validar el funcionamiento real del sistema y comprender cómo diferentes tecnologías pueden interactuar eficientemente dentro de una misma solución.

Finalmente, el proyecto contribuyó significativamente al crecimiento académico y profesional, ya que simuló un entorno de desarrollo cercano a la realidad laboral. La experiencia adquirida en la implementación de reglas de negocio, manejo de errores, diseño de servicios y trabajo colaborativo representa una base sólida para el desarrollo de sistemas más complejos en el futuro.

Referencias bibliográficas

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2015). *Patrones de diseño: Elementos de software orientado a objetos reutilizable*. Pearson Educación.
- Pressman, R. S., & Maxim, B. R. (2020). *Ingeniería del software: Un enfoque práctico* (9.ª ed.). McGraw-Hill.
- Sommerville, I. (2016). *Ingeniería de software* (10.ª ed.). Pearson Educación.
- Oracle Corporation. (2024). *Documentación oficial de Java Platform, Standard Edition*.