



SEMINARIO DE PRACTICA DE INFORMATICA

TRABAJO PRACTICO N°3

Materia: Seminario De Practica De Informatica

Titular Experto: Virgolini Pablo Alejandro

Fecha de entrega: 27/10/2024

Estudiante: Levato Camila Jazmín.

Legajo: VINF013268

Carrera: Licenciatura en Informática.

Camila Levato

2024

INDICE

Titulo	2
Introducción	2
Justificación	2
Definición del proyecto	3
Elicitación	4
Conocimiento del negocio	9
Diagrama de dominio	10
Procesos de negocio	11
Propuesta de solución	11
Propuesta funcional	11
Propuesta técnica	12
Requerimientos	12
Trazabilidad	16
Especificaciones de casos de uso	16
Etapas de análisis	20
Etapas de diseño	24
Etapas de implementación	25
Etapas de pruebas	26
Plan de pruebas	26
Caso de prueba	27
Evaluación de pruebas y tratamiento de defectos	30
Interfaz grafica	31
Definición de base de datos para el prototipo	33
Diagrama entidad relacion	34
Creación de tablas	35
Requerimientos de comunicación del sistema	36
Desarrollo del sistema utilizando Java	36

Título

Desarrollo de un sistema de gestión de stock integrado para el E-commerce y sucursal física del Monarca Market.

Introducción

Supermercado Monarca Market es una empresa familiar con una extensa trayectoria en el rubro de la alimentación y venta al por menor. La compañía ofrece a sus clientes una amplia gama de productos de calidad al mejor precio posible, abarcando categorías como productos de almacén, frutas y verduras, fiambres y lácteos, perfumería e higiene personal, bebidas, pastas, artículos de limpieza, carnicería, panadería, y verdulería. Estos productos están disponibles tanto en los salones comerciales como a través de su página de e-commerce.

Hace aproximadamente tres años, Monarca Market decidió incursionar en el comercio electrónico, un cambio que ha transformado significativamente la forma en que los clientes adquieren productos. Este nuevo enfoque ha renovado sus pilares fundamentales: servicio, atención y excelencia. Como empresa proveedora de servicios, se focaliza en brindar a los consumidores soluciones a sus necesidades de alimentación, higiene personal y limpieza, buscando siempre la excelencia en la atención al cliente a través de un ambiente organizado, limpio, iluminado y debidamente comunicado.

Sin embargo, este avance también ha traído consigo desafíos importantes. La falta de sincronización entre el stock disponible en el e-commerce y el de las sucursales físicas ha generado problemas significativos, como la falta de productos en los pedidos, la disminución de la participación de mercado y una mala experiencia del cliente, lo que resulta en la pérdida de fidelización. Por ejemplo, en el último año, se ha reportado un aumento del 19.8 % en las quejas de clientes relacionadas con la disponibilidad de productos.

El proyecto actual tiene como objetivo diseñar y desarrollar un sistema de gestión de stock en tiempo real que integre ambos canales, representando un paso fundamental hacia el cumplimiento de sus valores. La empresa cuenta con personal capacitado y en constante aprendizaje y desarrollo para hacer frente a los intereses y necesidades dinámicas de los clientes. La meta es conformar la cadena de supermercados más importante de la provincia de Buenos Aires, instalando y posicionando positivamente la marca Monarca Market en la mente de cada consumidor.

Justificación

La gestión eficiente del stock es crucial para Monarca Market, ya que una administración ineficiente entre el e-commerce y la sucursal física limita la capacidad de respuesta ante las demandas del mercado y genera frustración en los clientes. Cuando los consumidores realizan un pedido y descubren que ciertos artículos no están disponibles, a pesar de que se les informa sobre la posibilidad de reembolso en 72 horas, esto crea desconfianza y afecta negativamente la percepción de la empresa.

Este proyecto de implementación de un sistema de gestión de stock requerirá avances en tecnología, análisis y procesamiento de datos, así como la alineación de objetivos estratégicos. Además, será fundamental fomentar la generación de nuevos conocimientos mediante la formación continua del personal, lo que garantizará la adaptabilidad y resiliencia de la organización ante cambios en el entorno. La capacitación en herramientas digitales y

metodologías ágiles permitirá al equipo optimizar la gestión de inventarios de manera efectiva, reduciendo quejas y devoluciones.

Dado que el comercio electrónico permite a empresas de todos los tamaños vender productos a gran escala, una gestión de inventarios efectiva se vuelve indispensable. Al mejorar la eficiencia en el manejo de recursos, el proyecto promoverá un enfoque sostenible, reduciendo desperdicios y fomentando prácticas responsables. En definitiva, esto no solo fortalecerá la competitividad de Monarca Market en un entorno dinámico, sino que también mejorará la confianza y satisfacción del cliente, asegurando el crecimiento sostenible de la empresa.

Definición del proyecto

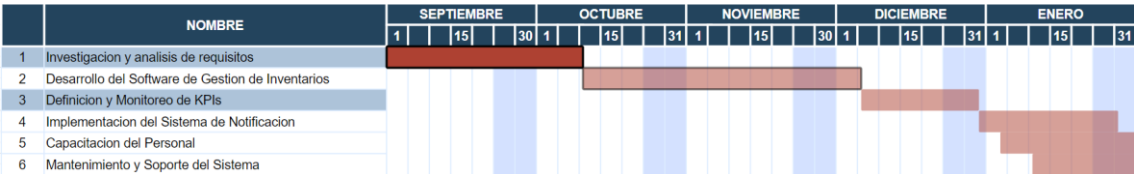
Objetivo general del proyecto

Desarrollar un sistema de gestión de inventario en tiempo real que integre los canales de e-commerce y tiendas físicas, optimizando el inventario para aumentar la participación en el mercado y mejorar la experiencia del cliente. Este sistema debe reflejar en las compras en línea los mismos valores de servicio y excelencia que se ofrecen en las tiendas físicas, explorando nuevas alternativas tecnológicas y capacitando al personal para asegurar una gestión eficiente del inventario desde el equipo de reposición hasta el personal administrativo.

1. **Implementar y validar** un software de gestión de inventarios en tiempo real que permita la integración de datos entre los canales de e-commerce y tiendas físicas.
2. **Establecer y supervisar** indicadores clave de rendimiento (KPIs) como la tasa de satisfacción del cliente, cantidad de productos pedidos y entregados, tasa de cumplimiento de pedidos, tasa de devoluciones, tiempo promedio de entrega, entre otros.
3. **Desarrollar e implementar** un sistema de notificaciones para la gestión proactiva del inventario, capacitando al personal en su uso para asegurar una respuesta eficiente y oportuna.

A continuación, para llevar a cabo correctamente el proyecto de forma organizada, realizaremos el diagrama de Gantt. Esta es una herramienta de gestión de proyectos que ilustra el trabajo realizado durante un período de tiempo, en este caso, de septiembre a enero. Normalmente, el diagrama de Gantt tiene dos secciones: en la parte izquierda se incluye una lista de tareas y, en la derecha, un cronograma con barras que representan el trabajo. Hoy en día, el diagrama de Gantt es conocido como una “herramienta de hoja de ruta”. Se utiliza para crear y gestionar un proyecto completo, determinar la logística y las dependencias de las tareas, y supervisar el progreso del proyecto.

Existen dos razones principales por las que los diagramas de Gantt son tan apreciados en la gestión de proyectos. Por un lado, facilitan la creación de planes complejos, especialmente aquellos en los que participan varios equipos y cuyos plazos cambian. Los diagramas ayudan a los equipos a planificar el trabajo basándose en los plazos y a asignar los recursos correctamente. Por otro lado, los gestores de proyectos utilizan los diagramas de Gantt para tener una visión general de los proyectos. En ellos se representan, entre otras cosas, la relación entre las fechas de inicio y finalización de las tareas, los hitos y las tareas dependientes. Los gestores de proyectos y líderes utilizan estos diagramas para ayudar a las organizaciones a alcanzar sus objetivos.



Definiciones del sistema

Objetivo general del sistema

El proyecto tiene como objetivo desarrollar e implementar un sistema que sincronice el inventario del supermercado entre la plataforma de e-commerce y las sucursales físicas, proporcionando así una solución integral. Este sistema ofrecerá una visión unificada del stock en tiempo real, enfocándose en la optimización de la gestión del inventario y mejorando la experiencia del cliente en cuanto a los pedidos. Esto se logrará a través de la recopilación, análisis y procesamiento de datos en tiempo real, asegurando que la información esté actualizada en ambos canales de venta. Todo esto se realizará siguiendo los valores de la empresa: servicio, atención y excelencia.

Límites del sistema:

- Desde: La implementación del sistema se limita a las sucursales y operaciones de e-commerce ubicadas en la provincia de Buenos Aires.
- Hasta: Las localidades específicas de implementación incluyen San Carlos de Bolívar, Olavarría, Las Flores y Daireaux.
- Desde: El sistema integrará los datos de inventario de los canales de venta de e-commerce y las sucursales físicas de Monarca Market.
- Hasta: Inicialmente se enfocará en las categorías de productos más vendidos y de mayor rotación.

Alcances del sistema:

- El sistema permitirá la gestión en tiempo real de los stocks, integrando los inventarios de la sucursal física y del e-commerce.
- Generará reportes periódicos sobre el rendimiento, niveles de stock, cantidad de pedidos, pedidos por clientes, entre otros.
- Enviará notificaciones al personal sobre niveles de stock bajo.
- Capacitará al personal para que pueda responder adecuadamente, involucrándolo en el nuevo sistema de stock y mejorando la eficiencia en conjunto.

Restricciones del sistema:

- Se requerirá la adaptación tecnológica para las nuevas necesidades, lo que implica contar con una infraestructura adecuada para eficientizar los procesos, estando sujetos a un presupuesto previamente definido.
- Habrá restricciones de tiempo para la implementación del sistema, buscando llevar a cabo el proceso de manera eficiente lo antes posible.
- Se deberá prestar un enfoque cuidadoso en la capacitación del personal, ya que será un cambio significativo en el funcionamiento, lo que requiere una gestión del cambio y una comunicación interna efectivas.

Elicitación

La elicitación es el proceso de adquirir “todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema” (Loucopoulos, 1995,

como se citó en Oliveros y Antonelli, 2015, p. 2). En este proyecto, llevaremos a cabo diversas actividades para comprender mejor la experiencia de compra en línea y la gestión de stock en el Supermercado Monarca Market.

Nuestro enfoque estará en optimizar la experiencia de compra en línea y mejorar la gestión de stock, siempre centrados en los valores de servicio, atención y excelencia. Nos enfocaremos en analizar y mejorar los procesos actuales del inventario, identificando las áreas que necesitan mejoras para lograr una correcta sincronización entre las sucursales físicas y el e-commerce.

Para obtener información más precisa, realizaremos encuestas a los clientes una vez recibido el pedido y llevaremos a cabo entrevistas con el dueño, el encargado de compras y los empleados que preparan los pedidos. Estas entrevistas nos proporcionarán datos críticos sobre la percepción del cliente y la eficiencia operativa interna.

Encuesta de Satisfacción de Compra en Línea

Objetivo: Esta encuesta tiene como finalidad recopilar información detallada sobre la experiencia del cliente tras recibir su pedido, centrándose en evaluar la satisfacción con el proceso de compra y entrega.

Tecnologías de la Información y la Comunicación:

- Herramienta de Distribución: La encuesta será enviada al correo electrónico registrado en el e-commerce.
- Plataforma Utilizada: Google Forms, que permite una fácil recopilación y gestión de respuestas.
- Incentivo: Los participantes recibirán un 20% de descuento en su próxima compra luego de completar la encuesta.
- Herramienta de Análisis de Datos: Utilizaremos Excel para procesar y visualizar los datos recopilados, lo que nos permitirá identificar áreas de mejora y tendencias en la satisfacción del cliente.

Preguntas

1. ¿Cómo calificarías la facilidad de uso de nuestro sitio?
(1 muy difícil - 5 muy fácil)
2. ¿Encontraste fácilmente la información que buscabas?
 - Sí
 - No
3. ¿Cómo calificarías la disponibilidad de productos en nuestro sitio?
(1 muy baja - 5 muy alta)
4. ¿Recibiste el producto que pediste?
 - Sí
 - No
5. ¿El estado del stock fue claro durante tu proceso de compra?
 - Sí
 - No
6. ¿Experimentaste problemas de lentitud durante el proceso de compra?

- Sí
- No

7. ¿Recibiste el producto en el tiempo estimado?

- Sí
- No

8. ¿Hubo algún obstáculo durante la finalización de la compra?

- Sí (especificar)
- No

9. En una escala del 1 al 10, ¿qué tan satisfecho/a estás con tu experiencia de compra en línea?

(1 muy insatisfecho - 10 muy satisfecho)

10. ¿Volverías a comprar en nuestro sitio?

