INSTITUTO DE EDUCACIÓN SECUNDARIA ISIDRA DE GUZMÁN

PRÁCTICA DE EVALUACIÓN CONTINUA (PEC)
Acceso a Datos (JDBC)

Alumno/a: Diego Luengo Gil
Grupo: 2º DAM
Módulo: Acceso a Datos
Profesor/a: Ruth Lospitao Ruiz
Fecha: 21/11/2025
Curso académico: 2025/2026

Índice

Índice 2
1. Descripción del proyecto y objetivos 3
2. Requisitos y entorno 4
3. Estructura del proyecto (paquetes y clases) 5
4. Diseño lógico de la base de datos 8
5. Casos de uso 10
6. Implementación 10
7. Validación y robustez 14
8. Pruebas y evidencias 16
9. Dificultades encontradas y soluciones 17
10. Conclusiones y líneas de mejora 17
11. Bibliografía y recursos 17
12. Anexos 17

1. Descripción del proyecto y objetivos

• El proyecto consiste en un software con opciones para que una tienda gestione sus productos, sus clientes y las ventas que se hacen a los clientes

• El sistema permite a los usuarios agregar eliminar actualizar y obtener productos y clientes, y tambien permite crear ventas actualizando las cantidad correspondientes en productos y clientes

• Objetivos técnicos de la PEC: CRUD, DDL/DML, transacciones + savepoint, procedimientos/funciones, ResultSet/ResultSetMetaData, menú por consola.

2. Requisitos y entorno

• Java 17+, Maven, JDBC, MySQL (versión), controlador/URL/puerto/bd.

• Dependencias del pom.xml (añadir fragmento si procede).

```
<!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <version>9.5.0</version>
    </dependency>
```

• Configuración/localización de credenciales src\main\resources\config.properties.

3. Estructura del proyecto (paquetes y clases)

Estructura del Proyecto

A continuación, se describe la estructura de paquetes y clases del proyecto Inventario, detallando el rol de cada componente.

Árbol de Directorios

```
src/main/java/com/inventario
├── Main.java
├── bbdd
│   ├── ConexionBBDD.java
│   └── GestionBBDD.java
├── clientes
│   ├── Cliente.java
│   ├── ClientesBBDD.java
│   └── GestionDeClientes.java
├── excepciones
│   └── DatoInvalidoException.java
├── productos
│   ├── GestionDeProductos.java
│   ├── Producto.java
│   └── ProductosBBDD.java
├── util
│   └── Util.java
└── ventas
    ├── DetalleVenta.java
    ├── GestionVentas.java
    ├── Venta.java
    └── VentasBBDD.java
```

Descripción de Paquetes y Clases

Paquete Principal (com.inventario)

Contiene el punto de entrada de la aplicación.

- Main.java: Clase principal que inicia el programa. Muestra el menú principal y dirige el flujo hacia los diferentes módulos (productos, clientes, ventas).

Paquete bbdd

Encargado de la conectividad y gestión genérica de la base de datos.

- ConexionBBDD.java: Gestiona la conexión JDBC con la base de datos (MySQL). Provee métodos para obtener y cerrar la conexión.
- GestionBBDD.java: Contiene utilidades genéricas para la base de datos, como métodos para imprimir tablas de forma dinámica a partir de consultas SQL.

Paquete clientes

Módulo para la gestión de la entidad Cliente.

- Cliente.java: Clase POJO (Plain Old Java Object) que representa el modelo de datos de un cliente (id, nombre, email, teléfono, dinero). Incluye validaciones en el constructor.
- ClientesBBDD.java: Clase DAO (Data Access Object) que maneja las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la tabla cliente de la base de datos.
- GestionDeClientes.java: Maneja la lógica de interacción con el usuario para el módulo de clientes (menús, lectura de datos por consola).

Paquete productos

Módulo para la gestión de la entidad Producto.

- Producto.java: Clase POJO que representa el modelo de datos de un producto (id, nombre, descripción, precio, stock).
- ProductosBBDD.java: Clase DAO que maneja las operaciones CRUD en la tabla producto de la base de datos.
- GestionDeProductos.java: Maneja la lógica de interacción con el usuario para el módulo de productos.

Paquete ventas

Módulo para la gestión de ventas y transacciones.

