DISEÑO DE UNA SOLUCIÓN DE SOFTWARE INCORPORANDO PATRONES CREACIONALES

PATRONES DE DISEÑO DE SOFTWARE

PRESENTADO POR:

CASTILLA CRUZ ROBERT MAURICIO

OROZCO SANTANA ALEX DANIELS

ROJANO RECIO ORLANDO YESITH

TORRES CALEÑO JEAN CARLOS

PRESENTADO A:

MSC. ING. SEOANES LEON JAIRO FRANCISCO

UNIVERSIDAD POPULAR DEL CESAR

FACULTAD DE INGENIERIAS Y TECNOLOGICAS

ESPECIALIZACIÓN EN INGENIERIA DE SOFTWARE

2025-I

TABLA DE CONTENIDO

INTRODUCCIÓN

En la actualidad, los negocios con servicios de entrega a domicilio enfrentan múltiples desafíos operativos relacionados con la gestión eficiente de pedidos, el monitoreo de repartidores y la satisfacción del cliente. La creciente demanda por parte de los consumidores de soluciones ágiles, transparentes y confiables ha evidenciado la necesidad de sistemas tecnológicos que integren funcionalidades avanzadas como el rastreo en tiempo real, la automatización de notificaciones y la optimización de rutas de entrega.

Este proyecto surge como respuesta a una problemática real identificada en la ciudad de Valledupar, donde numerosos comercios carecen de herramientas que les permitan gestionar de forma efectiva sus operaciones de reparto. Con base en este contexto, se ha diseñado una solución de software web-móvil que incorpora patrones de diseño creacionales y estructurales, integrando además frameworks modernos que garantizan modularidad, escalabilidad y facilidad de mantenimiento.

La propuesta se fundamenta en una arquitectura de microservicios que permite desacoplar componentes, mejorar la resiliencia del sistema y facilitar su evolución tecnológica. Asimismo, se aplican patrones como Factory Method, Singleton, Adapter, Fa cade y Proxy, con el fin de estructurar una solución robusta, extensible y alineada con las buenas prácticas de la ingeniería de software.

OBJETIVOS

OBJETIVO GENERAL

Formular una propuesta de solución de software para la gestión de pedidos a domicilio, basada en una arquitectura de microservicios e incorporando patrones de diseño creacionales y estructurales, con el fin de mejorar la eficiencia operativa de los negocios, la experiencia del cliente y la trazabilidad del proceso de entrega.

OBJETIVOS ESPECIFICOS

- Analizar el contexto del problema que enfrentan los negocios de entrega a domicilio en Valledupar, identificando las necesidades funcionales y no funcionales del sistema propuesto.
- Estructurar los requerimientos del sistema, detallando sus funcionalidades principales y criterios de calidad conforme a la norma ISO 25000.
- Diseñar una arquitectura de software adecuada, seleccionando un enfoque de microservicios para garantizar escalabilidad, modularidad y mantenibilidad.
- Seleccionar e integrar patrones de diseño creacionales y estructurales, justificando su aplicabilidad dentro del sistema propuesto y alineándolos con las responsabilidades de cada componente.
- Documentar la propuesta técnica mediante diagramas de clases y componentes, evidenciando la distribución lógica de responsabilidades y la integración de los patrones seleccionados.

1. Análisis del escenario y definición de requerimientos

1.1 Descripción del Producto

El sistema es una aplicación web que permite a los negocios gestionar de manera eficiente sus pedidos a domicilio, brindando a los clientes la posibilidad de rastrear en tiempo real el estado y la ubicación de su pedido. La plataforma proporciona a los comercios herramientas de monitoreo de repartidores, optimización de rutas y reducción de tiempos de entrega, mejorando así la eficiencia del servicio y la experiencia del usuario.

1.2 Objetivos del Sistema

- Optimizar la gestión de pedidos a domicilio mediante un sistema que reduzca la carga administrativa de los negocios.
- Ofrecer seguimiento en tiempo real a los clientes para mejorar su experiencia y reducir la incertidumbre en la entrega.
- Brindar herramientas de monitoreo a los negocios para analizar el rendimiento de sus repartidores y optimizar rutas.
- Reducir los tiempos de entrega mediante un sistema de gestión eficiente basado en geolocalización.
- Automatizar las notificaciones de estado del pedido para minimizar la necesidad de consultas manuales por parte de los clientes.