- Sí
- No

11. ¿Hay algo que mejorarías en relación con la gestión de stock o el proceso de compra?
(Espacio para respuesta abierta)

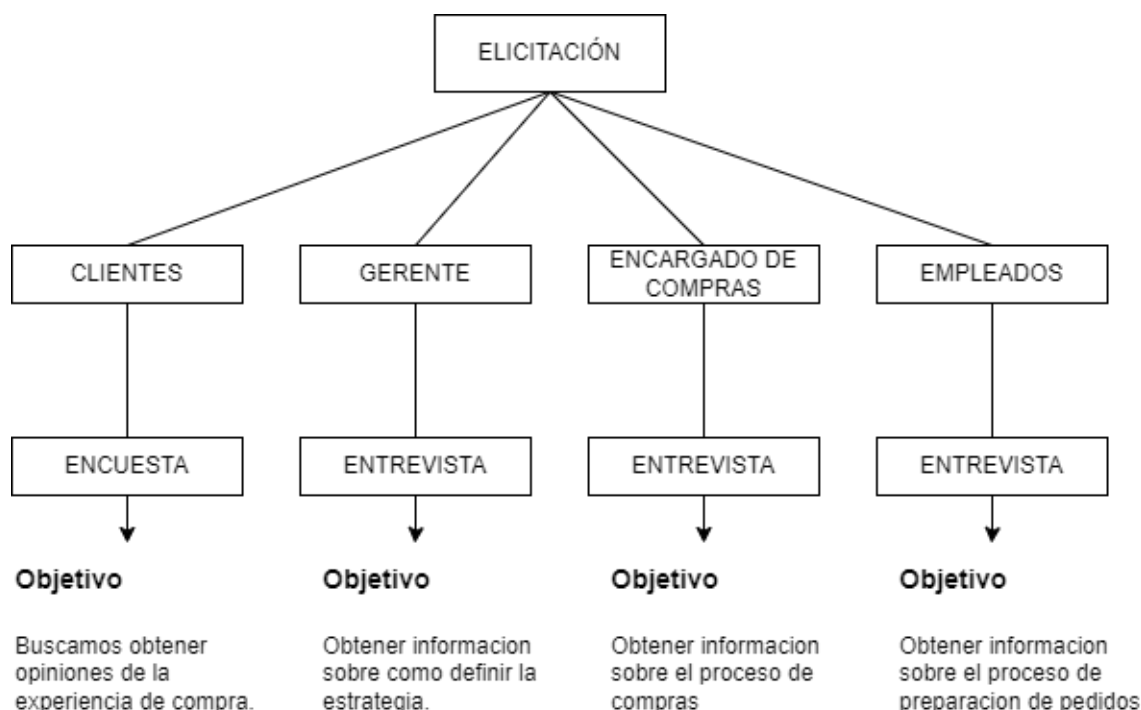
12. Comentarios adicionales sobre tu experiencia en nuestro sitio web:
(Espacio para respuesta abierta)

Análisis de competencia:

Utilizaremos benchmarking, un estudio que profundiza en las encuestas realizadas por nuestros competidores para entender las mejores prácticas que ellos implementan. A través de este análisis, Monarca Market podrá adaptar ciertas acciones para atraer y reconquistar a su público. En este caso, investigaremos las encuestas de satisfacción de otras empresas y compararemos las preguntas utilizadas.

Además, realizaremos un estudio de casos de éxito y fracasos, analizando las estrategias de gestión de stock que han tenido éxito en nuestros competidores, así como los errores que han cometido. Este enfoque nos permitirá aprender de sus experiencias y evitar repetir sus fallos.

También es crucial analizar la facilidad, eficiencia y velocidad del proceso de compra en línea de Monarca Market en comparación con el de nuestros competidores. Esto nos ayudará a identificar áreas de mejora y a implementar cambios que optimicen la experiencia del cliente.



ENTREVISTA CON EL GERENTE

Durante la entrevista, el gerente del Supermercado Monarca Market, se discutió la situación actual del e-commerce y los desafíos que enfrenta la empresa. Mencionó que, aunque cuentan con cinco sucursales físicas, el mantenimiento de estas implica gastos significativos que son difíciles de afrontar en la actualidad. Por esta razón, la empresa está enfocándose cada vez más en el comercio electrónico como su principal canal de venta.

Jorge también compartió que, a futuro, están planificando la implementación de un "supermercado ciego", donde las cajas serán operadas manualmente. Sin embargo, su mayor preocupación en este momento radica en la falta de coincidencia entre los stocks de las sucursales y los pedidos en línea. Esta discrepancia está afectando la capacidad de la empresa para cumplir con las expectativas de los clientes, lo que es crítico para alcanzar sus objetivos comerciales.

Él enfatizó que, hasta que no se mejore la gestión del stock, no podrán avanzar hacia sus metas. El gerente destacó que un stock incorrecto impacta negativamente en toda la operación, desde la atención al cliente hasta la logística de reposición.

Además, mencionó la importancia de establecer procesos adecuados para la reposición y el armado de pedidos, ya que estos son elementos fundamentales para asegurar un funcionamiento eficiente del stock. Es consciente de que, sin un sistema de gestión de inventario sólido, será difícil satisfacer la demanda del cliente y optimizar los recursos de la empresa.

La entrevista con el gerente subraya la urgencia de transformar la gestión del stock en Monarca Market, ya que la transición hacia un enfoque más centrado en el e-commerce es una necesidad para la supervivencia y el crecimiento del negocio. Implementar procesos claros para la reposición y el armado de pedidos será crucial para cumplir con las expectativas de los clientes y mejorar la competitividad. Por ello, es fundamental abordar el proyecto de gestión

de stock coordinando el inventario físico con el del e-commerce, asegurando que ambas áreas trabajen de manera sinérgica para avanzar hacia sus objetivos y asegurar un futuro próspero en el comercio electrónico.

Conclusiones

La conversación con el gerente resalta la necesidad urgente de implementar un sistema que garantice la sincronización del stock entre las sucursales físicas y el e-commerce. Solo a través de una gestión eficiente del inventario podrán mejorar la satisfacción del cliente y, en última instancia, la rentabilidad del negocio.

ENTREVISTA CON EL ENCARGADO DE COMPRAS

La charla que tuvimos con el encargado de compras también reflejó las preocupaciones planteadas por el gerente. Comentó que mantiene un contacto constante con el sector de marketing, lo que le permite tener una visión más amplia de la situación. Al principio, las compras eran abundantes, pero debido a ciertos faltantes en pedidos esenciales de e-commerce, como lácteos, ha habido una disminución en las compras diarias en las sucursales. Esta situación se debe, en gran medida, a la falta de coordinación entre el stock de las sucursales y el del e-commerce.

Sin embargo, el encargado es optimista y considera que, si se mejora esta coordinación, la tendencia hacia las compras online continuará en aumento. La gente se está acostumbrando a comprar por internet, especialmente porque muchos productos son más económicos en línea. También es importante educar a los consumidores sobre las diferencias entre el e-commerce y las sucursales físicas, ya que en el canal digital se priorizan los productos más vendidos y solicitados, lo que puede no reflejar la totalidad del inventario disponible en las tiendas físicas.

Enfatizó que, para maximizar las ventas en el e-commerce, es crucial no solo mejorar la disponibilidad de productos, sino también implementar estrategias de marketing que informen a los clientes sobre las ventajas de comprar en línea. Esto incluye promociones exclusivas y una comunicación clara sobre las diferencias en el inventario. Al hacerlo, no solo se aumentarán las ventas, sino que también se fomentará una mayor lealtad del cliente hacia la plataforma de e-commerce.

Conclusiones

Las conclusiones de la entrevista con el encargado de compra destacan la importancia de coordinar el stock entre las sucursales y el e-commerce para mejorar la disponibilidad de productos y satisfacer la demanda del cliente. A medida que los consumidores se acostumbran a comprar en línea, es esencial implementar estrategias de marketing que resalten las ventajas del e-commerce y educar a los clientes sobre las diferencias en el inventario. Además, priorizar los productos más vendidos en la plataforma digital y fomentar una mejora continua en la gestión de inventario son pasos clave para asegurar el éxito a largo plazo de Monarca Market en un entorno cada vez más competitivo.

ENTREVISTA CON EMPLEADOS

En esta entrevista, hablamos con la encargada de realizar los pedidos, quien trabaja en la sucursal de Brown, San Carlos de Bolívar. Durante nuestra conversación, compartió detalles sobre los procesos específicos del e-commerce. Mencionó que la preparación de pedidos es relativamente sencilla, sin embargo, el sistema ofrece a los clientes la opción de sustituir un artículo si no está disponible. Esta sustitución debe ser por un producto de igual o menor valor, lo que puede complicar el proceso, ya que requiere que la persona que prepara analice en tiempo real qué producto ofrecer como sustituto. Además, ella tiene un contacto directo con los clientes, quienes a veces expresan quejas cuando se les sustituyen productos esenciales para sus comidas.

Si el cliente opta por no aceptar sustituciones y un producto no está disponible, se realiza la devolución del dinero en un plazo de 24 horas hábiles. La encargada señala que, aunque el sistema de e-commerce funciona bien en general, la principal inconsistencia radica en la gestión del stock, lo cual impacta negativamente en la dinámica diaria de trabajo. Al final de cada jornada, ella envía un informe de los faltantes del día, pero ha notado que estos ajustes no siempre se reflejan correctamente en el sistema.

Además, destacó que a veces los empleados encargados de preparar los pedidos no encuentran un producto rápidamente y marcan que no está disponible. En estos casos, hay un supervisor que debe autorizar cualquier cambio en el pedido. Esto subraya la importancia de que todos los miembros del equipo realicen sus tareas de manera eficiente; de lo contrario, el pedido no se completa adecuadamente, lo que agrava el problema del stock. Para mejorar la situación, sugiere implementar un sistema de comunicación más eficaz entre los empleados y una capacitación adicional sobre la gestión de inventario, lo que podría facilitar la identificación de productos y disminuir las quejas de los clientes.

Conclusiones

La entrevista resalta la necesidad de mejorar la gestión del stock en el e-commerce para optimizar la experiencia del cliente y la eficiencia operativa. Aunque el sistema de pedidos funciona adecuadamente, la falta de coordinación en la disponibilidad de productos y la comunicación entre los empleados generan inconsistencias que afectan la satisfacción del cliente. Implementar un sistema de comunicación más efectivo y ofrecer capacitación adicional sobre la gestión de inventario son pasos cruciales para abordar estos desafíos y garantizar que los pedidos se completen de manera adecuada y oportuna.

Conocimiento del negocio

El sistema que se va a desarrollar debe permitir la gestión y monitoreo del inventario en Monarca Market, optimizando así la disponibilidad de productos tanto en el e-commerce como en las sucursales físicas.

En un supermercado, el inventario incluye una variedad de productos, desde alimentos hasta artículos de limpieza. La correcta gestión de este inventario es crucial para asegurar que los productos estén disponibles cuando los clientes los necesiten. Un sistema eficiente permitirá actualizar en tiempo real los niveles de stock, evitando faltantes y mejorando la experiencia del cliente.

Para optimizar la gestión del stock, se debe implementar estrategias como el análisis de datos de ventas para prever la demanda y ajustar los niveles de inventario en consecuencia. Además, se pueden establecer alertas automáticas para notificar a los empleados cuando los niveles de ciertos productos estén por debajo de un umbral crítico, facilitando así la reposición oportuna.

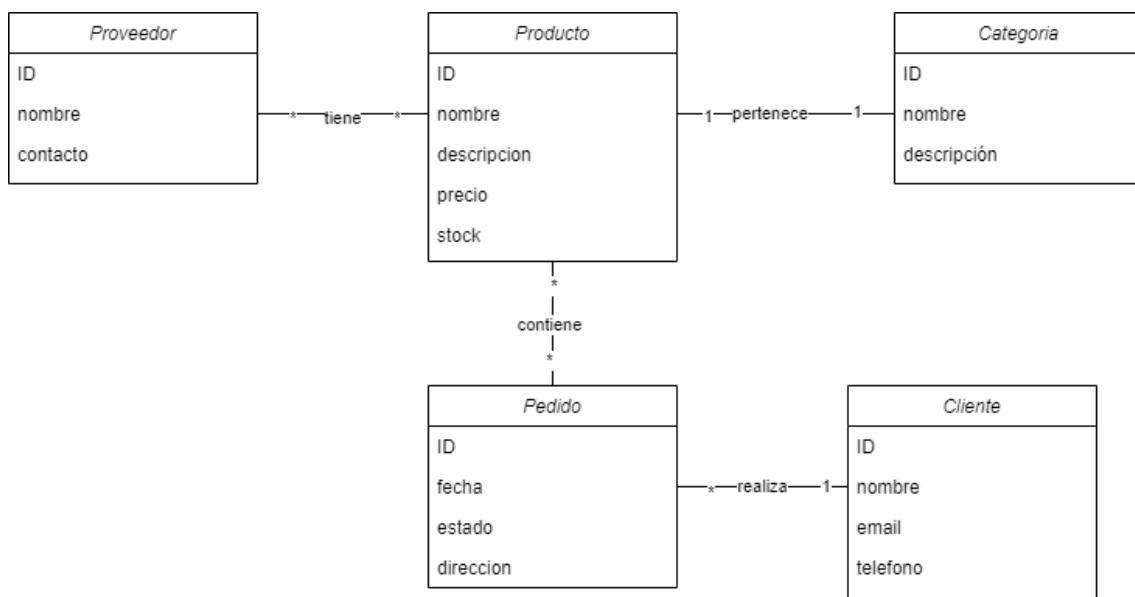
El sistema también permitirá la integración de información de ventas y stock, lo que ayudará a identificar productos de alta rotación y a planificar mejor las compras. Esto no solo mejorará la

eficiencia operativa, sino que también contribuirá a una experiencia de compra más satisfactoria para los clientes, asegurando que siempre encuentren lo que buscan en Monarca Market.

Diagrama de dominio

En esta sección del trabajo, realizaremos el diagrama de dominio, que es una representación gráfica de los conceptos, entidades y relaciones clave dentro del dominio del negocio. Este diagrama nos ayudará a comprender mejor el contexto del software. Es una herramienta fundamental para el desarrollo de un sistema de gestión de stock integrado para el E-commerce y la sucursal física de Monarca Market, ya que facilitará la comunicación entre los involucrados, mejorará la calidad del software y promoverá un diseño coherente y alineado con las necesidades del negocio.

Al proporcionar una representación clara y precisa del dominio, podremos desarrollar soluciones más robustas, mantenibles y adaptables.



La clase **Producto** está asociada a la clase **Categoría**, lo que significa que cada producto pertenece a una única categoría. Por otro lado, la clase **Pedido** está vinculada con **Producto**, ya que un pedido puede incluir múltiples productos, y muchos productos pueden estar presentes en varios pedidos.

