Caratula

Contenido

[**Introducción** 3](#_Toc144403777)

[**Justificación** 3](#_Toc144403778)

[**Objetivo general del proyecto** 3](#_Toc144403779)

[**Objetivos específicos del proyecto** 3](#_Toc144403780)

[**Hipótesis** 4](#_Toc144403781)

[**Requisitos funcionales** 4](#_Toc144403782)

[**Requisitos no funcionales** 4](#_Toc144403783)

[**Limitaciones** 4](#_Toc144403784)

# **Introducción**

En un entorno comercial, la administración eficiente de inventarios es esencial para garantizar un funcionamiento fluido y satisfacer las demandas de los clientes. Con el objetivo de proporcionar una solución moderna y eficaz, este proyecto se centra en el desarrollo de un Gestor de Inventarios en Java. Este sistema estará diseñado para ayudar a las empresas a gestionar sus productos de manera efectiva, permitiendo a los usuarios agregar nuevos productos, verificar niveles de existencias y eliminar productos obsoletos.

# **Justificación**

El desarrollo de este Gestor de Inventarios en Java es esencial para abordar desafíos 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 sus fluctuaciones en existencias.

# **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 la 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.
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 capacidad de agregar nuevos productos al inventario, permitiendo a los usuarios ingresar información relevante.
5. **Desarrollar la Función de Verificación de Existencias:** Implementar la función que permitirá a los usuarios verificar los niveles actuales de existencias de 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 hayan sido descontinuados.
8. **Implementar Búsqueda y Filtrado**: Desarrollar funciones de búsqueda que permitan a los usuarios encontrar productos por nombre, categoría u otros atributos relevantes. Agregar opciones de filtrado por precio, cantidad, etc.

# **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.

# **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 han vuelto obsoletos.

# **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 no 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.

# **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 aumenta, el rendimiento del sistema podría verse afectado. Las operaciones de búsqueda, actualización y eliminación podrían volverse más lentas si no se optimizan adecuadamente.
* **Seguridad**: La seguridad de los datos es una preocupación importante. Si no se implementan adecuadamente medidas de autenticación y autorización, podría haber riesgo de acceso no autorizado a la información del inventario.
* **Capacidades de Personalización:** Dependiendo de la complejidad del sistema, podría haber limitaciones en la capacidad de personalizar la interfaz de usuario y las funciones según las necesidades específicas 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.
* **Integración con Sistemas Existentes:** Si el sistema debe integrarse con otras aplicaciones o sistemas existentes, podría haber desafíos en términos de interoperabilidad y compatibilidad.
* **Actualización y Mantenimiento:** Si no se planifica un proceso de actualización y mantenimiento adecuado, el sistema podría volverse obsoleto con el tiempo y enfrentar problemas de compatibilidad con nuevas tecnologías.