1.3 Alcance del Producto

- Módulo de rastreo en tiempo real: Visualización en mapa de la ubicación del repartidor y el pedido.
- Gestión de pedidos: Registro, actualización y monitoreo de órdenes de entrega.
- Panel de administración para negocios: Control de pedidos, seguimiento de repartidores y reportes de eficiencia.
- Interfaz de cliente: Consulta de estado del pedido y notificaciones en vivo.
- Optimización de rutas: Recomendaciones de trayectos más eficientes para los repartidores.
- Soporte multiplataforma: Accesible desde computadoras, tablets y smartphones.

1.4 Valor del Producto

- Mayor eficiencia en la entrega: Reducción de tiempos de espera y optimización de rutas de reparto.
- Mejor experiencia del cliente: Transparencia en la entrega y comunicación automatizada del estado del pedido.
- Reducción de carga operativa: Menos llamadas y mensajes de clientes preguntando por el estado de sus pedidos.
- Mayor control para los negocios: Datos analíticos sobre tiempos de entrega y desempeño de repartidores.
- Mayor confianza en el servicio: Un sistema transparente que genera mayor fidelización de clientes.

1.5 Público Objetivo

- Negocios con servicio a domicilio: Restaurantes, supermercados, farmacias, tiendas minoristas.
- Clientes que usan servicios de entrega: Personas que buscan comodidad y rapidez en la recepción de sus pedidos.
- Repartidores: Conductores de entrega que requieren rutas optimizadas y comunicación eficiente con clientes y comercios.

1.6 Uso Previsto

- Los clientes ingresan a la aplicación web y consultan en tiempo real la ubicación de su pedido.
- Los negocios gestionan sus pedidos y pueden monitorear a los repartidores en un panel de administración.
- El sistema envía notificaciones automáticas a los clientes sobre el estado de su pedido.
- Los repartidores utilizan la aplicación para recibir instrucciones de entrega y seguir rutas optimizadas.
- Se generan reportes de desempeño para mejorar la eficiencia de las entregas en el futuro.

1.7 Elicitación de requisitos

1.7.1 Definición de la visión

Para las empresas de pedidos a domicilios en la ciudad de Valledupar quienes tienen la necesidad de conocer el estado de sus pedidos el sistema web para la gestión y seguimiento de entregas a domicilio en Valledupar que es un producto que proporciona a los comercios herramientas de monitoreo de repartidores, optimización de rutas y reducción de tiempos de entrega, a diferencia de la competencia que solo ofrece un contacto entre el usuario, empresa y el domiciliario, nuestro producto permite a las empresas saber que tan optimos son sus tiempos de entrega ya que permite rastrear a sus domiciliarios con información en tiempo real (usuario-empresa).

1.7.2 Identificación del problema

En la ciudad de Valledupar, muchos negocios enfrentan dificultades en la gestión eficiente de sus pedidos a domicilio. La constante necesidad de responder a los clientes sobre el estado y la ubicación de sus pedidos genera una carga adicional para los comercios, que deben desviar tiempo y recursos en la atención de estas consultas. Este proceso no solo ralentiza el servicio, sino que también afecta la experiencia del cliente, que muchas veces se siente frustrado por la falta de información o la demora en la entrega.

Por otro lado, los negocios carecen de herramientas efectivas para monitorear en tiempo real la ubicación de sus domiciliarios y medir la eficiencia de sus operaciones de reparto, lo que puede generar retrasos, rutas ineficientes y dificultades para optimizar el servicio.

Ante esta situación, se hace evidente la necesidad de un sistema que permita a los clientes rastrear sus pedidos de forma autónoma, con información en tiempo real sobre su ubicación y estado, y que al mismo tiempo proporcione a los negocios datos clave sobre el rendimiento de sus repartidores. Un aplicativo movil-web que integre el uso de GPS para el seguimiento de los pedidos y los domiciliarios podría mejorar la transparencia del servicio, reducir la carga administrativa de los negocios y optimizar las rutas de entrega, resultando en una mejor experiencia tanto para los clientes como para los comercios.