Además, la clase **Pedido** está conectada con la clase **Cliente**, donde un cliente puede realizar múltiples pedidos, lo que implica que muchos pedidos pueden estar asociados a un solo cliente.

Finalmente, la clase **Producto** también está relacionada con **Proveedor**, ya que un producto puede tener múltiples proveedores.

Procesos de negocio

Gestión de Inventario:

- Revisión periódica del inventario para identificar niveles bajos de stock.
- Proceso de pedido de nuevos productos a proveedores para mantener niveles óptimos de stock.
- Registro y actualización de entradas y salidas de productos en el sistema.

Procesamiento de Pedidos:

- Recepción y confirmación de pedidos realizados por clientes a través del e-commerce.
- Selección y empaquetado de productos para cumplir con los pedidos de los clientes.
- Coordinación con servicios de remisería para el envío de productos a los clientes.

Atención al Cliente:

- Respuesta a consultas de clientes sobre disponibilidad de productos, estado de pedidos, etc.
- Proceso para manejar devoluciones y reembolsos de productos no deseados o defectuosos.

Relaciones con Proveedores:

- Establecimiento de términos y condiciones con proveedores para el suministro de productos
- Evaluación periódica de proveedores para asegurar calidad y cumplimiento de plazos.

Análisis de Datos:

- Creación de informes sobre ventas, niveles de stock, y desempeño de proveedores.
- Análisis de datos para identificar tendencias de consumo y ajustar estrategias de inventario.

Capacitación y Comunicación Interna:

- Programas de formación para mejorar las habilidades de los empleados en la gestión de inventario.
- Establecimiento de canales de comunicación efectivos entre los empleados para mejorar la coordinación.

Propuesta de solución

La propuesta de solución que brindamos a la problemática es un desarrollo de un sistema de gestión de stock integrado para el E-commerce y la sucursal física de Monarca Market

Propuesta funcional

El sistema permitirá gestionar el stock de productos, realizar un seguimiento de pedidos y facilitar la integración de stock entre la plataforma de E-commerce y la sucursal física de Monarca Market. Las funcionalidades claves incluirán stock en tiempo real, alertas de bajo stock, informes de ventas y un panel de control intuitivo, lo que permitirá a los usuarios tomar decisiones informadas y rápidas para optimizar la gestión del inventario. Además, se implementarán herramientas de análisis que ayudarán a identificar tendencias de compra y a anticipar necesidades de reabastecimiento, asegurando así que siempre haya disponibilidad de productos para los clientes.

Propuesta técnica

La propuesta técnica para el sistema de gestión de stock de Monarca Market se fundamenta en una arquitectura de servicios. Cada funcionalidad, como la gestión de stock, el seguimiento de pedidos y la generación de informes, se implementará como un servicio independiente. Esto facilitará actualizaciones y mejoras sin afectar el sistema en su totalidad.

El sistema utilizará una base de datos MySQL para almacenar de manera segura la información de productos, pedidos y usuarios, garantizando la consistencia de los datos. Esta elección es fundamental para almacenar, organizar y recuperar datos de manera eficiente, especialmente al manejar grandes volúmenes de información.

La interfaz de usuario se desarrollará con JavaFX, que ofrece un alto nivel de flexibilidad. Su enfoque basado en componentes permite construir aplicaciones a partir de elementos más pequeños y reutilizables.

En cuanto a la conectividad, utilizaremos Ethernet, una tecnología que nos permitirá conectar dispositivos y formar redes locales (LAN). Esta opción es fundamental para diversas tareas de Monarca Market, como la conexión de dispositivos, el intercambio de archivos y el almacenamiento de datos, brindando una base sólida para nuestro proyecto de inventario en tiempo real y asegurando escalabilidad y adaptabilidad.

Además, para garantizar el correcto funcionamiento de todos los servicios, se llevará a cabo una capacitación para los usuarios finales y se ofrecerá soporte técnico continuo para resolver cualquier duda que pueda surgir durante el uso del sistema. Esta propuesta técnica permitirá que Monarca Market optimice la gestión de stock en tiempo real de manera eficiente.

Requerimientos

A continuación, se listarán los requerimientos funcionales y no funcionales para este proyecto. Esta etapa es una de las más críticas e importantes, ya que permite abordar las diversas problemáticas identificadas desde la perspectiva de las necesidades del cliente o del usuario. Los requerimientos son descripciones de los servicios y las restricciones del sistema.

Requerimientos funcionales

Requerimiento sistema	Descripción
RFS01: Registrar usuarios	El sistema debe permitir a los usuarios registrarse en el sistema ingresando un correo electrónico, una contraseña y el rol correspondiente.
RFS02: Iniciar sesión	El sistema debe permitir a los usuarios iniciar sesión ingresando sus credenciales.
RFS03: Cambiar contraseña	El sistema debe permitir a los usuarios cambiar su contraseña, ya sea por olvido o porque lo deseen.
RFS04: Acceder según el rol	El sistema debe permitir a los usuarios acceder al sistema según su rol correspondiente.
RFS05: Registrar proveedores	El sistema debe permitir registrar proveedores, incluyendo su información de contacto y los productos asociados.
RFS06: Modificar proveedores	El sistema debe permitir a los usuarios modificar la información de proveedores ya existentes en el sistema.

RFS07: Bajar proveedores	El sistema debe permitir a los usuarios dar de baja a proveedores ya existentes.
RFS08: Consultar proveedores	El sistema debe permitir a los usuarios realizar consultas sobre proveedores ya existentes en el sistema.
RFS09: Registrar productos	El sistema debe permitir a los usuarios registrar nuevos productos, incluyendo nombre, descripción, precio y stock actual.
RFS10: Modificar productos	El sistema debe permitir a los usuarios modificar la información de productos ya existentes en el sistema.
RFS11: Bajar productos	El sistema debe permitir a los usuarios dar de baja a productos ya existentes.
RFS12: Consultar productos	El sistema debe permitir a los usuarios realizar consultas sobre productos ya existentes en el sistema.
RFS13: Visualizar stock disponible	El sistema debe permitir a los usuarios visualizar el stock disponible en tiempo real de cada producto.
RFS14: Recibir alerta sobre stock	El sistema debe permitir a los usuarios recibir alertas cuando un producto esté bajo en stock.
RFS15: Realizar transferencias entre sucursales	El sistema debe permitir a los usuarios realizar transferencias de stock entre sucursales según sea necesario.
RFS16: Registrar fecha	El sistema debe registrar la fecha y hora de cada acción realizada por el usuario.
RFS17: Acceder al historial de cambios:	El sistema debe permitir a ciertos usuarios de niveles más altos consultar las modificaciones realizadas en cada área, con su fecha y hora correspondiente.
RFS18: Historial de ventas	El sistema debe permitir a los usuarios acceder a un historial de ventas.
RFS20: Gestionar múltiples ubicaciones	El sistema debe permitir administrar inventarios en diferentes sucursales desde un único sistema.

Requerimientos no funcionales

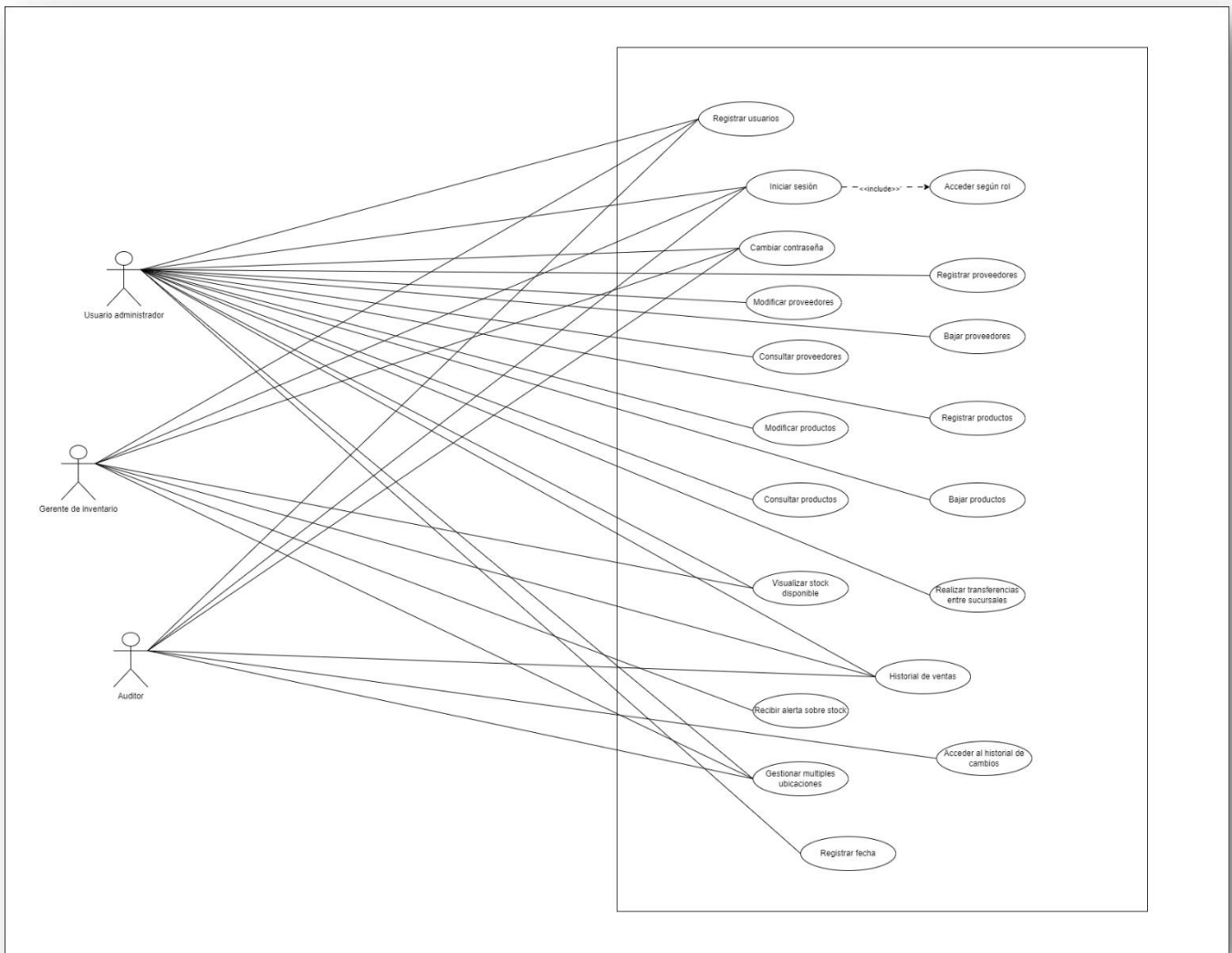
Requerimiento sistema	Descripción
RNF001: Disponibilidad	El sistema debe estar disponible las 24 horas del día para los usuarios. Se permite un tiempo de inactividad mínimo para realizar cambios necesarios en la plataforma.
RNF002: Rendimiento	El sistema debe realizar la carga de acciones de manera rápida y fluida. Se espera que cada acción se cargue en menos de 4 segundos, sin experimentar interrupciones ni retrasos.

RNF003: Seguridad	El sistema debe cumplir con estándares de seguridad de datos, comprometiéndose a proteger la información personal de la empresa y de los usuarios. La plataforma implementará medidas técnicas y organizativas adecuadas para prevenir el acceso no autorizado y la retención innecesaria de datos personales. La base de datos MySQL debe implementar medidas de seguridad robustas para proteger la información.
RNF004: Escalabilidad	El sistema debe ser escalable, permitiendo la adición de nuevos servicios o funciones sin realizar cambios significativos, adaptándose al crecimiento de Monarca Market.
RNF005: Mantenibilidad	El sistema debe permitir la actualización y mejora de las funcionalidades sin afectar el funcionamiento general del sistema.

En este punto de la consigna, también se elaboró el diagrama de casos de uso, que es una representación gráfica de los principales procesos que ocurren en un sistema. Para este caso, se basó en el sistema de gestión de stock integrado para el E-commerce y sucursal física del Monarca Market, mostrando los diferentes autores “Usuario administrador”, “Gerente de inventario”, “Auditor”.

Cada tipo de usuario tiene diferentes niveles de acceso en el sistema. Todas las interacciones son gestionadas por el sistema de gestión de stock integrado.

Este diagrama nos ayudara a comprender de manera clara y visual como interactúan los diferentes actores con el sistema, mostrando las acciones específicas que cada tipo de usuario puede llevar a cabo. Se ve detalladamente la funcionalidad del sistema y las distintas características disponibles para los usuarios.



Identificación de actores

- **Usuario Administrador:** Persona que utiliza el sistema para realizar diversas acciones, como gestionar usuarios, proveedores y productos. Como: Registrar usuarios, registrar proveedores, modificar productos, bajar productos, etc.
- **Gerente de Inventario:** Responsable de supervisar y gestionar el stock de productos, algunas funciones de este: recibir alertas de stock, historial de ventas, asegurando que la información del inventario esté actualizada.
- **Auditor:** Encargado de revisar y verificar el historial de cambios y ventas en el sistema.

