

Desarrollado por:

Armando Ambrosio Nocedal

Omar Arias Dominguez

Documentación:

Gestor de inventarios

Contenido

[Introducción 3](#_Toc146132464)

[Planteamiento del problema 3](#_Toc146132465)

[Objetivo general del proyecto 3](#_Toc146132466)

[Objetivos específicos del proyecto 3](#_Toc146132467)

[Hipótesis 4](#_Toc146132468)

[Requisitos funcionales 4](#_Toc146132469)

[Requisitos no funcionales 4](#_Toc146132470)

[Limitaciones 4](#_Toc146132471)

# Introducción

En un entorno empresarial, la gestión eficiente del inventario es fundamental para garantizar operaciones fluidas y satisfacer la demanda de los clientes. Para brindar una solución efectiva, este proyecto se enfoca en desarrollar un administrador de inventario en Java. Diseñado para ayudar a las pequeñas empresas a administrar sus productos de manera eficiente, el sistema permite a los usuarios agregar nuevos productos, verificar los niveles de existencias y eliminar productos que ya no son necesarios.

# Planteamiento del problema

El desarrollo de este Gestor de Inventarios en Java es esencial para abordar desafíos y problemas comunes en la gestión de inventarios. La mayoría de los sistemas tradicionales carecen de la automatización necesaria para manejar de manera eficiente una amplia variedad de productos y la cantidad de existencias que pueden variar con el tiempo.

# Objetivo general del proyecto

El objetivo general de este proyecto es desarrollar un sistema de gestión de inventarios en Java que permita a los usuarios administrar eficientemente los productos, controlar los niveles de existencias, actualizar la información y gestionar de manera efectiva la eliminación de productos obsoletos. El sistema proporcionará una solución automatizada y fácil de usar para mejorar la precisión, la agilidad y la eficiencia en la administración de inventarios.

# Objetivos específicos del proyecto

1. **Diseñar la Estructura del Sistema**: Definir la arquitectura, las clases y las relaciones necesarias para la representación de productos, inventarios y funcionalidades asociadas.
2. **Implementar una Interfaz de Usuario**: Desarrollar una interfaz de usuario intuitiva y amigable que permita a los usuarios acceder a las funciones del sistema de manera efectiva y que no tengan problemas con su uso.
3. **Crear Clases de Productos**: Diseñar y desarrollar las clases que representarán diferentes tipos de productos, incluyendo atributos como nombre, precio y cantidad en stock.
4. **Implementar Funcionalidad de Agregar Productos**: Crear la función de agregar nuevos productos al inventario, permitiendo a los usuarios ingresar la información relevante del producto.
5. **Desarrollar la Función de Verificación de Existencias:** Implementar la función que permitirá a los usuarios verificar los niveles actuales de existencias de los productos en el inventario.
6. **Realizar Actualizaciones de Información**: Diseñar y programar métodos que permitan a los usuarios actualizar detalles como precio, descripción y cantidad de productos.
7. **Crear Función de Eliminación de Productos**: Implementar la capacidad de eliminar productos que ya no estén disponibles o que vayan a ser retirados.
8. **Implementar Búsqueda**: Desarrollar funciones de búsqueda que permitan a los usuarios encontrar productos por nombre u otros atributos relevantes.

# Hipótesis

Se postula que, mediante la implementación de un Gestor de Inventarios en Java, será posible mejorar significativamente la eficiencia en la administración de inventarios. Al automatizar procesos que antes eran manuales como el seguimiento de existencias y la actualización de información de manera escrita y en continuas hojas.

# Requisitos funcionales

1. Registro de nuevos productos: Los usuarios podrán ingresar los detalles de un nuevo producto, incluyendo nombre, descripción, precio y cantidad inicial.
2. Consulta de niveles de existencias: Los usuarios podrán ver la cantidad actual de un producto específico en el inventario.
3. Actualización de información de productos: Se permitirá a los usuarios modificar la información de un producto, como cambiar su precio, descripción, etc.
4. Eliminación de productos: Los usuarios podrán eliminar productos que ya no están disponibles o que se vayan a retirar del inventario.

# Requisitos no funcionales

1. Rendimiento: El sistema debe ser capaz de manejar un gran número de productos y usuarios de manera eficiente.
2. Usabilidad: La interfaz de usuario debe ser intuitiva y fácil de entender incluso para usuarios que no tienen conocimientos técnicos.
3. Escalabilidad: El sistema debe ser diseñado de manera que pueda escalarse fácilmente en caso de un aumento en la cantidad de productos o usuarios.
4. Mantenibilidad: El código debe estar bien estructurado y documentado para facilitar futuras actualizaciones y mejoras.