1.7.3 Definición de Requisitos funcionales

ID	*RF01*
Nombre	*Gestión de usuarios*
Descripción	*El sistema debe permitir gestionar a los usuarios.*
Importancia	*Mantener el control de los usuarios (CRUD)*
Prioridad	*Alta*

ID	*RF02*
Nombre	*Crear de pedidos*
Descripción	*El sistema debe permitir a los usuarios hacer un pedido.*
Importancia	*Para poder recibir su producto.*
Prioridad	*Alta*

ID	*RF03*
Nombre	*Asignación de pedidos*
Descripción	*El sistema debe permitir a los recepcionistas asignar un pedido.*
Importancia	*Para poder gestionar la entrega de sus pedidos.*
Prioridad	*Alta*

ID	*RF04*
Nombre	*Rastreo de domiciliario*

Descripción	*El sistema debe permitir a los recepcionistas conocer la ubicación de sus domiciliarios.*
Importancia	*Para poder elegir el domiciliario mas cercano.*
Prioridad	*Alta*

ID	*RF05*
Nombre	*Rastreo de pedido*
Descripción	*El sistema debe permitir a los usuarios rastrear su pedido.*
Importancia	*Para saber el estado y donde se encuentra su pedido.*
Prioridad	*Alta*

ID	*RF06*
Nombre	*Información de rutas*
Descripción	*El sistema debe mostrar a los domiciliarios la ruta mas optima.*
Importancia	*Para reducir los tiempos de entrega de pedidos.*
Prioridad	*Alta*

ID	*RF07*
Nombre	*Comunicación de usuarios-domiciliarios*
Descripción	*El sistema debe permitir la comunicación entre usuarios y domiciliarios.*
Importancia	*Para resolver o aclarar dudas sobre la entrega del pedido.*

Prioridad	*Alta*

ID	*RF08*
Nombre	*Comunicación de recepcionistas-domiciliarios*
Descripción	*El sistema debe permitir la comunicación entre recepcionistas y domiciliarios.*
Importancia	*Para resolver o aclarar dudas sobre la entrega o asignación del pedido.*
Prioridad	*Alta*

ID	*RF09*
Nombre	*Datos estadisticos*
Descripción	*El sistema debe mostrar a los dueños de comercios los datos de sus domiciliarios.*
Importancia	*Para determinar la eficiencia de sus domiciliarios.*
Prioridad	*Alta*

1.7.4 Definición de Requisitos No funcionales

ID	*RNF01*
Nombre	*Acceso Multiplataforma*
Descripción	*El sistema debe permitir el acceso a traves de la WEB y aplicación movil.*
Clasificación	*Compatibilidad*
Criterios de aceptación	*Los usuarios deben poder acceder desde cualquier dispositivo movil o navegador web.*

ID	*RNF02*
Nombre	*Acceso al sistema*
Descripción	*El sistema no debe permitir mas de 4 intentos fallidos.*
Clasificación	*Seguridad*
Criterios de aceptación	*Los usuarios no deben poder realizar mas de 4 intentos de sesión si sus datos no son correctos.*