Trazabilidad

Vamos a demostrar la trazabilidad de los requerimientos a partir de la especificación mencionada anteriormente. La trazabilidad es un artefacto esencial que permite rastrear un requisito particular a lo largo de todo el documento

Requerimiento	Caso de Uso	Actor principal	Paquete del Analisis	Comentario	Extends o include
RFC01	CU001	Usuario administrador, Gerente de inventario,	Registrar usuarios	Registro de nuevos usuarios asegurando la integridad	CU BASE
RFC02	CU002	Usuario administrador, Gerente de inventario,	Iniciar sesión	Iniciar sesion sin interrupciones	CU BASE
RFC03	CU003	Usuario administrador, Gerente de inventario,	Cambiar contraseña	Permitir el cambio de contraseña de manera segura.	CU BASE
RFC04	CU004	Usuario administrador	Acceder según el rol	Acceder al sistema en funcion al rol del usuario.	Include de CU002
RFC05	CU005	Usuario administrador	Registrar proveedores	Permitir el registro completo de proveedores con toda la informacion necesaria	CU BASE
RFC06	CU006	Usuario administrador	Modificar proveedores	Modificar los datos de proveedores ya registrados	CU BASE
RFC07	CU007	Usuario administrador	Bajar proveedores	Dar de baja a proveedores que ya no estan activos.	CU BASE
RFC08	CU008	Usuario administrador	Consultar proveedores	Consultar informacion de proveedores de manera eficiente	CU BASE
RFC09	CU009	Usuario administrador	Registrar productos	Permitir el registro completo de productos con toda la informacion necesaria	CU BASE
RFC10	CU010	Usuario administrador	Modificar productos	Modificar los datos de productos ya registrados	CU BASE
RFC11	CU011	Usuario administrador	Bajar productos	Dar de baja a productos que ya no estan activos.	CU BASE
RFC12	CU012	Usuario administrador	Consultar productos	Consultar informacion de productos de manera eficiente	CU BASE
RFC13	CU013	Usuario administrador, Gerente de inventario	Visualizar stock disponible	Visualizar el stock en tiempo real para la toma de decisiones	CU BASE
RFC14	CU014	Gerente de inventario	Recibir alerta sobre stock	Recibir alerta sobre niveles bajos de stock para evitar faltantes.	CU BASE
RFC15	CU015	Usuario administrador	Realizar transferencias entre sucursales	Facilitar el movimiento de stock entre diferentes sucursales	CU BASE
RFC16	CU016	Usuario administrador	Registrar fecha	Registrar de manera precisa la fecha de las acciones realizadas en el sistema	CU BASE
RFC17	CU017	Auditor	Acceder al historial de cambios:	Acceder al historial de cambios que se realizan en el sistema	CU BASE
RFC18	CU018	Usuario administrador, Gerente de inventario,	Historial de ventas	Realizar consulta de las ventas realizadas en diferentes periodos.	CU BASE
RFC19	CU019	Usuario administrador, Gerente de inventario,	Gestionar múltiples ubicaciones	Gestionar de manera centralizada las diversas ubicaciones físicas o virtuales de la empresa.	CU BASE

Especificaciones de casos de uso

En esta consigna también vamos a seleccionar 4 funcionalidades y realizar la especificación de los 4 casos de uso. La especificación de caso de uso implica detallar como un actor interactúa con el sistema, describiendo el proceso paso a paso. Las funcionalidades elegidas son “Registrar productos” “Visualizar Stock disponible”, “Recibir alerta sobre Stock”, “Bajar proveedores”. A continuación, proporcionare los detalles de estos casos de uso:

Caso de Uso	CU009 Registrar productos	
Actores	Usuario administrador	
Referencias	RFC09	
Descripcion	En este caso de uso, se describe como el sistema de gestion de stock permite a los usuarios registrar un nuevo producto en el sistema mediante una serie de pasos especificos.	
Precondicion	1) El usuario ha iniciado sesion en el sistema. 2) El usuario debe pertenecer al rol que tiene la funcion de agregar productos.	
Flujo Principal	1	El usuario se registra en el sistema de gestion de stock.
	2	El usuario ingresa al sistema, con el rol correcto.
	3	El usuario selecciona la opcion "Registrar producto".
	4	El sistema muestra un formulario para ingresar la informacion del producto.
	5	El usuario ingresa los siguientes datos: Nombre del producto, marca, descripcion, precio, codigo, cantidad de stock, categoria
	6	El usuario confirma el registro del producto.
Postcondicion	El nuevo producto queda registrado en el sistema, actualizandose el inventario.	
Flujo Alternativo	1	Si el usuario olvida su usuario o contraseña al abrir el sistema, puede restablecerla utilizando la opcion ¿Olvidaste la contraseña?.
	2	Si el usuario no visualiza imágenes mientras navega por el sistema, puede intentar abrir y cerrar el sistema para probar si se resuelve.
	3	Si el boton de "Registrar producto" no le responde al usuario puede intentar abrir y cerrar el sistema, actualizandolo o en caso que no se solucione reportar al soporte tecnico.
	4	Si el usuario experimenta dificultades y el sistema no carga, debe intentar reiniciar el sistema. Si el problema persiste, debe contactar al soporte técnico.
	5	Si el usuario experimenta un error de "Datos inválidos", debe proceder a revisar cada dato ingresado, corroborando si son correctos. Si los datos son correctos y el error persiste, se debe contactar al soporte técnico.
	6	Si el usuario presenta alguno de estos inconvenientes, puede intentar usar otra máquina para determinar si el problema es específico del dispositivo actual.
	7	Si el usuario encuentra algun problema al realizar algunas de las acciones anteriores debe comprobar su conexión a internet: 1. Verificar de donde proviene el problema: el usuario debe verificar si el problema es causado por el sistema o por la conexion a internet 2. Si el problema es causado por el sistema, el usuario intenta cerrar y abrir el sistema. 4.Si el problema es causado por la conexion a internet, el usuario debe verificar la conexion en su dispositivo
Excepciones	Errores tecnicos de el sistema a la hora de registrar un producto	

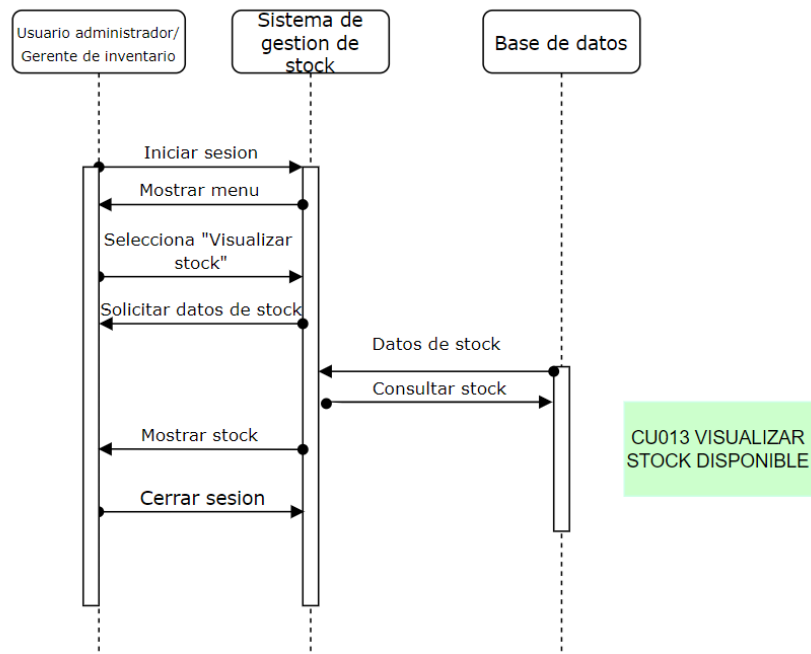
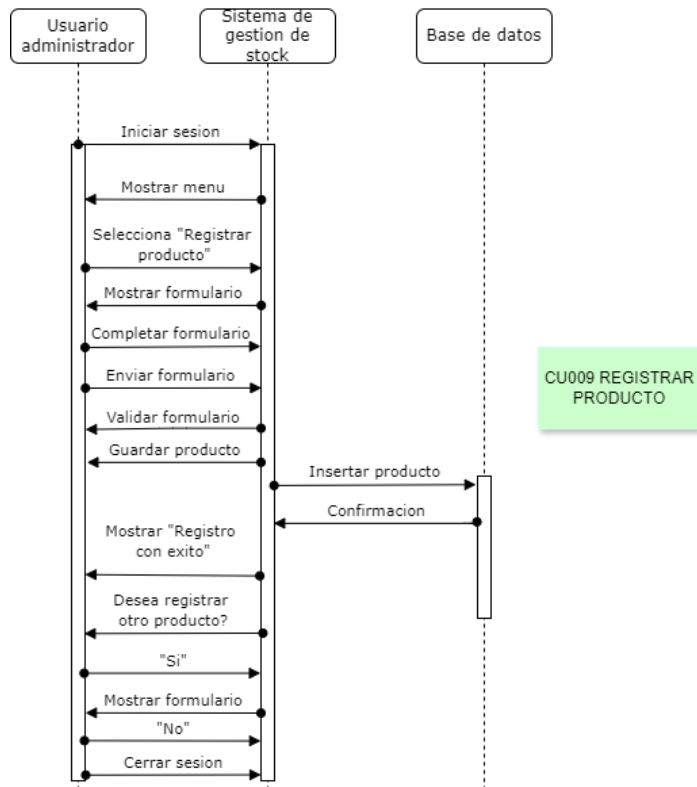
Caso de Uso	CU013 Visualizar Stock Disponible	
Actores	Usuario administrador, Gerente de inventario	
Referencias	RFC013	
Descripcion	En este caso de uso, se describe como el sistema de gestion de stock permite a los usuarios visualizar el stock de los productos registrados.	
Precondicion	1) El usuario ha iniciado sesion en el sistema. 2) El usuario debe pertenecer al rol que tiene la funcion visualizar stock disponible	
Flujo Principal	1	El usuario se registra en el sistema de gestion de stock.
	2	El usuario ingresa al sistema, con el rol correcto.
	3	El usuario selecciona la opcion "Visualizar Stock Disponible".
	4	El usuario visualiza una lista de productos con sus respectivas cantidades en stock.
	5	El usuario puede filtrar la lista por categoria o buscar un producto especifico.
	6	El usuario selecciona un producto para ver detalladamente las características del mismo: Nombre del producto, marca, descripcion, precio, codigo, cantidad de stock, categoria
Postcondicion	El usuario visualiza correctamente la informacion del stock.	
Flujo Alternativo	1	Si el usuario olvida su usuario o contraseña al abrir el sistema, puede restablecerla utilizando la opcion ¿Olvidaste la contraseña?.
	2	Si el usuario no visualiza imágenes mientras navega por el sistema, puede intentar abrir y cerrar el sistema para probar si se resuelve.
	3	Si el boton de "Visualizar stock disponible" no le responde al usuario puede intentar abrir y cerrar el sistema, actualizandolo o en caso que no se solucione reportar al soporte tecnico.
	4	Si el usuario experimenta dificultades y el sistema no carga, debe intentar reiniciar el sistema. Si el problema persiste, debe contactar al soporte técnico.
	5	Si el usuario experimenta un error de "Producto con error", debe proceder a revisar cada dato ingresado, corroborando si son correctos. Si los datos son correctos y el error persiste, se debe contactar al soporte técnico.
	6	Si el usuario presenta alguno de estos inconvenientes, puede intentar usar otra máquina para determinar si el problema es específico del dispositivo actual.
	7	Si el usuario encuentra algun problema al realizar algunas de las acciones anteriores debe comprobar su conexión a internet: 1. Verificar de donde proviene el problema: el usuario debe verificar si el problema es causado por el sistema o por la conexion a internet 2. Si el problema es causado por el sistema, el usuario intenta cerrar y abrir el sistema.
Excepciones	Si no hay productos registrados, el sistema muestra un mensaje indicando que no hay stock disponible	

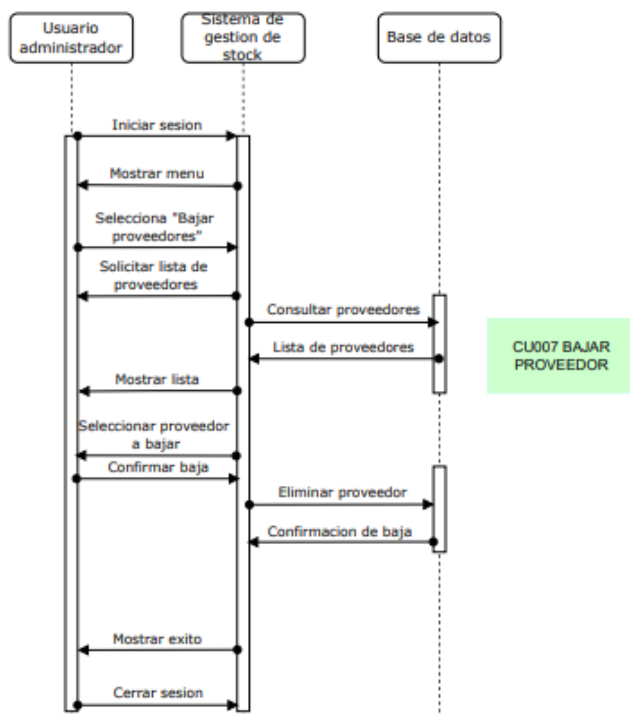
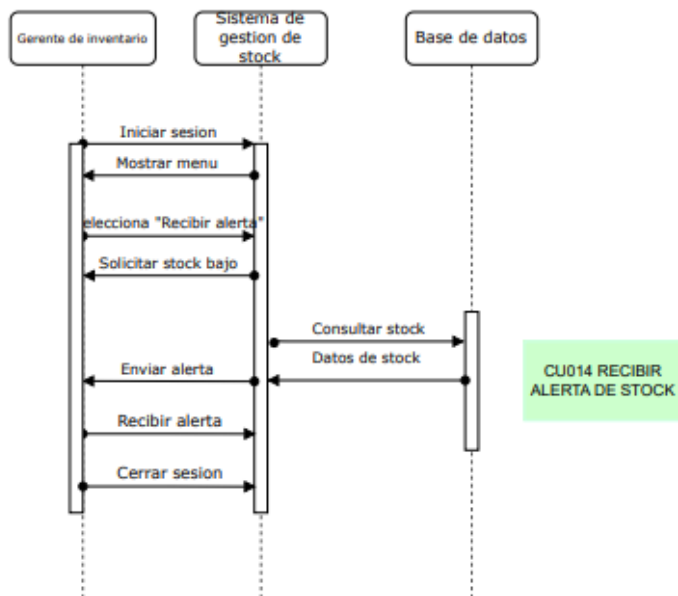
Caso de Uso	CU014 Recibir alerta sobre stock	
Actores	Gerente de inventario	
Referencias	RFC014	
Descripción	En este caso de uso, se describe como el sistema envía al gerente del inventario alertas cuando el stock de un producto está debajo de un umbral definido	
Precondición	1) El usuario ha iniciado sesión en el sistema. 2) El usuario debe pertenecer al rol que tiene la función recibir alerta sobre stock	
Flujo Principal	1	El usuario se registra en el sistema de gestión de stock.
	2	El usuario ingresa al sistema, con el rol correcto.
	3	El sistema envía una alerta al dispositivo registrado, cuando el stock de un producto cae.
	4	El usuario visualiza una notificación de stock en el dispositivo
	5	El usuario debe analizar ese producto y tomar una decisión
Postcondición	El usuario es notificado de la baja de stock	
Flujo Alternativo	1	Si el usuario olvida su usuario o contraseña al abrir el sistema, puede restablecerla utilizando la opción ¿Olvidaste la contraseña?.
	2	Si el usuario no visualiza imágenes mientras navega por el sistema, puede intentar abrir y cerrar el sistema para probar si se resuelve.
	3	Si el usuario no recibe una alerta de un producto con bajo stock, puede intentar abrir y cerrar el sistema, actualizándolo o en caso que no se solucione reportar al soporte técnico.
	4	Si el usuario experimenta dificultades y el sistema no carga, debe intentar reiniciar el sistema. Si el problema persiste, debe contactar al soporte técnico.
	5	Si el usuario experimenta un error de "Error en alerta", debe proceder a revisar cada dato ingresado, corroborando si son correctos. Si los datos son correctos y el error persiste, se debe contactar al soporte técnico.
	6	Si el usuario presenta alguno de estos inconvenientes, puede intentar usar otra máquina para determinar si el problema es específico del dispositivo actual.
	7	Si el usuario encuentra algún problema al realizar algunas de las acciones anteriores debe comprobar su conexión a internet: 1. Verificar de dónde proviene el problema: el usuario debe verificar si el problema es causado por el sistema o por la conexión a internet 2. Si el problema es causado por el sistema, el usuario intenta cerrar y abrir el sistema. 4. Si el problema es causado por la conexión a internet, el usuario debe verificar la conexión en su dispositivo
Excepciones	Si el sistema no puede enviar la alerta por problemas de conexión, registrar el error y reintentar más tarde.	

