Documentación del proyecto. Acceso a BD relacionales

- Acceso a Datos

Francisco Javier Soler Aguado
2º DAM

Indice

Documentación del Proyecto: Gestor Tienda

1. Introducción y Definición del Problema

Descripción del Problema

El proyecto nace de la necesidad de gestionar de manera eficiente la información de un comercio minorista. Actualmente, la gestión manual de productos, proveedores, clientes y ventas conlleva errores de inventario, pérdida de información de clientes y dificultades para calcular facturaciones históricas .

Objetivo de la Aplicación

El objetivo es desarrollar una aplicación de escritorio en Java que permita realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) sobre una base de datos relacional . La aplicación resuelve la impedancia objeto-relacional mediante el uso de JDBC y el patrón de diseño DAO, permitiendo:

- Gestión de inventario (control de stock).
- Registro de proveedores y clientes.
- Procesamiento de ventas con cálculo automático de totales e impuestos.
- Persistencia de datos compatible tanto con MySQL como con H2 (Embebida) .

---

2. Estructura de la Base de Datos

Definición de la Estructura

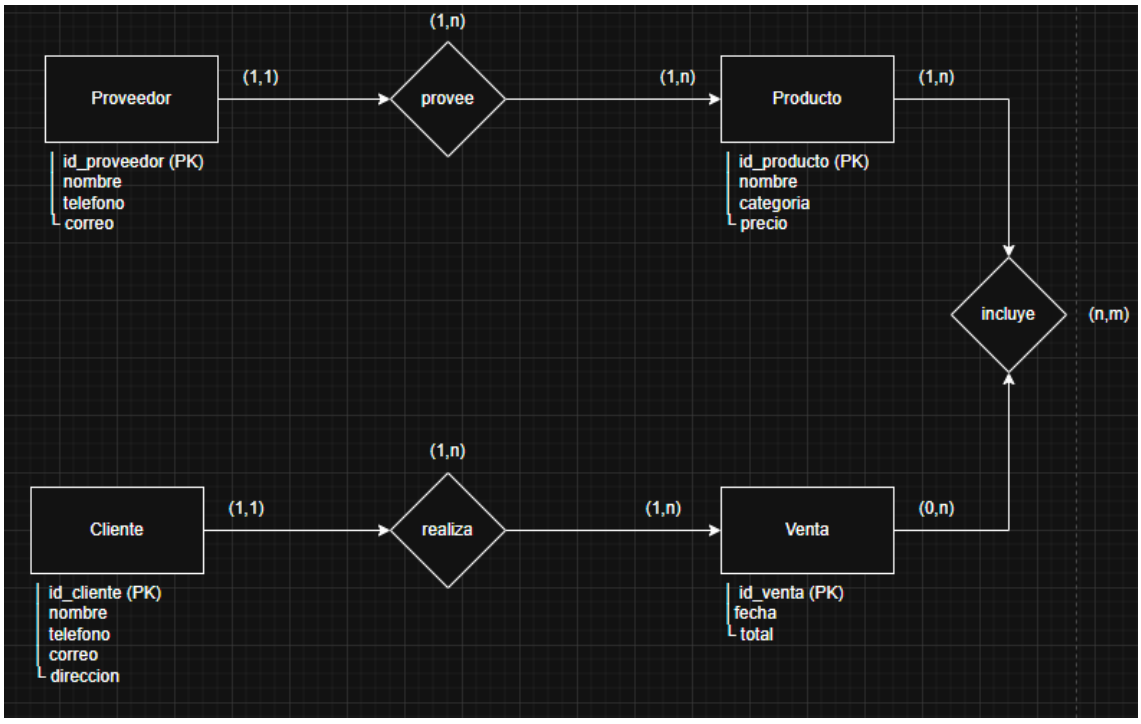
La base de datos, denominada tienda, consta de 5 tablas principales diseñadas en tercera forma normal (3NF) para garantizar la integridad de los datos .

1. **Proveedor:** Almacena la información de quienes suministran los productos.
2. **Cliente:** Registra los datos de contacto de los compradores.