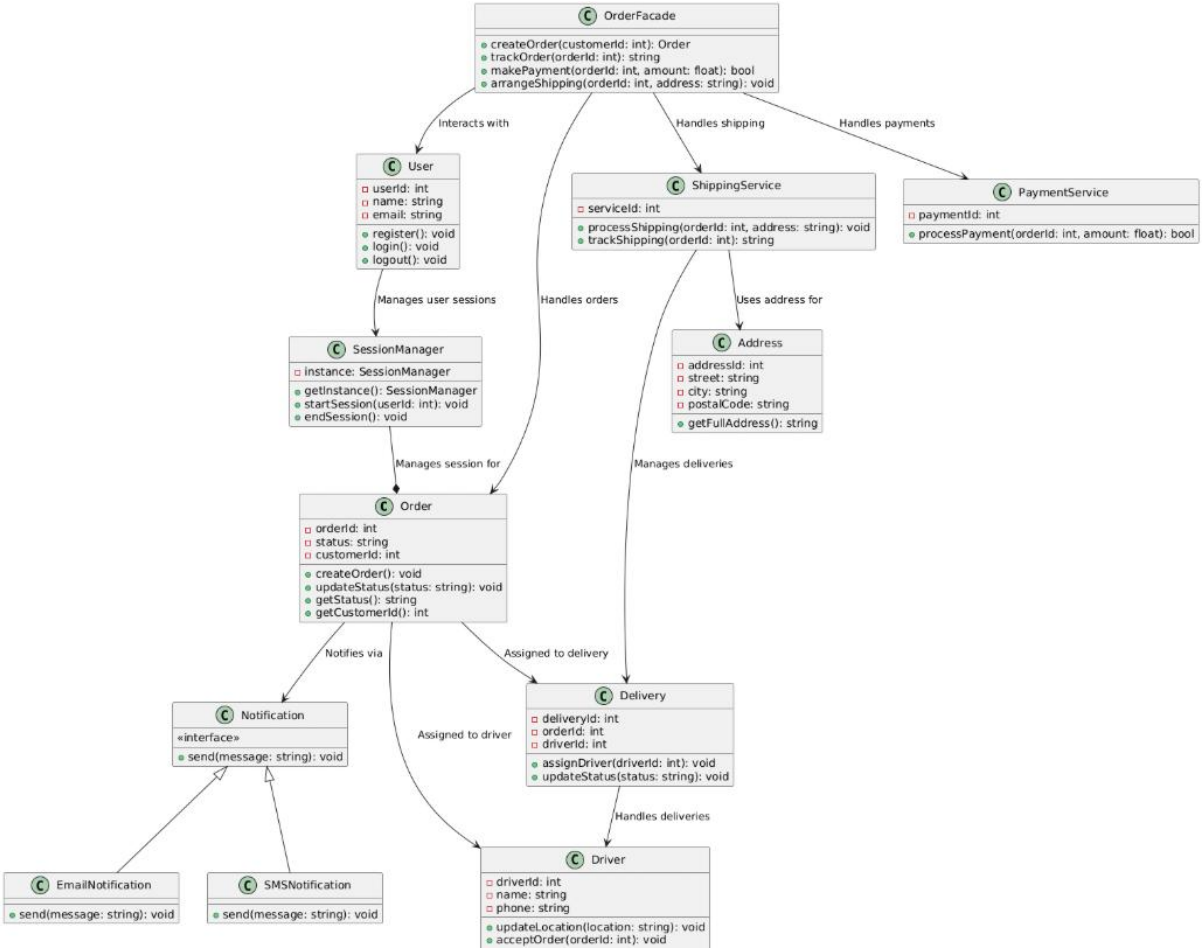
ID	*RNF03*
Nombre	*Integridad de los datos*
Descripción	*El sistema debe garantizar la integridad de los datos de los usuarios.*
Clasificación	*Seguridad*
Criterios de aceptación	*El sistema debe garantizar el cifrado de los datos y previnir el uso indebido de estos.*

ID	*RNF04*
Nombre	*Disponibilidad del sistema*
Descripción	*El sistema debe garantizar el correcto funcionamiento de la aplicación.*
Clasificación	*Fiabilidad*
Criterios de aceptación	*El sistema debe operar las 24 horas del dia los 7 dias de la semana.*