- Venta.java: Clase POJO que representa la cabecera de una venta (id, idCliente).
- DetalleVenta.java: Clase POJO que representa una línea de detalle de la venta (producto, cantidad, precio unitario).
- VentasBBDD.java: Clase DAO para gestionar las ventas en la base de datos. Maneja transacciones complejas, llamadas a procedimientos almacenados (RegistrarDetalleVenta, CalcularTotalVenta) y consultas con JOINs.
- GestionVentas.java: Maneja el flujo de venta en la consola, incluyendo la selección de clientes, adición de productos al carrito, y confirmación de la compra con control transaccional (commit/rollback).

Paquete util

Utilidades transversales a toda la aplicación.

- Util.java: Clase estática con métodos auxiliares para leer entradas del usuario de forma robusta (validación de números, textos, rangos) y formateo de salidas.

Paquete excepciones

Manejo de errores personalizados.

- DatoInvalidoException.java: Excepción personalizada (Checked Exception) utilizada para indicar errores de validación en los datos de negocio (ej. email inválido, precio negativo).

4. Diseño lógico de la base de datos

Modelo Relacional

La base de datos tienda está compuesta por cuatro tablas principales que gestionan el inventario, los clientes y el proceso de ventas.

Tablas

1.

Producto

Almacena la información del inventario disponible.

- **Clave Primaria (PK):** id_producto (INT, Auto-incrementable)
- **Atributos:**
    - nombre (VARCHAR): Nombre del producto.
    - descripción (VARCHAR): Descripción detallada.
    - precio (DECIMAL): Precio actual del producto.
    - stock (INT): Cantidad disponible (Restricción: stock >= 0).

2.

Cliente

Almacena la información de los usuarios que realizan compras.

- **Clave Primaria (PK):** id_cliente (INT, Auto-incrementable)
- **Atributos:**
    - nombre (VARCHAR): Nombre completo.
    - email (VARCHAR): Correo electrónico.
    - teléfono (VARCHAR): Número de contacto.
    - dinero (INT): Saldo disponible en la cuenta del cliente (Restricción: dinero >= 0).

3.

Venta

Representa la cabecera de una transacción de venta.

- **Clave Primaria (PK):** id_venta (INT, Auto-incrementable)

- **Clave Foránea (FK):** id_cliente referencia a

Cliente(id_cliente).

- **Cardinalidad:** Un cliente puede tener muchas ventas (1:N). Una venta pertenece a un único cliente.

4. DetalleVenta

Representa las líneas de productos dentro de una venta.

- **Clave Primaria (PK):** id_detalle (INT, Auto-incrementable)
- **Claves Foráneas (FK):**
    - id_venta referencia a

Venta(id_venta).

    - id_producto referencia a

Producto(id_producto).

- **Atributos:**
    - cantidad (INT): Unidades vendidas.
    - precio_unitario (DECIMAL): Precio del producto en el momento de la venta.
- **Cardinalidad:**
    - Una venta tiene muchos detalles (1:N).
    - Un producto puede aparecer en muchos detalles de venta (1:N).

5. Casos de uso

ID	Nombre	Descripción	Pre-condiciones	Salida / Resultado
CU-01	**Registrar Cliente**	Da de alta a un nuevo cliente en el sistema.	Base de datos accesible.	Cliente guardado con ID único. Mensaje de éxito.
CU-02	**Listar Clientes**	Muestra la lista de todos los clientes registrados.	Existencia de clientes en BBDD.	Tabla con datos de clientes (nombre, email, saldo).
CU-03	**Actualizar Cliente**	Modifica datos personales (email, teléfono) de un cliente.	El cliente debe existir.	Datos actualizados en BBDD.
CU-04	**Ingresar Saldo**	Añade dinero a la cuenta virtual de un cliente.	El cliente debe existir. Monto > 0.	Saldo incrementado. Mensaje con nuevo saldo.
CU-05	**Eliminar Cliente**	Borra un cliente de la base de datos.	El cliente debe existir.	Cliente eliminado. Mensaje de éxito.
CU-06	**Iniciar Venta**	Comienza una nueva transacción de venta para un cliente seleccionado.	Cliente registrado y seleccionado.	Sesión de venta activa.
CU-07	**Añadir al Carrito**	Agrega una cantidad de un producto a la venta actual.	Venta iniciada. Producto con stock suficiente.	Detalle añadido temporalmente. Stock reservado o validado.
CU-08	**Calcular Total**	Calcula el monto total a pagar sumando los subtotales del carrito.	Venta con al menos un producto.	Muestra el total en euros (€).
CU-09	**Finalizar Venta**	Confirma la compra, descuenta el saldo del cliente y guarda los cambios.	Cliente con saldo suficiente. Stock disponible.	Venta guardada en BBDD. Saldo descontado. Ticket en consola.
CU-10	**Cancelar Venta**	Aborta la venta en curso y deshace los cambios (Rollback).	Venta iniciada pero no finalizada.	Transacción revertida. Ningún cambio en BBDD.