# Alcances

1. **Registro de Productos:**
   * Los usuarios podrán registrar nuevos productos en el sistema, ingresando información como nombre, descripción, precio y cantidad inicial en stock.
2. **Visualización de Inventarios:**
   * Los usuarios podrán acceder y visualizar los inventarios existentes en el sistema.
3. **Consulta de Niveles de Existencias:**
   * Los usuarios podrán consultar los niveles de existencias de productos específicos en el inventario.
4. **Actualización de Información de Productos:**
   * Los usuarios tendrán la capacidad de actualizar la información de productos existentes, lo que incluye modificar su precio, descripción y cantidad en stock.
5. **Eliminación de Productos:**
   * Se proporcionará a los usuarios la opción de eliminar productos que ya no estén disponibles o que se hayan vuelto obsoletos.
6. **Búsqueda y Filtrado de Productos:**
   * Los usuarios podrán buscar productos por nombre, descripción, categoría u otros atributos relevantes.
7. **Interfaz de Usuario Intuitiva:**
   * El sistema contará con una interfaz de usuario intuitiva y fácil de usar, diseñada para usuarios de diversos niveles de experiencia técnica.
8. **Seguridad de Acceso:**
   * Se implementará un sistema de autenticación para garantizar que solo usuarios autorizados puedan acceder al sistema.
   * Se establecerán niveles de acceso para controlar quién puede realizar ciertas acciones dentro del sistema.
9. **Rendimiento Eficiente:**
   * El sistema se diseñará y optimizará para manejar eficientemente un gran número de productos, evitando ralentizaciones significativas en las operaciones clave.
10. **Escalabilidad:**
    * La arquitectura del sistema se diseñará teniendo en cuenta la posibilidad de escalabilidad, de modo que pueda adaptarse a un aumento en la cantidad de productos o usuarios sin requerir cambios drásticos en la infraestructura.
11. **Mantenibilidad:**
    * El código fuente se documentará adecuadamente y se seguirán buenas prácticas de programación para facilitar futuras actualizaciones y mejoras en el sistema.
12. **Integración Futura:**
    * Se dejará espacio para la posible integración con otros sistemas o aplicaciones empresariales existentes en el futuro.
13. **Soporte Técnico:**
    * Se proporcionará documentación de usuario y manuales para ayudar a los usuarios a comprender y utilizar el sistema de manera efectiva.
    * Se establecerá un mecanismo de soporte técnico para atender consultas y problemas relacionados con el sistema.
14. **Pruebas y Validación:**
    * Se llevarán a cabo pruebas exhaustivas, incluyendo pruebas de unidad, pruebas de integración y pruebas de usuario, para garantizar la calidad y el funcionamiento adecuado del sistema.

# Limitaciones

* **Escalabilidad**: Dependiendo de la arquitectura y el diseño del sistema, podría haber limitaciones en cuanto a la capacidad de manejar grandes volúmenes de productos y usuarios simultáneos de manera eficiente.
* **Rendimiento**: A medida que la cantidad de productos y usuarios vaya creciendo, el rendimiento del sistema puede verse afectado. Las funciones de búsqueda, actualización y eliminación podrían volverse más lentas si no se optimizan adecuadamente.
* **Seguridad**: La seguridad de los datos es una preocupación importante dentro del sistema. Si no se implementan adecuadamente medidas de autenticación y autorización, podría haber riesgo de acceso no autorizado a la información del inventario y perder información del inventario.
* **Capacidades de Personalización:** Dependiendo de la complejidad del sistema y del tiempo disponible, podría haber limitaciones en la capacidad de personalizar la interfaz de usuario y las funciones según las necesidades de cada negocio.
* **Usabilidad:** Si la interfaz de usuario no se diseña de manera intuitiva, los usuarios podrían tener dificultades para aprender y utilizar el sistema de manera eficiente por lo que el sistema no podría ser aprovechado.
* **Actualización y Mantenimiento:** Si no se planifica un proceso de evolución y mantenimiento adecuado el sistema puede volverse obsoleto con el tiempo y enfrentar problemas de compatibilidad con nuevas tecnologías.