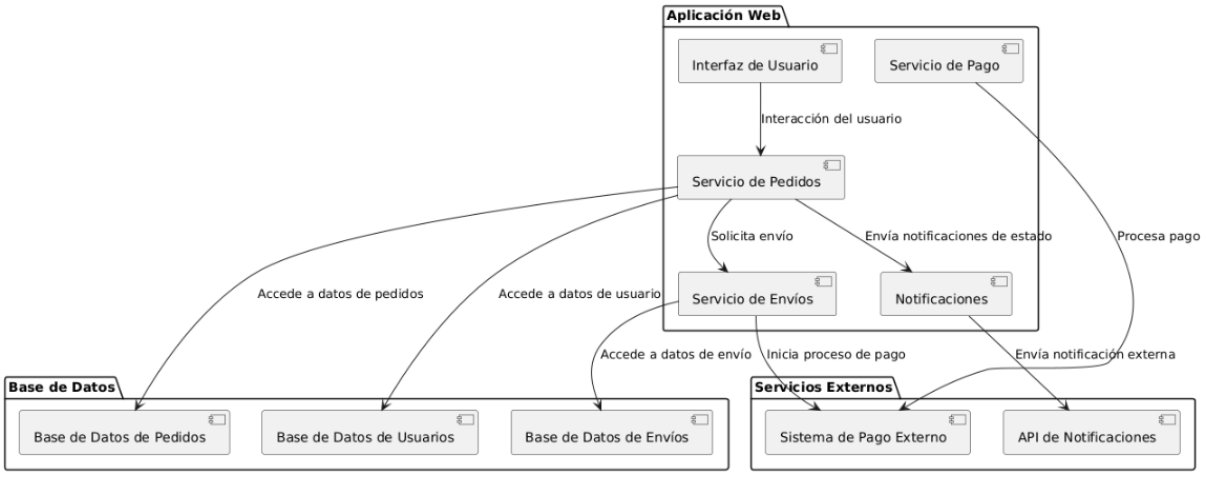
ID	*RNF05*
Nombre	*Usabilidad del sistema*
Descripción	*El sistema debe permitir un rapido aprendizaje.*
Clasificación	*Usabilidad*
Criterios de aceptación	*El sistema debe contar con un manual de usuario y con una curva de aprendizaje menor a 10 minutos.*

2. Diseño de la Arquitectura de la solución

2.1 Diagrama de clases



2.2 Diagrama de componentes



2.3 Selección del tipo de arquitectura

Para el sistema descrito, una arquitectura basada en **microservicios** es ideal. Esta arquitectura ofrece varias ventajas clave para un sistema de esta naturaleza:

- **Escalabilidad:** Los microservicios permiten escalar componentes individuales según sea necesario. Por ejemplo, el servicio de rastreo de pedidos y el servicio de optimización de rutas pueden ser escalados independientemente, según la carga de trabajo de cada uno.
- **Mantenibilidad y Desacoplamiento:** Cada microservicio tiene su propia responsabilidad (gestión de pedidos, rastreo de domiciliarios, optimización de rutas, etc.), lo que facilita la actualización, el mantenimiento y la implementación de nuevas funcionalidades sin afectar al sistema en su totalidad.
- **Desarrollo Independiente:** Los equipos pueden trabajar de manera independiente en cada microservicio, utilizando diferentes tecnologías y herramientas según las necesidades de cada componente, lo cual acelera el desarrollo.
- **Resiliencia:** Si un microservicio falla, no afecta al resto del sistema. Por ejemplo, si el servicio de rastreo de pedidos falla, el sistema de gestión de pedidos puede seguir operando sin interrupciones.

2.4 Identificación de patrones

Patrones Creacionales

Factory Method (FM)

Aplicación: Creación de diferentes tipos de notificaciones (correo electrónico, SMS, notificación push).

Justificación Técnica: Permite encapsular la creación de objetos de notificación y hacerla extensible, agregando nuevos tipos de notificación sin modificar el código existente.

Singleton (SG)

Aplicación: Gestión de sesiones de usuario, donde solo debe haber una instancia para manejar las sesiones a lo largo del sistema.

Justificación Técnica: Garantiza que solo exista una instancia para manejar las sesiones, mejorando el rendimiento y la gestión de acceso concurrente.

Patrones Estructurales

Adapter (AD)

Aplicación: Adaptación del sistema de gestión de pedidos con servicios de pago de terceros (como PayPal o Stripe) y sistemas de envío.

Justificación Técnica: Permite que el sistema interactúe con diferentes plataformas sin modificar su lógica interna, facilitando la integración con servicios externos.

Facade (FA)

Aplicación: Proveer una interfaz simplificada para la interacción con otros subsistemas (por ejemplo, sistemas de pago, inventarios, seguimiento de pedidos).

Justificación Técnica: Facilita la interacción entre subsistemas complejos de forma más sencilla, manteniendo la simplicidad para los usuarios del sistema.