Caso de Uso	CU007 Bajar proveedores	
Actores	Usuario administrador	
Referencias	RFC07	
Descripción	En este caso de uso, se describe como el sistema de gestión de stock permite a los usuarios dar de baja un proveedor en el sistema mediante una serie de pasos	
Precondición	1) El usuario ha iniciado sesión en el sistema.	
Flujo Principal	1	El usuario se registra en el sistema de gestión de stock.
	2	El usuario ingresa al sistema, con el rol correcto.
	3	El usuario selecciona la opción "Baja de proveedores".
	4	El sistema muestra un formulario con los datos del proveedor.
	5	El usuario revisa los siguientes datos: Nombre del proveedor, marca, descripción, categoría.
	6	El usuario confirma la baja del proveedor.
Postcondición	El proveedor queda inactivo en el sistema.	
Flujo Alternativo	1	Si el usuario olvida su usuario o contraseña al abrir el sistema, puede restablecerla utilizando la opción ¿Olvidaste la contraseña?.
	2	Si el usuario no visualiza imágenes mientras navega por el sistema, puede intentar abrir y cerrar el sistema para probar si se resuelve.
	3	Si el botón de "Baja de proveedores" no le responde al usuario puede intentar abrir y cerrar el sistema, actualizándolo o en caso que no se solucione reportar al soporte técnico.
	4	Si el usuario experimenta dificultades y el sistema no carga, debe intentar reiniciar el sistema. Si el problema persiste, debe contactar al soporte técnico.
	5	Si el usuario experimenta un error de "Datos inválidos", debe proceder a revisar cada dato ingresado, corroborando si son correctos. Si los datos son correctos y el error persiste, se debe contactar al soporte
	6	Si el usuario presenta alguno de estos inconvenientes, puede intentar usar otra máquina para determinar si el problema es específico del dispositivo actual.
	7	Si el usuario encuentra algún problema al realizar algunas de las
Excepciones	Si el proveedor tiene productos asociados, el sistema no permite la baja y muestra un mensaje de error	

Etapa de análisis

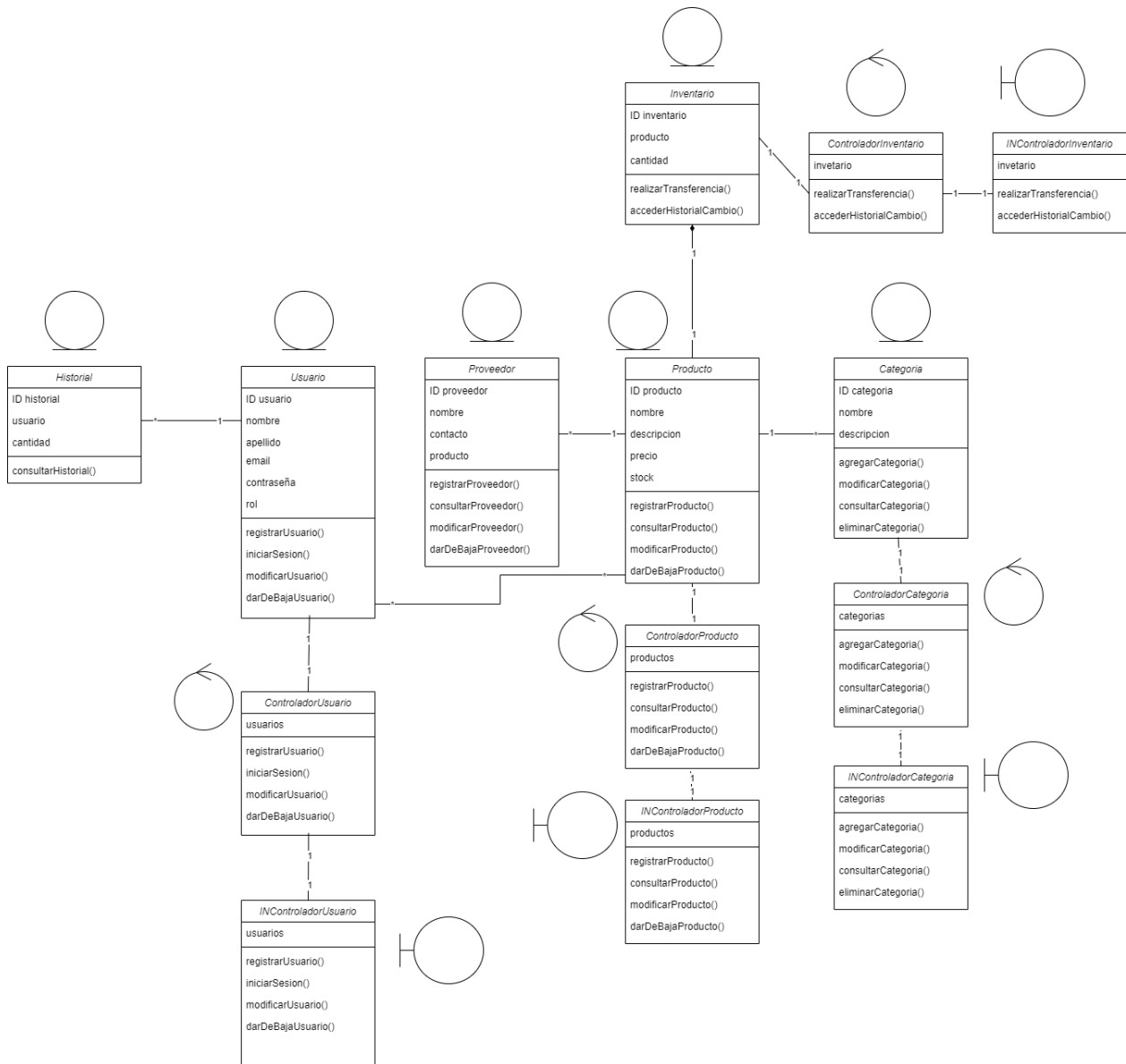
En esta etapa de análisis vamos a ver el diagrama de secuencia, una representación gráfica que muestra la interacción de objetos en un sistema a lo largo del tiempo. Estos diagramas capturan la secuencia de mensajes intercambiados entre objetos y el orden en que ocurren estas interacciones, presentándolas como líneas de vida verticales y flechas horizontales. Esto nos permite representar el comportamiento del sistema.



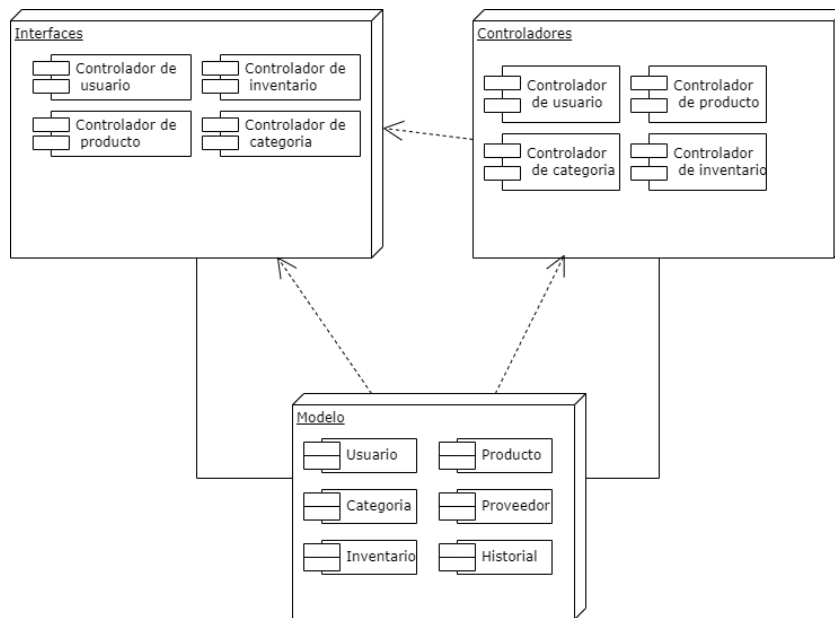


En esta consigna, presentaremos los diagramas de clases, los cuales representan la estructura estática de un sistema. En estos diagramas, se visualizan las clases del sistema y sus relaciones entre ellas. Cada clase define las características de los objetos que representa, incluyendo sus atributos y comportamientos. También hemos incluido los atributos correspondientes a cada clase, ya que estos son

propiedades esenciales de una clase de análisis y suelen ser necesarios para cumplir con las responsabilidades de cada clase: Entidad, Interfaz y Control.



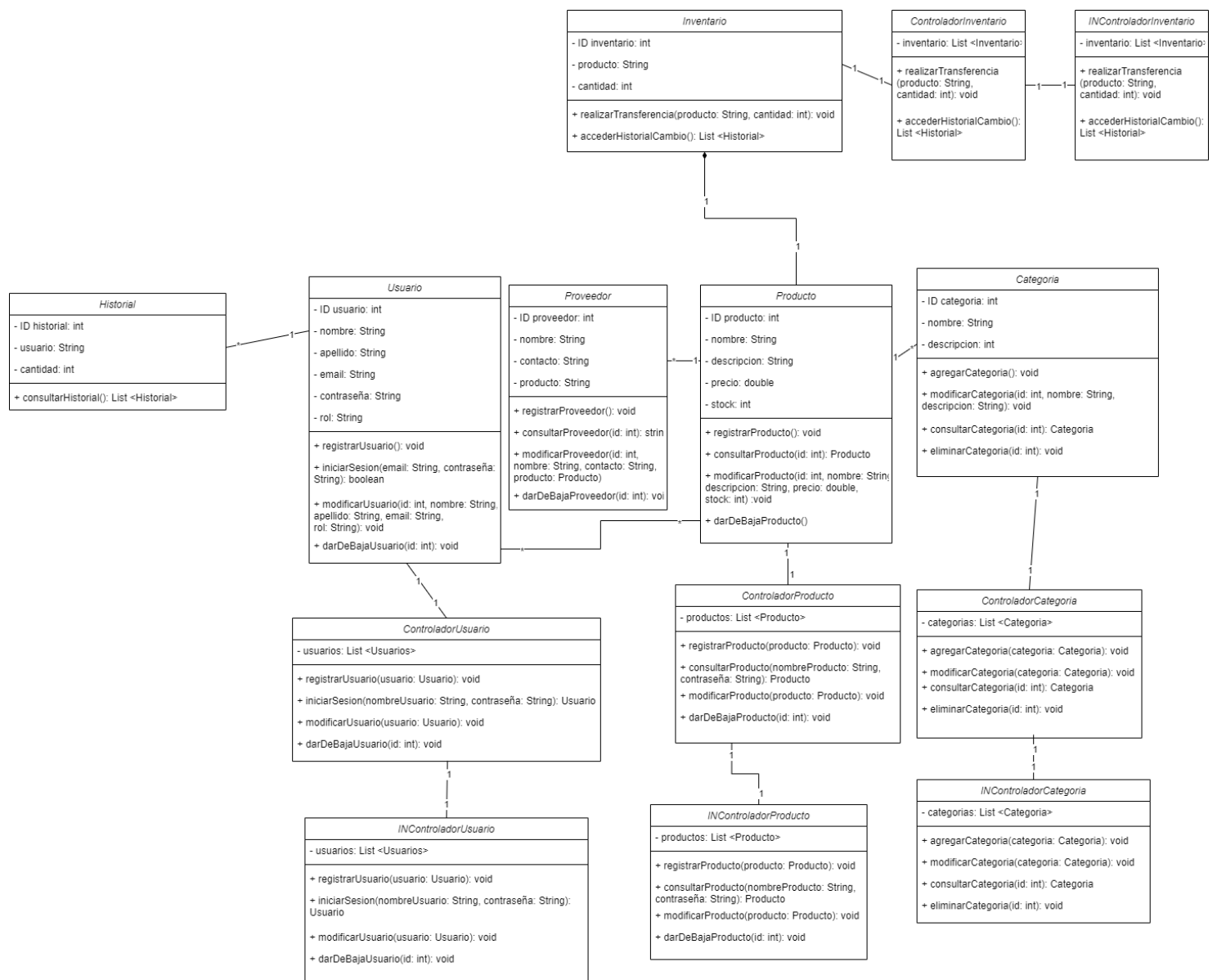
A continuación, realizaremos el diagrama de paquetes de nuestro sistema, este es un tipo de diagrama estructural en el Lenguaje de Modelado Unificado (UML) que se utiliza para representar la organización y disposición de varios elementos dentro de un sistema. Estos elementos pueden incluir clases, interfaces, componentes y otros paquetes. Esencialmente, es una manera de agrupar elementos relacionados en "paquetes" para simplificar sistemas complejos, de manera similar a como las carpetas ayudan a organizar archivos en un ordenador.