6. Implementación

Detalles de Implementación

A continuación se describen los aspectos técnicos clave de la implementación del sistema.

6.1 CRUD desde Código

Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se gestionan a través de clases BBDD específicas para cada entidad.

- **Clases Clave:**

ClientesBBDD,

ProductosBBDD.

- **Métodos Clave:**
    - **Create:**

insertarCliente(Cliente c),

insertarProducto(Producto p). Usan

PreparedStatement con INSERT para poder colocar los valores en la sentencia

    - **Read:**

obtenerClientes(),

buscarPorCampo(String campo, Object valor).

Ejecutan SELECT y mapean el ResultSet a objetos.

    - **Update:**

actualizarClienteCampo(...),

agregarDinero(...).

Usan UPDATE al que se le pasan los campos por parametros.

    - **Delete:**

eliminarCliente(int id),

eliminarProducto(int id).

Ejecutan DELETE por ID.

Justificación: Este patron centraliza el acceso a datos, facilitando la reutilización de código.

6.2 DML Ejecutadas

La aplicación ejecuta principalmente sentencias **DML (Data Manipulation Language)** para la gestión de datos.

- **DML:** INSERT, UPDATE, DELETE, SELECT. Se ejecutan constantemente durante el uso del programa (altas, ventas, consultas).

6.3 Procesamiento con ResultSet y ResultSetMetaData

Para consultar los datos de las tablas y imprimirlos por pantalla, se utiliza ResultSetMetaData.

- **Clase:**

GestionBBDD.

- **Método:**

imprimirTabla(String sql).

- **Funcionamiento:**
    - Se obtiene ResultSetMetaData del ResultSet (rs.getMetaData()).
    - metaData.getColumnCount(): Determina cuántas columnas tiene el resultado.
    - metaData.getColumnLabel(i): Obtiene el nombre de la columna para imprimir la cabecera dinámicamente.
    - Se itera sobre las filas y columnas imprimiendo rs.getObject(i).

Justificación: Permite imprimir cualquier tabla o resultado de consulta sin conocer sus columnas de antemano, haciendo el método

imprimirTabla totalmente reutilizable para Clientes, Productos o Ventas.

6.4 Transacciones

El sistema implementa control transaccional estricto en el proceso de ventas para asegurar la integridad de los datos (ACID).

- **Clase:**

GestionVentas.

- **Método:**

guardarVenta(Scanner scanner).

- **Mecanismo:**
    - **Inicio:** con.setAutoCommit(false) deshabilita el guardado automático.
    - **Savepoint:** Savepoint spInicioVenta = con.setSavepoint("InicioVenta") marca el inicio de una venta individual dentro de una sesión.
    - **Commit:** Si el usuario confirma y todo es correcto, se hace con.commit() al final de la sesión.
    - **Rollback Parcial:** Si una venta falla o el usuario la rechaza, se ejecuta con.rollback(spInicioVenta) para deshacer solo esa venta sin afectar a las anteriores de la misma sesión.
    - **Rollback Total:** Si el usuario cancela toda la sesión, con.rollback() deshace todo.

Justificación: Esto permite al usuario abandonar la venta sin perder todo el progreso o incluso desaciendolo todo para dependiendo de la situacion

6.5 Procedimientos y Funciones Almacenadas

Se utilizan CallableStatement para invocar lógica compleja alojada en la base de datos.

- **Clase:**

VentasBBDD.