Proxy (PR)

Aplicación: Implementación de un proxy de seguridad para controlar el acceso a datos sensibles o servicios externos.

Justificación Técnica: Añade una capa de seguridad y control de acceso sin modificar la estructura del sistema, lo que mejora la gestión de acceso a recursos críticos.

3. Implementación de la Solución

3.1 Descripción y Justificación Técnica del Stack de Herramientas Seleccionadas

El stack tecnológico seleccionado para el desarrollo de este sistema de gestión de pedidos a domicilio está basado en tecnologías modernas y escalables que permiten manejar la arquitectura de microservicios de forma eficiente. A continuación, se detallan las herramientas elegidas:

Lenguaje de Programación: Java

- **Justificación Técnica:**
  - **Portabilidad y Escalabilidad:** Java es un lenguaje robusto, de alto rendimiento y ampliamente utilizado en aplicaciones empresariales. Es ideal para sistemas

distribuidos como el propuesto (microservicios), lo que asegura una fácil integración con otros servicios y escalabilidad horizontal.

- **Comunidad y Ecosistema:** Java tiene una extensa comunidad de desarrolladores, lo que facilita la resolución de problemas y el acceso a bibliotecas y frameworks probados.
- **Compatibilidad con Frameworks Modernos:** Java es compatible con frameworks modernos que facilitan la implementación de microservicios y patrones de diseño.

Framework: Spring Boot (versión 2.6.x)

- **Justificación Técnica:**
  - **Desarrollo Rápido y Sencillo:** Spring Boot permite desarrollar aplicaciones de microservicios de forma rápida y sencilla, con configuraciones mínimas. Este framework también incluye capacidades como la configuración automática de componentes, lo que facilita la implementación de patrones como la inyección de dependencias.
  - **Integración con Microservicios:** Spring Boot es muy adecuado para arquitectura de microservicios debido a su soporte para RESTful APIs, integración con bases de datos, y gestión de dependencias.
  - **Seguridad y Manejo de Sesiones:** Spring Security y Spring Session proporcionan mecanismos robustos para la gestión de sesiones de usuario, autenticación y autorización, ideales para el **SessionManager**.
  - **Soporte para Patrones de Diseño:** Spring Boot facilita la implementación de patrones de diseño como Singleton, Factory Method, y Proxy mediante anotaciones y mecanismos de inyección de dependencias.

Base de Datos: PostgreSQL

- **Justificación Técnica:**
  - **Desempeño y Fiabilidad:** PostgreSQL es una base de datos relacional robusta y altamente escalable. Se elige por su capacidad para manejar grandes volúmenes de datos y su compatibilidad con transacciones complejas.

- **Soporte para Integridad de Datos:** PostgreSQL proporciona mecanismos avanzados de seguridad, integridad referencial y control de transacciones, ideales para el manejo de datos sensibles de pedidos y pagos.
- **Relación con el Modelo de Datos:** La estructura relacional de PostgreSQL es adecuada para modelar datos como pedidos, usuarios, envíos y pagos.

Gestor de Dependencias: Maven (versión 3.x)

- **Justificación Técnica:**
  - **Facilidad de Gestión de Dependencias:** Maven es uno de los gestores de dependencias más populares y utilizados en Java. Permite administrar bibliotecas de manera sencilla, facilitando la integración con herramientas como Spring Boot.
  - **Automatización del Ciclo de Vida del Proyecto:** Maven ofrece un ciclo de vida de construcción automatizado que facilita la implementación y despliegue de microservicios.

Contenedor de Servicios: Docker

- **Justificación Técnica:**
  - **Aislamiento de Componentes:** Docker permite crear contenedores ligeros para cada microservicio, lo que asegura que cada servicio se ejecute en un entorno controlado y sin interferencias entre ellos.
  - **Facilidad de Despliegue y Escalabilidad:** Los contenedores Docker se pueden desplegar fácilmente en servidores locales o en la nube, permitiendo escalar los microservicios de manera eficiente.
  - **Integración con Kubernetes:** Docker se integra bien con Kubernetes para orquestar los microservicios y gestionar su escalabilidad.