Etapas de diseño

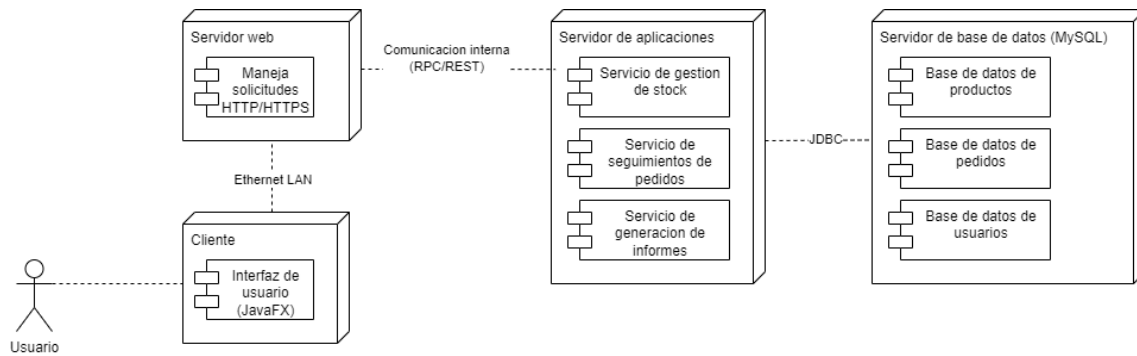
En esta consigna, examinaremos el diagrama de clases de diseño, el cual describe gráficamente las especificaciones de las clases de software y de las interfaces que participan en la solución de software de aplicación. Este tipo de diagramas incluyen clases, asociaciones, atributos, métodos, interfaces, información sobre los tipos de los atributos, entre otros.

A continuación, presentaremos los diagramas de clases de diseño de los casos de uso que hemos estado trabajando.



Etapa de implementación

En esta etapa, desarrollaremos el diagrama de despliegue, un tipo de diagrama UML que ilustra la arquitectura de ejecución de un sistema. Este diagrama incluye nodos que representan entornos de ejecución, tanto de hardware como de software, así como el middleware que los conecta. Nos ayudará a modelar la topología de hardware de nuestro sistema de gestión de inventario de stock de Monarca Market.



Etapa de pruebas

Plan de pruebas

A continuación, presentaremos un plan de pruebas básicos con algunos de los casos de usos del sistema de gestión de stock de Monarca Market.

- CP: caso de prueba.
- Tercera letra: tipo de prueba.
- Cuarta y quinta letra: número de prueba.

Caso de uso	Codigo de prueba	Tipo de prueba	Técnica propuesta	Observación
CU009 Registrar producto	CPF01	Funcional	Prueba de caja negra	Verificar que el producto se registra correctamente con todos los campos obligatorios.
CU009 Registrar producto	CPF02	Funcional	Prueba de caja negra	Probar el registro de un producto con campos opcionales vacíos
CU009 Registrar producto	CPS01	Seguridad	Prueba de penetración	Verificar que no se pueda ingresar al sistema sin tener autorización
CU009 Registrar producto	CPR01	Rendimiento	Prueba de carga	Evaluar el tiempo de respuesta del sistema al registrar múltiples productos simultáneamente

CU009 Registrar producto	CPU01	Usabilidad	Prueba de usabilidad	Asegurarse de que la interfaz de usuario sea fácil e intuitiva de utilizar
CU009 Registrar producto	CPC01	Componente	Prueba de integración	Probar la integración del módulo de base de datos con el módulo de registro de productos

Caso de uso	Codigo de prueba	Tipo de prueba	Técnica propuesta	Observación
CU014 Recibir alerta sobre stock	CPF01	Funcional	Prueba de caja negra	Verificar que se recibe una alerta cuando el stock cae por debajo del nivel mínimo
CU014 Recibir alerta sobre stock	CPF02	Funcional	Prueba de caja negra	Probar la recepción de alertas para múltiples productos simultáneamente
CU014 Recibir alerta sobre stock	CPS01	Seguridad	Prueba de penetración	Asegurarse de que las alertas no puedan ser interceptadas por usuarios no autorizados
CU014 Recibir alerta sobre stock	CPC01	Componente	Prueba de integración	Probar la integración entre el módulo de gestión de stock y el sistema de alertas
CU014 Recibir alerta sobre stock	CPSS	Sistema	Prueba de sistema	Verificar que el sistema completo funcione correctamente al recibir y gestionar alertas de stock

Caso de prueba

Es un conjunto de condiciones que se diseñan para verificar que una determinada función o componente de un desarrollo funcione de acuerdo con lo previsto. En nuestro caso vamos a demostrar el caso de uso “CU014: Recibir alerta sobre stock”

Caso de prueba: **Verificar Alerta de Stock**

Descripción: Este caso de prueba verifica que el sistema de inventario de Monarca Market muestre correctamente alertas relacionadas con el nivel de stock de un producto

Requerimiento	Descripción
RFS21	El sistema debe mostrar un mensaje de alerta si el stock de un producto cae por debajo de 5 unidades
RFS22	El sistema debe mostrar un mensaje informativo si el stock de un producto es igual o mayor a 5 unidades
RFS23	El sistema debe permitir a los usuarios establecer niveles de stock mínimo personalizados para cada producto

Se establece que un nivel adecuado de stock para cada producto en Monarca Market es de 500 unidades. Este valor representa el punto óptimo de inventario, asegurando que haya suficiente producto disponible para satisfacer la demanda sin incurrir en costos innecesarios de almacenamiento o riesgo de obsolescencia.

Clases de equivalencias válidas

CEV1: Stock bajo

Rango: 0 – 5 unidades

Descripción: Nivel de stock que requiere atención para evitar rupturas de inventario

CEV2: Stock adecuado

Rango: 6 – 100 unidades

Descripción: Nivel de stock que se considera suficiente para las operaciones diarias.

CEV3: Stock alto

Rango: 101 – 500 unidades

Descripción: Nivel de stock que puede indicar exceso, pero dentro de los límites de capacidad.

Clases de equivalencias inválidas

CEI1: Stock negativo**Rango:** valor < 0**Descripción:** Situación que indica error en el sistema.**CEI2: Exceso de stock****Rango:** valor > 500**Descripción:** Indica que el stock superar la capacidad de almacenamiento prevista.**Análisis de valores de frontera**

Para aplicar el análisis de valores de frontera al sistema de inventario de Monarca Market, primero identificaremos los valores límites para cada clase de equivalencia definida anteriormente.

Clase de equivalencia	Limite frontera	Límite inferior	Límite superior
CEV1: Stock bajo	5	4	6
CEV2: Stock adecuado	500	499	501
CEI1: Stock negativo	0	-1	1
CEI2: Exceso de stock	500	499	501

Con base en los límites definidos, procederemos a completar el siguiente cuadro que representa el comportamiento del sistema en cada caso.

Dato de entrada	Comportamiento esperado	Mensaje emitido por el sistema
-1	Error en el sistema	"Stock negativo no permitido"
0	Alerta stock critico	"Stock insuficiente, reabastecer"
1	Alerta stock critico	"Stock insuficiente, reabastecer"
4	Alerta stock bajo	"Stock bajo, considerar reabastecimiento"
5	Alerta stock bajo	"Stock bajo, considerar reabastecimiento"
6	Stock normal	"Stock adecuado"
499	Stock normal	"Stock adecuado"
500	Stock normal	"Stock optimo alcanzado"
501	Alerta de exceso de stock	"Exceso de stock, considerar reducir inventario"

Realicé los caso de pruebas basándome en un solo caso de uso (Verificar alerta de stock), ya que este es uno de los principales. Los demás casos son similares en cuanto a las situaciones que se llevarían a cabo, por lo que las pruebas también se pueden realizar de manera análoga.

Evaluación de pruebas y tratamiento de defectos

El componente específico que vamos a probar es el método registrarProducto() de la clase Producto, el cual se encarga de registrar un producto específico.

Procedimiento de prueba

El procedimiento de prueba define los pasos a seguir para verificar que el método registrarProducto() funcione correctamente

Pasos:

1. Preparar el entorno
 - Configurar la base de datos de prueba
 - Inicializar los objetos necesarios para el registro de productos
2. Ingresar datos:
 - Proporcionar el nombre, precio, cantidad y descripción del producto
3. Verificar resultados
 - Comprobar que el producto se haya almacenado correctamente en la base de datos.
 - Comprobar cada campo del registro.
4. Valores limites
 - Intentar registrar productos con entradas inválidas, productos duplicados y valores en los límites permitidos.
5. Pruebas de rendimiento
 - Verificar que el registro se complete en un tiempo razonable y con la información correcta.

Evaluación de prueba


Caso de prueba	Descripción	Resultado esperado	Estado	Observaciones
CU009 Registrar producto (valido)	Registrar un producto con datos validos	Producto registrado correctamente en la base de datos	Aprobado	
CU009 Registrar producto (sin nombre)	Intentar registrar un producto sin nombre	Debería devolver un error o mensaje invalido	Fallido	No hay validación
CU009 Registrar producto (precio negativo)	Intentar registrar un producto con precio negativo	Debería devolver un error o mensaje invalido	Fallido	
CU009 Registrar producto (producto duplicado)	Intentar registrar un producto duplicado	Debería devolver un error por duplicado	Fallido	Validando manejo de duplicaciones

Tratamiento de defectos

Defecto	Descripción del defecto	Prioridad	Estado	Acción realizada	observación
D01	No se valida que el campo nombre este vacío	Alta	Abierto	Se notifico al equipo de programadores	Pendiente de validación
D02	El sistema no detecta productos duplicados	Media	Abierto	Revision del que ingresa los datos	Se requiere test del personal

Interfaz grafica

En esta etapa, desarrollaremos un prototipo de la interfaz con el objetivo principal de mejorar la experiencia del usuario. Este prototipo es una representación preliminar, tanto visual como funcional, de la interfaz de la aplicación en desarrollo. Su propósito es probar y evaluar la apariencia y la interacción antes de avanzar hacia el desarrollo del producto final.



Username

Password

☐ Remember me

[Forgot my password](#)

Sistema de Gestion
Monarca Market

HOME > PROVEEDORES

REGISTRAR
PROVEEDORES

MODIFICAR
PROVEEDORES

CONSULTAR
PROVEEDORES

DAR DE BAJA
PROVEEDORES

Sistema de Gestion
Monarca Market

HOME > ADMINISTRACION

STOCK DISPOBIBLE

RECIBIR ALERTA
SOBRE STOCK

TRANSFERENCIAS

HISTORIAL
DE CAMBIOS

HISTORIAL DE
VENTAS

Sistema de Gestion
Monarca Market

HOME > PRODUCTOS

REGISTRAR
PRODUCTOS

MODIFICAR
PRODUCTOS

CONSULTAR
PRODUCTOS

DAR DE BAJA
PRODUCTOS

**Sistema de Gestion
Monarca Market**

HOME > REGISTRAR PROVEEDORES

ID. Prv

Nombre

Contacto

Producto

Sucursal

ACEPTAR MODIFICAR CANCELAR

Definición de base de datos para el prototipo

Para implementar el sistema de gestión de inventario de stock de Monarca Market, es esencial contar con una base de datos relacional. Utilizamos MySQL, que nos proporciona un excelente rendimiento, flexibilidad y escalabilidad. Estos atributos son fundamentales para avanzar de manera ordenada en la implementación completa del proyecto. A continuación, procederemos a definir la estructura de la base de datos.

