Trabajo Integrador Final

Programación II - Tecnicatura Universitaria en Programación a Distancia

Integrantes:

- Rousseaux Carolina
- Santagada Leandro
- Rosales Gastón

Introducción

Este Trabajo Final Integrador tuvo como objetivo la creación de un CRUD con patron de diseño DAO, teniendo como relaciones a dos tablas, Producto y CodigoBarras.
La estructura de carpetas elegida se baso en el modelo propuesto por la catedra. En el esquema se implementaron servicios, modelos, conexiones con la base de datos y un menu para manejar la interfaz del listado, creación o eliminación de los productos.
Para la relación elegida - Producto → CodigoBarras, la regla de negocio establece que 1 producto puede tener 1 codigo de barras asociado, asi como el codigo de barras, esta directamente asociado a 1 producto, esta relación es de tipo.
Para la restricción de tipos de dominio en los datos, se especifica que:
id: entero;

Producto

eliminado: boolean (el tipo TINYINT con 1 carácter reemplaza al booleano)
nombre: string/varchar con un máximo de 120 caracteres
marca: string/varchar con un máximo de 80 caracteres
categoria: string/varchar con un máximo de 80 caracteres
precio: double; para aceptar numeros enteros o decimales con punto

CodigoBarras

id: entero/int
eliminado: boolean (el tipo TINYINT con 1 carácter reemplaza al booleano)
tipo: string/varchar con un máximo de 45 caracteres
fechaAsignacion: es un tipo de dato DATETIME donde su formato predeterminado es yyyy-MM-dd HH:mm:ss.fff
observaciones: string/varchar con un máximo de 255 caracteres

1. Estructura y Arquitectura por Capas

El proyecto sigue una arquitectura por capas, esta arquitectura sigue el patron de diseño DAO, y la estructura de paquetes y clases utilizada es la siguiente:

Paquete	Responsabilidad
Config	Clases relacionadas con la configuración del sistema, específicamente la conexión a la base de datos (DatabaseConnection.java) y la gestión de transacciones (TransactionManager.java).
Dao	Data Access Object. Es responsable de la persistencia de los datos. Contiene interfaces y clases concretas para realizar las operaciones CRUD (Crear, Leer, Actualizar, Borrar) contra la base de datos para las entidades del dominio (ProductoDAO.java, CodigoBarrasDAO.java, GenericDAO.java).
Main	Contiene las clases de ejecución y la interfaz de usuario (UI). Aquí se encuentra el punto de entrada de la aplicación (Main.java), la lógica del menú (MenuHandlerMain.java), la visualización de datos (Display.java) y las pruebas de conexión (TestConexion.java).
Models	Define los objetos del dominio (entidades). Representa la estructura de los datos del negocio (Producto.java, CodigoBarras.java, Base.java).
Service	Contiene la lógica de negocio o Business Logic Layer. Implementa operaciones que pueden coordinar múltiples DAOs y aplicar reglas de negocio (ProductoServiceImpl.java, CodigoBarrasServiceImpl.java, GenericService.java).

2. Dominio

El dominio central del proyecto gira en torno a la gestión de Productos y sus Códigos de Barras. Es un dominio común para aplicaciones de inventario o sistemas de punto de venta (POS).
La relación fundamental se establece entre el Producto y el Código de Barras. Esta es comúnmente una relación de uno a muchos (1:N), lo que significa que un único Producto puede estar asociado a múltiples Códigos de Barras; esto podría deberse a diferentes formatos de empaque o presentaciones del mismo producto.

- Entidad Principal: Producto
- Entidad Dependiente: CodigoBarras

Esta separación en dos entidades (Producto y CodigoBarras) es adecuada para normalizar la base de datos y manejar eficientemente el caso donde múltiples identificadores (Códigos de Barras) refieren a un único artículo lógico (Producto).

***La normalización de la base de datos, se incluye en archivos .sql adjuntos**

3. Persistencia y Estructura de la Base

Persistencia

La capa de persistencia está implementada en el paquete Dao utilizando el patrón DAO (Data Access Object). Las clases en Dao interactúan directamente con la base de datos (presumiblemente a través de JDBC, dado el contexto de Programación 2) para mapear objetos de Java (Models) a registros de la base y viceversa.

Estructura de la Base de Datos

La estructura mínima de la base de datos debe reflejar las entidades del dominio:

Producto:

eliminado: boolean (el tipo TINYINT con 1 carácter reemplaza al booleano)
nombre: string/varchar con un máximo de 120 caracteres
marca: string/varchar con un máximo de 80 caracteres
categoria: string/varchar con un máximo de 80 caracteres
precio: double; para aceptar numeros enteros o decimales con punto

CodigoBarras:

id: entero/int
eliminado: boolean (el tipo TINYINT con 1 carácter reemplaza al booleano)
tipo: string/varchar con un máximo de 45 caracteres
fechaAsignacion: es un tipo de dato DATETIME donde su formato predeterminado es yyyy-MM-dd HH:mm:ss.fff
observaciones: string/varchar con un máximo de 255 caracteres

4. Validaciones y Reglas de Negocio

Las Validaciones (comprobación de integridad de datos, ej: campos no nulos, formatos correctos) y las Reglas de Negocio (lógica específica del dominio, ej: calcular el precio final con impuestos, no permitir eliminar un producto con stock positivo) deberían residir principalmente en el paquete Service.

Reglsa de Negocio:

- Al actualizar un producto, se debe verificar que el nuevo código de barras, si se proporciona, no esté ya asociado a otro producto (o si se permite tener varios códigos, la validación de unicidad debe ser en la tabla CodigoBarras).
- Los campos que se introducen en las tablas no sean null (!= null)
- Las consultas se harán dentro de sentencias Prepared Statement para protegerse ante posibles inyecciones SQL.
- Los IDs ingresados (tanto para Producto como para Codigo de barras sea >=0)

5. Conclusión y Errores Cometidos Comunes (Hipótesis)

- El diseño del proyecto fue en la practica estructuralmente correcto siguiendo un modelo de arquitectura de tres capas (Presentación/Main, Lógica de Negocio/Service, Acceso a Datos/DAO + Configuración).
- En la fase de implementación aparecieron errores que impactaron la funcionalidad, como los mencionados:
    1. **Fallos de Sintaxis en ProductoDAO (**Errores de tipeo, mayúsculas/minúsculas o falta de puntos y comas en sentencias SQL o código Java.)
    2. **Consultas a la Base de Datos** (uso incorrecto de la sintaxis SQL (ej. WHERE mal escrito, nombres de columnas erróneos) o no liberar los recursos (ResultSet, Statement) después de la consulta)
    3. **Problemas en la conexión con la BD** (La cadena de conexión en DatabaseConnection.java es incorrecta (URL, usuario, contraseña) o falta el driver JDBC en el classpath.)

Software utilizado:

XAMPP 8.2.12
MySQL Worbench
Interfaz gráfica PHPMyAdmin
Gemini AI

Link Video Explicativo: https://www.youtube.com/watch?v=EW0wCN9sxzg
Link Repositorio Github: https://github.com/gst-n/Trabajo_final_integrador_P2