3.2 Explicación de los Mecanismos del Framework para Integración con Patrones de Diseño

Patrón Singleton:

- **Implementación:**
    - En **Spring Boot**, el patrón Singleton se implementa utilizando el mecanismo de **beans** gestionados por el contenedor de Spring. Por ejemplo, el SessionManager podría ser definido como un @Service o @Component, lo que garantiza que solo haya una instancia del SessionManager a lo largo de la aplicación.

Justificación Técnica: Spring maneja el ciclo de vida de los beans y, por defecto, crea los beans como singletons. Este comportamiento facilita la implementación del patrón Singleton sin necesidad de una implementación manual de la lógica de instanciación.

Patrón Factory Method:

- **Implementación:**
    - **Factory Method** puede ser utilizado para crear diferentes tipos de notificaciones (correo electrónico, SMS, notificación push). Usamos la anotación @Bean en Spring para definir un método que devuelve instancias de objetos según el tipo solicitado.

Justificación Técnica: El patrón Factory Method permite encapsular la creación de objetos complejos. En este caso, Spring gestiona la creación de las instancias y la configuración de dependencias, lo que facilita la extensión del sistema (por ejemplo, agregando nuevos tipos de notificación).

Patrón Proxy:

- **Implementación:**
    - El **Patrón Proxy** se puede utilizar para proteger el acceso a los servicios de pago o gestión de datos sensibles. Por ejemplo, un proxy podría verificar el estado de la sesión antes de permitir una transacción.

Justificación Técnica: El patrón Proxy proporciona una capa adicional de control de acceso o funcionalidad (en este caso, autenticación) antes de permitir que la lógica del servicio real se ejecute. La inyección de dependencias en Spring facilita este tipo de patrón sin necesidad de configurar manualmente las relaciones entre clases.

Patrón Adapter:

- **Implementación:**
    - El **Patrón Adapter** se utiliza para permitir que el sistema interactúe con sistemas de pago o notificación de terceros (como PayPal o Twilio). Spring permite la implementación de adaptadores utilizando interfaces y clases concretas para adaptarse a los requisitos de la API externa.

Justificación Técnica: El patrón Adapter permite que el sistema se comunique con servicios de terceros sin modificar la lógica interna del sistema. En Spring, la inyección de dependencias permite utilizar diferentes adaptadores para integrar varios servicios externos.

CONCLUSIÓN

El desarrollo de la solución propuesta permitió abordar de manera efectiva las necesidades identificadas en los procesos de gestión de pedidos a domicilio. A través de la incorporación de patrones de diseño creacionales y estructurales, se logró construir un sistema modular, flexible y escalable que facilita el mantenimiento, la integración de nuevas funcionalidades y la adaptación a futuras necesidades del negocio.

La elección de una arquitectura basada en microservicios, soportada por tecnologías como Java, Spring Boot, PostgreSQL y Docker, no solo garantiza la escalabilidad y disponibilidad del sistema, sino que también permite el desarrollo independiente de componentes, favoreciendo el trabajo colaborativo y la mejora continua. Asimismo, la aplicación de patrones como Singleton, Factory Method, Adapter, Facade y Proxy permitió resolver problemáticas comunes de diseño, aportando robustez y claridad en la implementación de los distintos módulos del sistema.

En conjunto, esta solución brinda a los comercios un mayor control sobre sus operaciones logísticas, optimiza los tiempos de entrega, mejora la experiencia de los clientes mediante el rastreo en tiempo real y reduce la carga operativa asociada a la atención de solicitudes manuales. El sistema desarrollado representa una herramienta tecnológica alineada a las demandas actuales del mercado, con un diseño que permite su evolución y crecimiento futuro.

BIBLIOGRAFIA

R. S. Pressman, Software engineering: a practitioner's approach, Eighth edition. New York, NY: McGraw-Hill Education, 2015.

(S/f). Edu.pe. Recuperado el 20 de junio de 2025, de https://revistas.ulasalle.edu.pe/innosoft/article/view/37

(S/f-b). Upv.es. Recuperado el 20 de junio de 2025, de https://polipapers.upv.es/index.php/RIAI/article/view/17791/15334