Descripción General:

Este proyecto es un Sistema de Gestión de Inventario implementado en Java. Utiliza una arquitectura de tres capas (presentación, lógica de negocio y acceso a datos) y sigue el patrón de diseño DAO (Data Access Object). El sistema permite a los usuarios realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en productos de un inventario a través de una interfaz de línea de comandos.

Arquitectura:

1. Capa de Presentación: InventoryUI

2. Capa de Lógica de Negocio: InventoryService

3. Capa de Acceso a Datos: ProductDAO

Patrones de Diseño:

1. DAO (Data Access Object): Utilizado en ProductDAO para abstraer y encapsular el acceso a la base de datos.

2. Singleton: Implícitamente usado en InventoryService (aunque no está implementado estrictamente).

Tecnologías:

1. Java

2. JDBC para la conexión a la base de datos

3. MySQL como sistema de gestión de base de datos

4. Docker para la contenerización de la base de datos

Requerimientos Funcionales:

1. Añadir nuevos productos al inventario

2. Actualizar información de productos existentes

3. Eliminar productos del inventario

4. Ver detalles de un producto específico

5. Listar todos los productos en el inventario

Requerimientos No Funcionales:

1. Persistencia de datos utilizando una base de datos MySQL

2. Interfaz de usuario basada en línea de comandos

3. Manejo de errores y excepciones

4. Contenedorización de la base de datos con Docker

Métodos Principales:

1. addProduct(): Añade un nuevo producto

2. updateProduct(): Actualiza un producto existente

3. deleteProduct(): Elimina un producto

4. getProduct(): Obtiene detalles de un producto específico

5. getAllProducts(): Lista todos los productos

Estructura en árbol de carpetas y archivos:

```

project\_root/

│

├── src/

│ └── main/

│ └── java/

│ └── com/

│ └── inventory/

│ ├── model/

│ │ └── Product.java

│ │

│ ├── dao/

│ │ └── ProductDAO.java

│ │

│ ├── service/

│ │ └── InventoryService.java

│ │

│ ├── ui/

│ │ └── InventoryUI.java

│ │

│ └── Main.java

│

├── pom.xml (suponiendo que se usa Maven)

│

└── docker-compose.yml

```

Esta estructura refleja la organización del código en paquetes según su función en la arquitectura de tres capas. El archivo `docker-compose.yml` en la raíz del proyecto se utiliza para configurar y ejecutar el contenedor de MySQL.