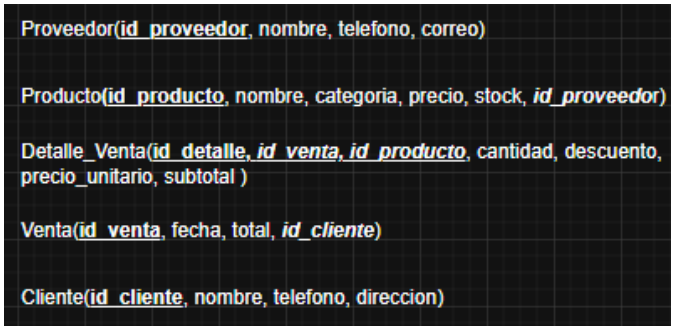
3. **Producto:** Contiene el inventario, categorización, precios y la relación con el proveedor (Clave Foránea: id_proveedor) .
4. **Venta:** Cabecera de la factura que vincula una fecha y un total con un cliente (Clave Foránea: id_cliente).
5. **Detalle_Venta:** Tabla intermedia que resuelve la relación N:M entre Venta y Producto. Almacena la cantidad, precio unitario histórico y subtotales .

Modelo de Base de Datos (Diagramas)

Diagrama Entidad-Relación (ERD):



Modelo Relacional:



3. Diseño de la Aplicación: Arquitectura MVC

Explicación del uso de MVC

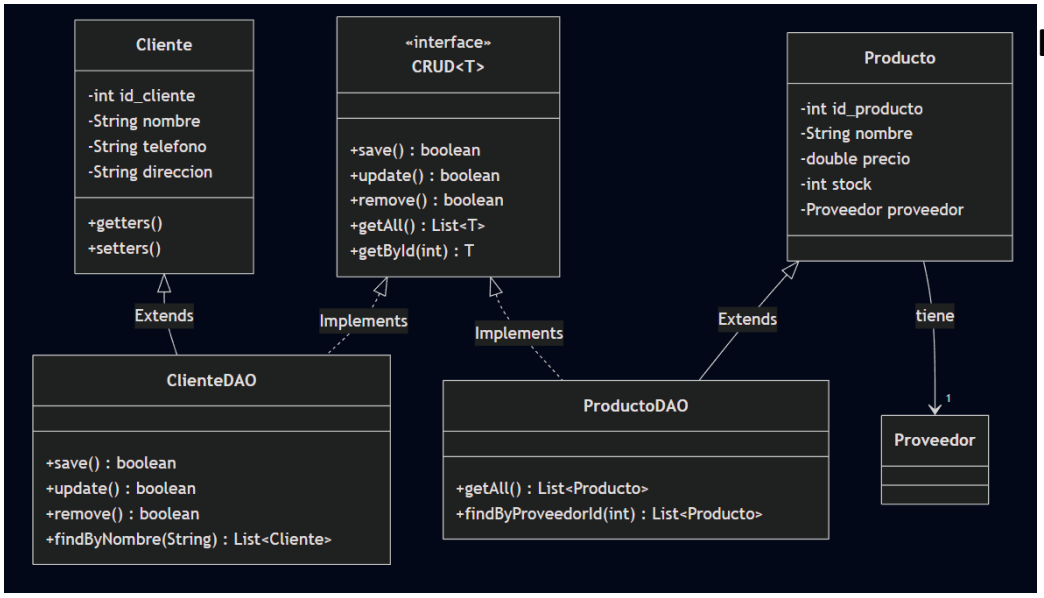
La aplicación implementa estrictamente el patrón Modelo-Vista-Controlador para desacoplar la lógica de negocio de la interfaz de usuario :

- **Modelo (org.fran.gestortienda.model y DAO):**
    - Contiene las entidades que representan las tablas de la base de datos.
    - Incluye los DAOs que implementan una interfaz genérica CRUD<T> . Estos se encargan de todas las sentencias SQL e interactúan con ConnectionFactory .
- **Vista (resources/org/fran/gestortienda/ui):**
    - Definida mediante archivos FXML y hojas de estilo CSS para una apariencia moderna y elegante.
- **Controlador (org.fran.gestortienda.controller):**
    - Clases Java (ej. VentasController.java, MainController.java) que gestionan los eventos de la interfaz, llaman a los métodos del DAO y actualizan la vista con los datos recibidos.