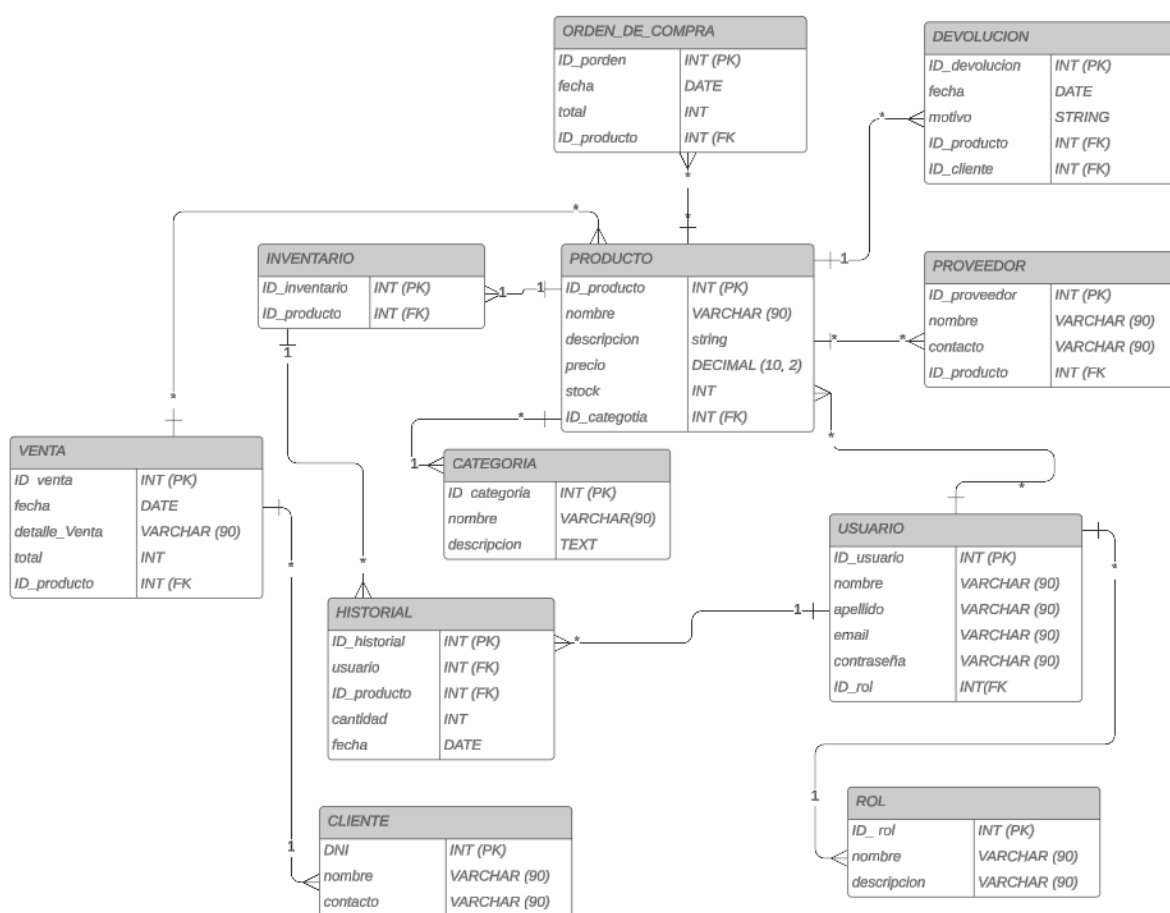
Entidades

- **Producto:** ID_producto (PK), nombre, descripción, precio, stock, ID_categoria (FK)
- **Categoría:** ID_categoria (PK), nombre, descripción
- **Proveedor:** ID_proveedor (PK), nombre, contacto, ID_producto (FK)
- **Usuario:** ID_usuario (PK), nombre, apellido, email, contraseña, rol
- **Historial:** ID_historial (PK), usuario, cantidad
- **Inventario:** ID_inventario (PK), ID_producto (FK)
- **Orden de compra:** ID_porden (PK), fecha, total, ID_producto (FK)
- **Devolución:** ID_devolucion (PK), fecha, motivo, ID_producto (FK), ID_cliente (FK)
- **Inventario:** ID_inventario (PK), ID_producto (FK)
- **Venta:** ID_venta (PK), fecha, detalle_venta, total, ID_producto (FK)
- **Cliente:** DNI (PK), nombre, contacto
- **Rol:** ID_rol (PK), nombre, descripción

Todo esto lo llevaremos en MySQL, que nos permitirá almacenar y acceder a los datos a través de múltiples motores de almacenamiento, asegurando así un sistema robusto y eficiente.

Diagrama entidad relacion

En esta consigna, crearemos un diagrama de entidad-relación (DER), una representación gráfica de la estructura de una base de datos. Este diagrama muestra cómo las entidades o tablas se relacionan e interactúan, destacando entidades, atributos, relaciones, claves primarias y foráneas, y cardinalidad. Así, se facilita la visualización y posterior implementación en la base de datos.



Creación de tablas

En esta etapa, creamos las diferentes identidades necesarias para el sistema. A continuación, mostraré la creación de la tabla "Producto", donde se definen todos sus atributos, identificando las claves primarias y foráneas, así como asignando la cardinalidad correspondiente. La elaboración del diagrama de entidad-relación nos ayuda a mantener un orden en el diseño y estructura de la base de datos.

```
• CREATE TABLE Producto (  
    ID_producto INT PRIMARY KEY,  
    nombre VARCHAR(90),  
    descripcion VARCHAR(255),  
    precio DECIMAL(10, 2),  
    stock INT,  
    ID_categoria INT,  
    FOREIGN KEY (ID_categoria) REFERENCES Categoria(ID_categoria)  
);
```

También realizamos la inserción de datos de prueba en las tablas para verificar que el modelo de datos y las funcionalidades de la base de datos funcionen correctamente antes de implementar la aplicación en un entorno de producción.

```
-- Insertar datos en Producto  
INSERT INTO Producto (ID_producto, nombre, descripcion, precio, stock, ID_categoria)  
VALUES (1, 'Yogur', 'Frutilla', 50.50, 100, 1);
```

A continuación, realizamos consultas para identificar cómo se obtienen los datos y verificar si las consultas realizadas devuelven la información que necesitamos.

```
-- Consultar ventas y productos vendidos  
SELECT Venta.ID_venta, Producto.nombre, Venta.total  
FROM Venta  
JOIN Producto ON Venta.ID_producto = Producto.ID_producto;
```

Por último, realizaremos el borrado de las tablas, asegurándonos de que se hayan eliminado correctamente.

```
-- Eliminar el producto  
DELETE FROM Producto WHERE ID_producto = 1;
```

A continuación, dejo el enlace del repositorio para que puedan revisar las diferentes tablas y probar su funcionamiento:

<https://github.com/CamilaLev07/SistemaInventario1/tree/main>

Requerimientos de comunicación del sistema

Este sistema ha sido diseñado para gestionar de manera eficiente el inventario de stock de Monarca Market, facilitando el acceso a la información para los empleados del supermercado. Es fundamental establecer una comunicación efectiva entre los diferentes componentes del sistema para garantizar la integridad de los datos y optimizar la eficiencia de cada operación. Esto permitirá integrar de manera fluida tanto el stock físico como el del comercio electrónico de todas las sucursales.

Para el desarrollo del sistema de gestión de stock, que integrará las sucursales físicas y el comercio electrónico, se requerirán componentes como un servidor web, una base de datos MySQL, interfaces de usuario (JavaFX) y servicios externos, como APIs de proveedores.

La comunicación del sistema se dividirá en dos tipos. Por un lado, la comunicación interna, que incluirá la interacción entre el servidor del sistema y la base de datos, así como servicios de gestión de stock, seguimiento de pedidos y generación de informes. Por otro lado, la comunicación externa se referirá a la interacción con APIs de terceros para el manejo de proveedores y servicios de pago.

Además, el sistema deberá cumplir con ciertos requerimientos de rendimiento. Se espera que las respuestas de la API tengan una latencia menor a 250 ms y que el ancho de banda tenga la capacidad de manejar múltiples solicitudes simultáneas sin degradación del servicio. También es crucial que el sistema esté disponible 24 horas al día, aunque se programarán momentos específicos para realizar actualizaciones.

En cuanto a los protocolos y estándares, utilizaremos HTTP y HTTPS. El primero es un conjunto de reglas para la comunicación cliente-servidor, mientras que el segundo es su versión segura, esencial para la comunicación entre el sistema y los usuarios. También emplearemos JDBC para conectar el servidor de aplicaciones con la base de datos MySQL. Para garantizar la seguridad, implementaremos TLS/SSL, que permite verificar la identidad y establecer conexiones de red cifradas.

En relación con la tipología de red, se optará por una topología en estrella, donde todos los componentes estarán conectados a un servidor central. La infraestructura física incluirá servidores locales para asegurar una mejor escalabilidad y un almacenamiento seguro de los datos, así como routers y switches para mantener la conectividad. Se utilizarán conexiones Ethernet para enlaces cableados y Wi-Fi para conexiones inalámbricas.

Finalmente, se irán utilizando diferentes protocolos y conexiones según sea necesario para llevar a cabo nuestro sistema de gestión de inventario de stock de Monarca Market de manera segura, escalable y eficiente.

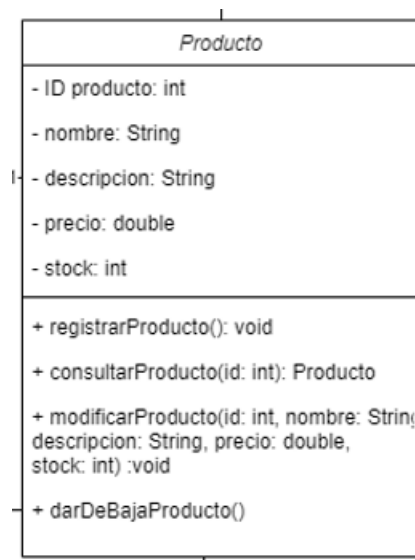
Desarrollo del sistema utilizando Java

En esta sección del proyecto, llevaremos a cabo el desarrollo de nuestro sistema utilizando Java, un lenguaje de programación versátil y compatible con una amplia variedad de sistemas operativos. Una de sus características distintivas es que es tanto compilado como interpretado, lo que significa que convierte automáticamente el código en instrucciones de máquina. Esto proporciona varias ventajas, como ser un lenguaje de alto nivel, orientado a objetos y de alto rendimiento. Además, su capacidad de desarrollo escalable nos permite crear un sistema de gestión de inventario para Monarca Market.

De acuerdo con los requerimientos no funcionales especificados en la primera parte del proyecto, es fundamental que realicemos el trabajo en Java. Nuestro objetivo es lograr un código estructurado, legible y modular, lo que facilitará la comprensión, el mantenimiento y la escalabilidad del proyecto.

El desarrollo abarcará la implementación completa del sistema. Comenzaremos analizando los diferentes diagramas que hemos elaborado, incluyendo el diagrama de clases de diseño y el diagrama de entidad-relación. A partir de estos diagramas, procederemos a implementar cada clase y sus métodos correspondientes.

A continuación, presentaremos una captura de una de las clases que hemos creado, seguida de una explicación detallada de su código. Esta explicación incluirá la funcionalidad de cada método y su interacción con otros componentes del sistema.



Aquí vemos que esta clase tiene cinco atributos:

- **ID producto:** int
- **Nombre:** String
- **Descripción:** String
- **Precio:** Double
- **Stock:** Int

Y los métodos:

- **registrarProducto():** void
- **consultarProducto(id: int):** Producto
- **modificarProducto** (id: int, nombre: String, descripción: String, precio: double, stock: int): void
- **darDeBajaProducto** ()

```

public abstract class Producto {           //clase abstracta
    private int id;
    private String nombre;
    private double precio;                ATRIBUTOS
    private int stock;

    // Constructor
    public Producto(int id, String nombre, double precio, int stock) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.stock = stock;              CONSTRUCTOR
    }
}

```

A continuación, analizaremos los métodos de esta clase, comenzando con **agregarProducto()**, que se encarga de registrar un producto.

Este método inicia con la lógica para determinar el tipo de producto: si es un producto común o electrónico. Primero, se solicita al usuario que indique el tipo de producto. Luego, se consulta el ID del producto y se implementan excepciones para evitar que se ingrese un producto con un ID que ya existe. Si el usuario intenta ingresar un ID duplicado, el sistema no permitirá el registro de este.

Además, se valida la entrada del ID para asegurarse de que sea un número entero (INT). En caso de que se ingrese un símbolo o carácter no válido, se solicitará al usuario que lo ingrese nuevamente.

Una vez completadas estas validaciones, se procede a solicitar el nombre del producto, su precio, el stock disponible y, en el caso de los productos electrónicos, la marca. Si toda la información se completa correctamente, se registra exitosamente el producto.

```

public void agregarProducto(Scanner scanner) {
    System.out.println(x:"1. Producto Normal");
    System.out.println(x:"2. Producto Electrónico");
    //obtener el tipo de producto

    int tipoProducto = obtenerNumero(scanner, mensaje:"Ingrese tipo: ");

    Producto producto = null;
    // Crear un ProductoNormal o ProductoElectrónico basado en la selección

    if (tipoProducto == 1) {
        int idProducto = obtenerNumero(scanner, mensaje:"Ingrese ID de Producto: ");
        if (idProductoExiste(idProducto)) {
            System.out.println(x:"Error: El ID del producto ya existe.");
            return;
        }
        System.out.print(s:"Ingrese Nombre de Producto: ");
        String nombreProducto = scanner.next();
        double precioProducto = obtenerNumeroDouble(scanner, mensaje:"Ingrese Precio de Producto: ");
        int stockProducto = obtenerNumero(scanner, mensaje:"Ingrese Stock: ");

        producto = new ProductoNormal(idProducto, nombreProducto, precioProducto, stockProducto);
    } else if (tipoProducto == 2) {
        int idProducto = obtenerNumero(scanner, mensaje:"Ingrese ID de Producto: ");
        if (idProductoExiste(idProducto)) {
            System.out.println(x:"Error: El ID del producto ya existe.");
            return;
        }
        System.out.print(s:"Ingrese Nombre de Producto: ");
        String nombreProducto = scanner.next();
        double precioProducto = obtenerNumeroDouble(scanner, mensaje:"Ingrese Precio de Producto: ");
        int stockProducto = obtenerNumero(scanner, mensaje:"Ingrese Stock: ");

        System.out.print(s:"Ingrese Marca: ");
        String marca = scanner.next();

        producto = new ProductoElectrónico(idProducto, nombreProducto, precioProducto, stockProducto, marca);
    } else {
        System.out.println(x:"Tipo de producto no válido. Producto no agregado.");
        return; // Salir del método si el tipo no es válido
    }

    productos.add(producto);
    System.out.println(x:"Producto agregado exitosamente.");
}

```

A continuación, veremos el método **consultarStock()**, que se refiere al método consultar Producto. Este método inicia preguntando al usuario si desea consultar un producto específico o todos los productos disponibles.