- **Procedimiento RegistrarDetalleVenta:**
    - **Parámetros IN:** ID venta, ID producto, cantidad, precio.
    - **Parámetro OUT:** p_estado (INT). Devuelve 1 (éxito), -1 (stock insuficiente), etc.
    - **Uso:** cs.registerOutParameter(5, Types.INTEGER) captura el resultado de la operación lógica (verificación de stock + inserción + actualización) realizada en el servidor.
- **Función CalcularTotalVenta:**
    - **Parámetro IN:** ID venta.
    - **Retorno:** Total (DECIMAL).
    - **Uso:** cs.registerOutParameter(1, Types.DECIMAL) obtiene el cálculo del total encapsulado en la BBDD.

Justificación: Reduce el tráfico de red y delega la lógica de negocio crítica (como el control de stock) al motor de base de datos.

6.6 Menú de Consola y Navegación

La interfaz de usuario es una aplicación de consola estructurada mediante bucles y switch.

- **Clases:**

Main, Util, y clases Gestion....

- **Estructura:**
    - **Bucle Principal:** do { ... } while (opcion != 0) mantiene el programa en ejecución.
    - **Switch:** Despacha la ejecución a los submenús (GestionDeProductos.menuProductos, etc.).
    - **Utilidades:** Util.pedirNumeroConRango maneja la entrada del usuario, validando que sea un número y esté dentro de las opciones válidas, evitando excepciones por entradas incorrectas (letras, fuera de rango).

Justificación: Provee una navegación intuitiva y robusta, asegurando que el usuario siempre pueda volver atrás o salir ordenadamente sin cierres abruptos por errores de entrada.

7. Validación y robustez

Validación y Robustez

El sistema ha sido diseñado priorizando la robustez para asegurar que errores de entrada o fallos inesperados no detengan la ejecución abruptamente.

7.1 Validación de Entradas del Usuario

La validación se realiza en dos niveles: **Interfaz de Usuario** y **Lógica de Negocio**.

Nivel de Interfaz (Clase

Util)

Se asegura que los datos introducidos por consola sean del tipo y rango correcto antes de procesarlos.

- **Control de Tipos:** Métodos como

pedirNumero y

pedirDecimal utilizan bucles while (!scanner.hasNext___()) para evitar que el programa falle si el usuario introduce letras cuando se espera un número.

- **Control de Rangos:** Métodos como

pedirNumeroConRango(min, max) o

pedirDecimalMinimo(min) obligan al usuario a introducir valores coherentes (ej. opciones de menú entre 0 y 4, precios positivos).

- **Control de Texto:**

pedirTexto evita entradas vacías.

Nivel de Negocio (Constructores y Setters)

Las clases del modelo (

Cliente,

Producto) validan la integridad de los datos antes de crear objetos.

- **Ejemplo (**

Cliente**):** El constructor llama a métodos privados como

validarEmail(email) que verifica el formato con expresiones regulares (regex), o

validarDinero(dinero) para impedir saldos negativos.

7.2 Manejo de Excepciones

El sistema utiliza una estrategia de manejo de excepciones jerárquica para informar al usuario sin interrumpir el flujo.

- **Excepciones Personalizadas:** Se ha creado DatoInvalidoException (Checked Exception) para errores de lógica de negocio (ej. email mal formado). Esto obliga a manejar explícitamente estos errores y crea una capa mas de seguridad a la hora de crear un objeto.
- **Bloques Try-Catch:**
    - En los menús (

GestionVentas, GestionDeClientes), las operaciones se envuelven en bloques try-catch.

    - **SQLException:** Se captura para errores de base de datos (conexión caída, violación de claves foráneas). Se muestra un mensaje amigable (System.err.println) y el programa vuelve al menú.
    - **Exception (Genérica):** Se captura como última red de seguridad para errores imprevistos, evitando el cierre de la JVM ("crash").

Ejemplo de Robustez: Si falla una inserción SQL en

GestionVentas, el catch captura el error, hace rollback de la transacción y permite al usuario intentar de nuevo o salir, en lugar de mostrar un "Stack Trace" y cerrar el programa.

7.3 Uso de Try-with-resources

Para la gestión eficiente de recursos externos (conexiones a BBDD, streams), se utiliza la sintaxis try-with-resources de Java 7+.

Implementación: En todas las clases DAO (

ClientesBBDD,

ProductosBBDD,

GestionBBDD), la apertura de Connection, PreparedStatement y ResultSet se realiza dentro de los paréntesis del try.

