

FORMATO DE REQUERIMIENTOS DEL SISTEMA

Sistema de inventario de productos ejercicio #2

Ingeniería de Software

Profesor: William Alexander Matallana Porras

Estudiante: Tania Lorena Beltrán Lara

Facultad ingeniería de sistemas y computación 2022

Universidad de Cundinamarca extensión Chía

2026

FORMATO DE REQUERIMIENTOS DEL SISTEMA

Ingeniería de Software I

1. Información general del proyecto

Nombre del proyecto	Sistema de inventario de productos
Integrantes	Tania Lorena Beltrán Lara
Programa académico	Ingeniería de Sistemas y Computación
Fecha	17/02/2026
Lenguaje de programación	Java
Tipo de aplicación	Consola

2. Descripción general del sistema

El sistema permite gestionar el inventario de productos de un negocio mediante una aplicación de consola desarrollada en Java.

El usuario puede registrar nuevos productos, listar los productos existentes, buscar productos por identificador, modificar el precio o la cantidad disponible y eliminar productos del inventario.

Cada producto contiene información básica como identificador único, nombre, precio y cantidad disponible. El identificador se genera automáticamente para facilitar el control dentro del sistema.

El proyecto fue desarrollado aplicando Programación Orientada a Objetos y arquitectura por capas, separando la lógica del sistema en modelo, repositorio, servicio y vista para mejorar la organización y mantenimiento del código.

3. Requerimientos Funcionales (RF)

ID	Nombre	Descripción	Entrada(s)	Proceso	Salida
RF-01	Registrar producto	Permite registrar un nuevo producto en el inventario	nombre, precio, cantidad	Crea un objeto producto y lo almacena	Producto registrado
RF-02	Listar productos	Muestra todos los productos almacenados		Recorre la lista de productos	Lista de productos
RF-03	Buscar producto	Permite buscar un producto por	id	Busca en la lista por ID	Producto encontrado o mensaje

		su identificador			
RF-04	Modificar producto	Permite actualizar precio y cantidad	id, precio, cantidad	Actualiza datos del producto	Producto actualizado
RF-05	Eliminar producto	Elimina un producto del inventario	id	Remueve producto de la lista	Producto eliminado

4. Requerimientos No Funcionales (RNF)

ID	Tipo	Descripción
RNF-01	Validación	El precio no puede ser negativo
RNF-02	Validación	La cantidad no puede ser negativa
RNF-03	Estructura	El sistema debe organizarse por capas (modelo, repositorio, servicio y vista)
RNF-04	Calidad	El sistema debe aplicar encapsulamiento y uso de interfaces
RNF-05	Usabilidad	El sistema debe funcionar mediante menú en consola

5. Relación Requerimiento – POO

ID Requerimiento	Clase	Método	Tipo
RF-01	ProductoServicioImpl	registrarProducto()	Funcional
RF-02	ProductoServicioImpl	listarProductos()	Funcional
RF-03	ProductoRepositorioImpl	buscarPorId()	Funcional
RF-04	ProductoServicioImpl	modificarProducto()	Funcional
RF-05	ProductoRepositorioImpl	eliminar ()	Funcional
RNF-01	Producto	setPrecio()	No Funcional
RNF-02	Producto	setCantidad()	No Funcional
RNF-03	Main	inicialización de capas	No Funcional

ESTUDIO DE CASO

Sistema de Inventario de Productos

1. Levantamiento del estudio

Durante el análisis inicial se identificó la necesidad de un pequeño negocio de llevar el control de sus productos de manera organizada.

Actualmente el registro del inventario se realizaba manualmente, lo que generaba problemas como pérdida de información, dificultad para actualizar cantidades y errores al momento de consultar precios o existencias.

A partir de esta situación se definieron los requerimientos principales del sistema:

- Registrar nuevos productos.
- Listar los productos disponibles.
- Buscar productos por identificador.
- Modificar el precio o la cantidad.
- Eliminar productos del inventario.
- Evitar valores negativos en precio o cantidad.
- Generar identificadores automáticos.

Se decidió desarrollar una aplicación de consola en Java aplicando Programación Orientada a Objetos y arquitectura por capas para mantener una estructura organizada.

2. Análisis

En esta fase se identificaron los elementos principales del sistema y las responsabilidades de cada componente.

Actores

- Usuario del sistema (administrador del inventario).

Entidades principales

- Producto: representa los artículos almacenados en el inventario.

Funcionalidades analizadas

- Registro de productos.

- Consulta de información.
- Actualización de datos.
- Eliminación de registros.

Reglas del negocio

- El precio no puede ser negativo.
- La cantidad no puede ser negativa.
- El identificador se genera automáticamente.

También se decidió separar el sistema por capas para facilitar el mantenimiento:

- Modelo → representación de datos.
- Repositorio → manejo de almacenamiento.
- Servicio → lógica del negocio.
- Vista → interacción con el usuario.

3. Diseño

El diseño del sistema se realizó siguiendo arquitectura por capas para garantizar una correcta separación de responsabilidades.

Estructura general

- **Modelo (Producto):** contiene atributos, constructor y validaciones.
- **Repositorio:** define y ejecuta operaciones de almacenamiento.
- **Servicio:** controla las operaciones del inventario.
- **Vista:** implementa el menú en consola.
- **Main:** inicializa el sistema y conecta las capas.

Flujo del sistema

Usuario → Vista → Servicio → Repositorio → Modelo

Principios aplicados

- Encapsulamiento (atributos privados).

- Abstracción mediante interfaces.
- Separación de responsabilidades.
- Reutilización de métodos.

Se decidió usar una lista dinámica (ArrayList) para almacenar productos en memoria simulando una base de datos.

4. Codificación

La etapa de codificación se desarrolló en el lenguaje Java aplicando los conceptos de Programación Orientada a Objetos.

Durante el desarrollo se implementaron:

- Clases organizadas por paquetes según su función.
- Uso de interfaces para desacoplar la lógica del sistema.
- Validaciones dentro del modelo para evitar datos incorrectos.
- Generación automática del identificador de productos.
- Menú interactivo mediante consola para facilitar el uso.

El sistema permite ejecutar todas las operaciones requeridas del inventario de forma sencilla, manteniendo una estructura clara y escalable para futuras mejoras.