Si el usuario elige consultar un producto específico, deberá ingresar el ID correspondiente. Si se encuentra el producto, el sistema proporcionará todos los datos relacionados con él. En caso de que el ID ingresado no corresponda a ningún producto registrado, el sistema mostrará un cartel que el producto no fue encontrado.

Además, si no hay productos registrados en el sistema, se mostrará un mensaje indicando que no se han encontrado productos.


```
// Método para consultar el stock de productos
public void consultarStock(Scanner scanner) {
    System.out.println(x:"Consulta de stock:");
    System.out.println(x:"Seleccione una opción:");
    System.out.println(x:"1. Consultar producto específico");
    System.out.println(x:"2. Consultar todos los productos");

    int opcion = obtenerNumero(scanner, mensaje:"Ingrese opción: ");

    if (opcion == 1) {
        // Consultar producto específico
        int idProducto = obtenerNumero(scanner, mensaje:"Ingrese ID de Producto: ");
        Producto producto = buscarProductoPorId(idProducto);

        if (producto != null) {
            System.out.println("Producto encontrado: " + producto.toString() + ", Tipo: " + producto.getTipoProducto());
        } else {
            System.out.println(x:"Producto no encontrado.");
        }
    } else if (opcion == 2) {
        // Consultar todos los productos
        System.out.println(x:"Lista de todos los productos:");
        if (productos.isEmpty()) {
            System.out.println(x:"No hay productos registrados.");
        } else {
            for (Producto p : productos) {
                System.out.println(p.toString() + ", Tipo: " + p.getTipoProducto());
            }
        }
    } else {
        System.out.println(x:"Opción no válida.");
    }
}
}
```

A continuación, procederemos al método **modificarProducto()**. Este método permite modificar cualquiera de los datos ingresados anteriormente en la clase Producto. Si el usuario no desea cambiar algún dato específico, puede dejar ese campo en blanco. De esta manera, el sistema solo actualizará la información que se haya proporcionado, manteniendo los datos que se dejaron sin modificar.

```

// Método en GestionInventario para modificar un producto
public void modificarProducto(int id, Scanner scanner) {
    Producto producto = buscarProductoPorId(id);
    if (producto != null) {
        System.out.println("Producto encontrado: " + producto);

        // Modificar los datos
        System.out.print(s:"Ingrese nuevo nombre (dejar en blanco para no cambiar): ");
        String nuevoNombre = scanner.nextLine();
        if (!nuevoNombre.isEmpty()) {
            producto.setNombre(nuevoNombre);
        }

        System.out.print(s:"Ingrese nuevo precio (dejar en blanco para no cambiar): ");
        String precioInput = scanner.nextLine();
        if (!precioInput.isEmpty()) {
            double nuevoPrecio = Double.parseDouble(precioInput);
            producto.setPrecio(nuevoPrecio);
        }

        System.out.print(s:"Ingrese nuevo stock (dejar en blanco para no cambiar): ");
        String stockInput = scanner.nextLine();
        if (!stockInput.isEmpty()) {
            int nuevoStock = Integer.parseInt(stockInput);
            producto.setStock(nuevoStock);
        }

        System.out.println(x:"Producto modificado exitosamente.");
    } else {
        System.out.println(x:"Producto no encontrado.");
    }
}

```

Por último, analizaremos el método **eliminarProducto()**. Este método permite eliminar un producto que ya ha sido ingresado en el sistema, solicitando al usuario que indique el ID del producto que desea eliminar. Una vez proporcionado el ID, el sistema procederá a eliminar el producto correspondiente.

```

// Método en GestionInventario para eliminar un producto
public void eliminarProducto(int id) {
    Producto producto = buscarProductoPorId(id);
    if (producto != null) {
        productos.remove(producto);
        System.out.println(x:"Producto eliminado exitosamente.");
    } else {
        System.out.println(x:"Producto no encontrado.");
    }
}

```

Este trabajo me ayudó a repasar los conocimientos básicos de la programación orientada a objetos e implementar los cuatro pilares de esta.

Comenzamos analizando el concepto y la aplicación del **encapsulamiento**, que es un mecanismo para agrupar datos y métodos dentro de una estructura que oculta la implementación del objeto. Esto impide el acceso a los datos por cualquier medio que no sean los servicios propuestos. Solo se exponen los elementos necesarios a través de métodos públicos, como getters y setters. En la clase Producto, los atributos están declarados como private, lo que ejemplifica este principio.

```
private int id;  
private String nombre;  
private double precio;  
private int stock;
```

Luego, implementamos métodos públicos que permiten acceder y modificar los atributos privados. Los getters nos permiten obtener el valor de los atributos, mientras que los setters nos permiten modificar el valor de estos atributos.

```
// Métodos getters y setters  
public int getId() {  
    return id;  
}  
  
public String getNombre() {  
    return nombre;  
}  
  
public double getPrecio() {  
    return precio;  
}  
  
public int getStock() {  
    return stock;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public void setPrecio(double precio) {  
    this.precio = precio;  
}  
  
public void setStock(int stock) {  
    this.stock = stock;  
}
```

Luego, analizamos el concepto de **herencia**, que se aplica en la clase **ProductoElectrónico**. Este concepto permite crear clases que reutilizan, extienden y modifican el comportamiento definido en otras clases. Gracias a la herencia, **ProductoElectrónico** puede aprovechar los atributos y métodos de la clase **Producto**, al mismo tiempo que implementa su propia funcionalidad específica.

```
public class ProductoElectrónico extends Producto { //herencia de la clase Producto
    private String marca;

    public ProductoElectrónico(int id, String nombre, double precio, int stock, String marca) {
        super(id, nombre, precio, stock);
        this.marca = marca;
    }

    @Override
    public String getDetalles() {
        return "ID: " + getId() + ", Nombre: " + getNombre() + ", Precio: "
            + getPrecio() + ", Stock: " + getStock() + ", Marca: " + marca + " (" + obtenerEstadoStock() + ")";
    }

    @Override
    public String getTipoProducto() {
        return "Producto Electrónico";
    }

    @Override
    public String toString() {
        return getDetalles();
    }
}
```

En este código, la superclase **Producto** define los atributos y métodos que son comunes a todos los productos, incluyendo también los métodos abstractos. Por otro lado, la clase **ProductoElectrónico** extiende la clase **Producto**, lo que significa que hereda todos los atributos y métodos de **Producto**. Esta relación se establece en el código mediante la palabra clave **extends**.

A continuación, analizamos el concepto de **polimorfismo**, que se refiere a la capacidad de los objetos para comportarse de diferentes maneras según el contexto, lo que aporta flexibilidad y simplicidad al código. En nuestro caso, el polimorfismo se aplica en los siguientes fragmentos del código:

```

public abstract class Producto {           //clase abstracta
    private int id;
    private String nombre;
    private double precio;
    private int stock;

    // Constructor
    public Producto(int id, String nombre, double precio, int stock) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.stock = stock;
    }

    // Método abstracto para obtener detalles del producto
    public abstract String getDetalles();

    // Método abstracto para obtener el tipo de producto
    public abstract String getTipoProducto();

    // Método para calcular un descuento
    public double calcularDescuento(double porcentaje) {
        return precio * (1 - porcentaje / 100);
    }

    // Método auxiliar para determinar el estado del stock
    protected String obtenerEstadoStock() {
        if (stock < 0) {
            return "Stock negativo";
        } else if (stock >= 0 && stock <= 5) {
            return "Stock bajo";
        } else if (stock >= 6 && stock <= 100) {
            return "Stock adecuado";
        } else if (stock >= 101 && stock <= 500) {
            return "Stock alto";
        }
    }
}

public class ProductoElectrónico extends Producto {           //herencia de la clase Producto
    private String marca;

    public ProductoElectrónico(int id, String nombre, double precio, int stock, String marca) {
        super(id, nombre, precio, stock);
        this.marca = marca;
    }

    @Override
    public String getDetalles() {
        return "ID: " + getId() + ", Nombre: " + getNombre() + ", Precio: "
            + getPrecio() + ", Stock: " + getStock() + ", Marca: " + marca + " (" + obtenerEstadoStock() + ")";
    }

    @Override
    public String getTipoProducto() {
        return "Producto Electrónico";
    }

    @Override
    public String toString() {
        return getDetalles();
    }
}

```

```

public class ProductoNormal extends Producto {
    public ProductoNormal(int id, String nombre, double precio, int stock) {
        super(id, nombre, precio, stock);
    }

    @Override
    public String getDetalles() {
        return "ID: " + getId() + ", Nombre: " + getNombre() + ", Precio: " + getPrecio() +
            ", Stock: " + getStock() + " (" + obtenerEstadoStock() + ")";
    }

    @Override
    public String getTipoProducto() {
        return "Producto Normal";
    }

    @Override
    public String toString() {
        return getDetalles();
    }
}

```

En el código, observamos un ejemplo de **polimorfismo** en la forma en que las subclases `ProductoNormal` y `ProductoElectrónico` sobrescriben los métodos abstractos `getDetalles()` y `getTipoProducto()` de la clase abstracta `Producto`. Ambos métodos se llaman de la misma manera, pero cada subclase proporciona su propia implementación, adaptada a las características particulares de su tipo.

El polimorfismo se manifiesta cuando un objeto puede ser una instancia de `ProductoNormal` o `ProductoElectrónico`, dependiendo de la opción que elija el usuario al momento de agregar un producto. Por ejemplo, el método `productos.add(producto)` es capaz de operar sobre diferentes tipos de productos, pero el comportamiento puede variar según la clase específica del objeto que se esté añadiendo (ya sea un producto normal o un producto electrónico).

Otro ejemplo claro de polimorfismo es el método `buscarProductoPorId`. Este método retorna un objeto del tipo `Producto`, pero en realidad puede devolver un objeto de cualquiera de las subclases de `Producto`, como `ProductoNormal` o `ProductoElectrónico`, dependiendo del resultado de la búsqueda. A pesar de que el tipo devuelto es genérico (`Producto`), el objeto real puede ser de una subclase específica, y sus métodos sobrescritos serán ejecutados de acuerdo con esa clase.

El polimorfismo en este código permite que el mismo método pueda interactuar con diferentes tipos de productos (`ProductoNormal` o `ProductoElectrónico`), brindando flexibilidad y extensibilidad al sistema.

```
// Método privado para buscar un producto por ID
private Producto buscarProductoPorId(int idProducto) {
    for (Producto p : productos) {
        if (p.getId() == idProducto) {
            return p;
        }
    }
    return null; // Retorna null si no se encuentra
}
```

Por último, veremos el concepto de **abstracción** es un proceso que consiste en crear un tipo de datos, generalmente una clase, que oculta los detalles de la representación de los datos para facilitar su manipulación. En nuestro código, este concepto se refleja claramente en la clase `Producto`, que es una clase abstracta que define los atributos y métodos comunes a todos los productos, pero no se puede instanciar directamente. Esto permite establecer una base general para otros tipos de productos sin necesidad de proporcionar una implementación completa. Además, la clase contiene métodos abstractos como `getDetalles()` y `getTipoProducto()`, que deben ser implementados por las subclases.

Después de construir nuestro sistema en Java, al ejecutar el código se mostrará un menú que permitirá a los usuarios elegir acciones como agregar, modificar o eliminar. Para iniciar, deberán abrir la carpeta "SeminarioJava" y ejecutar el código, donde podrán seleccionar la acción deseada.

```

*****
=== Gestión de Inventario ===
*****
***** Agregaciones *****
1. Agregar Orden de Compra
2. Agregar Devolución
3. Agregar Venta
4. Agregar Producto
5. Agregar Categoría
6. Agregar Cliente
7. Agregar Proveedor
8. Agregar Usuario
9. Agregar Rol
***** Consultas *****
10. Consultar Órdenes de Compra
11. Consultar Devoluciones
12. Consultar Ventas
13. Consultar Clientes
14. Consultar Proveedores
15. Consultar Stock
16. Consultar Usuarios
***** Modificaciones *****
17. Modificar Productos
18. Eliminar Productos
19. Modificar Órdenes de Compra
20. Eliminar Órdenes de Compra
21. Modificar Devoluciones
22. Eliminar Devoluciones
23. Modificar Usuarios
24. Eliminar Usuarios
25. Modificar Proveedores
26. Eliminar Proveedores
0. Salir
Seleccione una opción:

```

Hemos desarrollado todos los métodos necesarios para abordar la problemática del inventario de stock de Monarca Market. Implementamos una sintaxis correcta, especificando cada tipo de dato y estructurando adecuadamente el control del flujo. Además, ante entradas erróneas, implementamos manejos de excepciones para evitar errores o fallas en el sistema.

Aprendimos a emplear estructuras condicionales y repetitivas, lo que nos permite ejecutar o no un bloque de código en función de si se cumple o no una condición, así como repetir un bloque hasta que se cumpla cierta condición. Por ejemplo, al presentar un menú de opciones, los usuarios pueden elegir entre agregar, modificar o eliminar productos. Si ingresan una opción inválida, se les solicita que intenten nuevamente hasta que seleccionen una opción válida.

También declaramos y creamos los objetos necesarios para nuestro sistema, utilizando constructores para inicializar estos objetos de manera adecuada. Por ejemplo, al crear un nuevo producto, se establece su nombre, precio y cantidad en stock desde el inicio.

Finalmente, tuvimos en cuenta los pilares de la programación orientada a objetos, aplicando conceptos como encapsulamiento, herencia, polimorfismo y abstracción, tal como se demostró anteriormente.

A continuación, dejo el enlace del repositorio para que puedan visualizar mi código y probar su funcionamiento:

<https://github.com/CamilaLev07/SeminarioJava/tree/master>