Beneficios:

- **Cierre Automático:** Garantiza que close() se llame siempre al finalizar el bloque, evitando fugas de memoria y conexiones abiertas ("memory leaks").

8. Pruebas y evidencias

Las fotos correspondientes a los casos de prueba estan en el documento Anexos_PEC2_Diego_Luengo

ID	Funcionalidad	Entrada	Resultado esperado	Resultado obtenido	OK/KO
CP01CU01	**Registrar Cliente**	DATOS DEL USUARIO	USUARIO CREADO	USUARIO CREADO	OK
CP02CU01	**Registrar Cliente**	DATOS NO VALIDOS	ERROR DE DATOS	SOLICITA NUEVO DATO	OK
CP03CU02	**Listar Cliente**	PRECONDICION CLIENTES CREADOS	TABLA DE LOS CLIENTES	TABLA DE LOS CLIENTES	OK
CP04CU02	**Listar Cliente**	PRECONDICION SIN CLIENTES	TABLA VACIA	TABLA VACIA	OK
CP05CU03	**Actualizar Cliente**	ID CORRECTO	CONFIRMACION	CONFIRMACION	OK
CP06CU03	**Actualizar Cliente**	ID INCORRECTO	ERROR CLIENTE NO ECONTRADO	ERROR CLIENTE NO ACTUALIZADO	OK
CP07CU04	**Ingresar Saldo**	CANTIDAD SUPERIOR A 0	CONFIRMACION	CONFIRMACION	OK
CP08CU04	**Ingresar Saldo**	CANTIDAD INFERIOR A 0	ERROR NO ACTUALIZADO	ERROR NO ACTUALIZADO	OK
CP09CU05	**Eliminar Cliente**	ID CORRECTO	CONFIRMACION	CONFIRMACION	OK
CP10CU05	**Eliminar Cliente**	ID INCORRECTO	ERROR	ERROR	OK
CP11CU06	**Iniciar Venta**	PRECONDICION CLIENTES EXISTEN	LA VENTA CONTINUA	LA VENTA CONTINUA	OK
CP12CU06	**Iniciar Venta**	PRECONDICION CLIENTES NO EXISTE	ERROR NO HAY CLIENTES	ERROR NO HAY CLIENTES	OK
CP13CU07	**Añadir al Carrito**	CANTIDAD POSTIVIA	PEDIDO AÑADIDO	PEDIDO AÑADIDO	OK
CP14CU07	**Añadir al Carrito**	CANTIDAD NEGATIVA	ERROR CANTIDAD	ERROR CANTIDAD	OK
CP15CU08	**Calcular Total**	PRECONDICION PEDIDO EXISTE	MUESTRA EL PEDIDO CON LA CANTIDAD TOTAL	MUESTRA EL PEDIDO CON LA CANTIDAD TOTAL	OK
CP16CU08	**Calcular Total**	PRECONDICION PEDIDO NO EXISTE	NO SE PUEDE SELECCIONAR VENTA	NO SE PUEDE SELECCIONAR VENTA	OK
CP17CU09	**Finalizar Venta**	SALDO SUFICIENTE	VENTA CREADA	VENTA CREADA	OK

CP18CU09	**Finalizar Venta**	SALDO INSUFICIENTE	VENTA REVERTIDA	VENTA REVERTIDA	OK
CP19CU10	**Cancelar Venta**	VENTA A MEDIAS	VENTA CANCELADA	VENTA CANCELADA	OK

9. Dificultades encontradas y soluciones

• Al principio encontre problemas sobre todo al llamar a procesos y funciones, pero se pudo solucionar gracias a los archivos del aula virtual y la documentacion que sale al poner el raton sobre los metodos de las clases

10. Conclusiones y líneas de mejora

• El trabajo me ha permitido mejorar mis conocimientos con respecto al manejo de los datos en las BBDD con java ya que se a avanzado mas que el curso pasado en cuanto a temario.

11. Bibliografía y recursos

• Inteligencia Artificial: Chat GPT y Gemini para poder contrastar resultados para encontrar la mejor opcion.

• Archivos y PDF del aula virtual.

• Documentacion de las propias clases.

12. Anexos

• A. Script SQL (creación y datos mínimos).

• B. Imagenes de los casos de pruebas.

• C. Diagrama UML.

INSTITUTO DE EDUCACIÓN SECUNDARIA

ISIDRA DE GUZMÁN