Diagrama de Clases

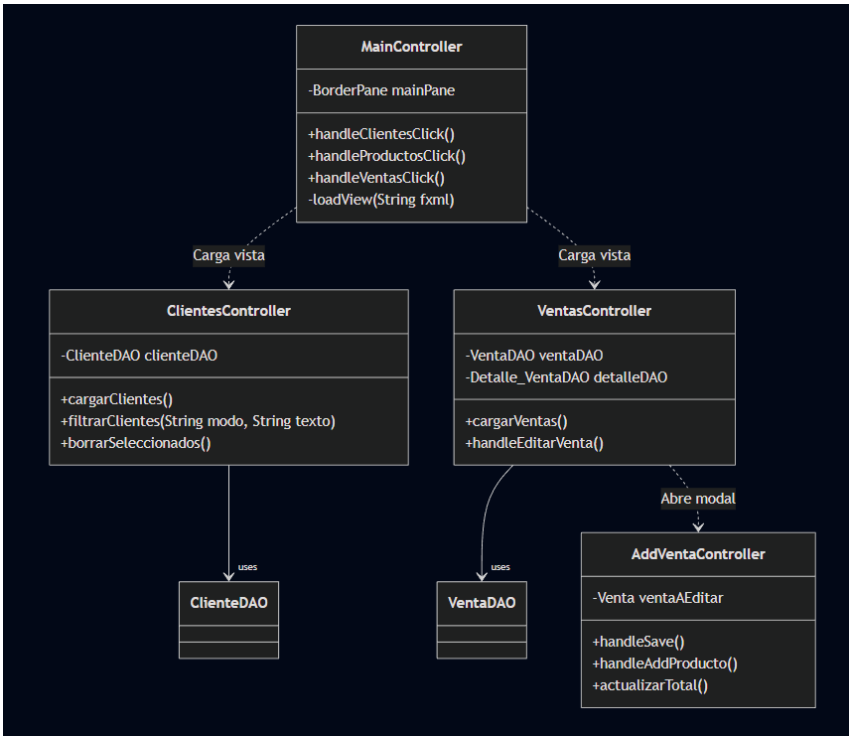
A. Paquete Model y DAO

Este diagrama muestra cómo las clases DAO heredan de las entidades y realizan la implementación de la interfaz CRUD.



B. Paquete Controller

Este diagrama muestra cómo el MainController orquesta la navegación y cómo los controladores específicos usan los DAOs.



4. Consultas Avanzadas y Gestión de Objetos

Se han implementado consultas avanzadas utilizando PreparedStatement para seguridad y eficiencia .

1. Consultas con JOIN

Para solucionar la separación de tablas, se utilizan JOINs en los DAOs. Un ejemplo lo tenemos en ProductoDAO:

- **Objetivo:** Al listar productos, necesitamos saber el nombre del proveedor, no solo su ID.
- **Implementación:**

  SQL

  SELECT p.*, pr.nombre AS proveedor_nombre, pr.telefono, pr.correo

  FROM producto p

  JOIN proveedor pr ON p.id_proveedor = pr.id_proveedor

2. Consultas Selectivas y Filtros

La aplicación permite filtrar dinámicamente en todas las vistas.

- **Ejemplo en ClienteDAO:** Búsqueda por coincidencia parcial (LIKE).

  SQL

  SELECT * FROM cliente WHERE nombre LIKE ?

  El controlador inserta el texto con los comodines % permitiendo búsquedas a gusto .

5. Configuración y Conexión

La conexión utiliza el patrón Singleton en la clase ConnectionFactory. La configuración es externa a través del archivo config.properties, permitiendo cambiar entre MySQL y H2 cambiando una sola línea (db.active=h2 o mysql) . Además, DatabaseManager incluye lógica para crear tablas automáticamente si se usa la base de datos embebida H2.

---

6. Enlace al Código Fuente

- **Repositorio GitHub:** https://github.com/FranciscoSolerAguado/Gestor-Tienda.git

---

7. Conclusiones y Aprendizajes

Reflexión sobre el desarrollo

El desarrollo ha permitido consolidar el ciclo completo de desarrollo de software con acceso a datos y los requisitos pedidos pero me hubiera gustado que el proyecto hubiera sido mas completo con algunas de la funcionalidades que se mencionan abajo como el login y la generación de facturas pero por tiempo no se ha podido hacer.

Posibles Mejoras

Para futuras versiones se podría implementar:

1. Implementar un sistema de Login y roles de usuario.
2. Generación de facturas en PDF usando librerías externas.
3. Migrar a Hibernate/JPA para automatizar el mapeo objeto-relacional que ahora se hace manualmente en los DAOs .