Reporte Técnico del Proyecto Java – Conexión a MySQL

1. Introducción

Este reporte analiza el funcionamiento de un proyecto desarrollado en Java con conexión a una base de datos MySQL. Se explican las funciones principales, la estructura del código, las librerías utilizadas y la lógica detrás de cada parte del programa. La explicación hace referencia a números de líneas de los archivos DB.java, MainApp.java y proyectofinal.sql, indicando qué hace cada bloque, por qué se implementó de esa manera y cómo afecta al funcionamiento global del sistema.

2. Librerías utilizadas

El programa hace uso de librerías estándar de Java:

java.sql (Connection, DriverManager, PreparedStatement, ResultSet, CallableStatement): se usan para establecer la conexión con MySQL, ejecutar consultas seguras y llamar procedimientos almacenados.

java.math.BigDecimal: permite manejar importes monetarios con precisión, evitando los errores comunes de double en cálculos financieros.

javax.swing y javax.swing.table.DefaultTableModel: construyen la interfaz gráfica de usuario (ventanas, botones, tablas) y administran los datos que aparecen en tablas visuales.

java.awt y java.awt.event**: manejan layouts, paneles y la captura de eventos de usuario (clics, escritura, etc.).

Estas librerías permiten que la aplicación tenga una interfaz sencilla pero funcional y que la comunicación con la base de datos sea consistente y segura.

3. Archivo DB.java

En este archivo se concentra toda la lógica de conexión a MySQL.

Líneas 5-7: se definen las constantes URL, USER y PASS. Contienen los parámetros de conexión a la base de datos. Al estar en un solo archivo, si cambian las credenciales o el host, basta con modificar estas líneas y toda la aplicación seguirá funcionando.

Línea 10: se declara el método estático getConnection(). Este método devuelve un objeto Connection que abre la comunicación con MySQL.

Línea 11: se utiliza DriverManager.getConnection(URL, USER, PASS). Este comando ejecuta la conexión real. Si la URL o la contraseña no son correctas, el método lanzará una excepción. Esto es crucial porque el resto de la aplicación depende de obtener este objeto para poder ejecutar sentencias SQL.

Impacto: DB.java centraliza el acceso a la base de datos y simplifica el mantenimiento. Cualquier cambio en las credenciales no afecta a los demás archivos.

4. Archivo MainApp.java

Este archivo contiene la interfaz de usuario y la lógica de interacción con la base de datos.

Líneas 20-50: se crean los elementos gráficos de la ventana, como pestañas, paneles y botones. Esto establece la estructura básica de la aplicación para dividir el área de “Hacer pedidos” y el “Panel de administración”.

Líneas 60-90 (cargarPedidos): aquí se implementa la lógica para leer la información de los pedidos. Se ejecuta un SELECT con JOIN sobre la tabla Pedido y otras relacionadas. Los resultados se recorren con un ResultSet y se agregan a un DefaultTableModel. Este modelo se usa para actualizar la tabla gráfica en la interfaz. El resultado es que el usuario puede visualizar todos los pedidos registrados ordenados por fecha.

Líneas 100-130 (cargarIngredientes): en este bloque se consultan los ingredientes y su proveedor. Se hace un JOIN entre Ingrediente y Proveedor. La tabla resultante se llena en la interfaz gráfica, lo que permite al administrador revisar inventario en tiempo real.

Líneas 140-180 (agregarIngrediente): este método toma los valores ingresados en los campos de texto de la interfaz, los valida y genera un INSERT INTO Ingrediente. Tras insertar, se vuelve a ejecutar cargarIngredientes() para refrescar la tabla y mostrar el nuevo ingrediente. Esto garantiza que los cambios se vean inmediatamente reflejados.

Líneas 200-250 (crear pedido): este bloque maneja el flujo de creación de un pedido. Primero se inserta un registro en la tabla Pedido (cabecera). Luego se busca el producto por nombre usando LOWER(...) para ignorar mayúsculas y minúsculas, se obtiene su precio con BigDecimal y se inserta un renglón en DetallePedido. Aquí el subtotal se guarda inicialmente en cero.

Líneas 260-280 (confirmar pedido): se llama al procedimiento almacenado sp_confirmar_pedido usando CallableStatement. Este SP calcula la suma de los subtotales en DetallePedido y actualiza el campo total de Pedido, además de cambiar el estado a “confirmado”. Con esto se asegura que la lógica de negocio esté centralizada en la base de datos.

Líneas 300-330 (login y CardLayout): aquí se valida el acceso al panel de administración. Si el usuario y contraseña son correctos, se usa CardLayout para

cambiar de vista (de login a administración). Esta decisión es importante porque mantiene todo en una sola ventana y permite una navegación más fluida.

Impacto: MainApp.java es el corazón de la aplicación. Maneja tanto la UI como la interacción con la base de datos. Cada método implementa un flujo de negocio claro (ver pedidos, gestionar inventario, crear pedidos, confirmar totales).

5. Base de datos y procedimiento almacenado

El archivo proyectofinal.sql define las tablas y las relaciones:

Cliente: guarda los datos personales y un teléfono único que lo identifica.

Empleado: personal que atiende pedidos.

Producto: catálogo con nombre y precio.

Pedido: cabecera con sucursal, empleado y estado.

DetallePedido: contiene los productos, cantidad y precio.

Proveedor e Ingrediente: administración de inventario.

El procedimiento almacenado sp_confirmar_pedido:

Recibe un id_pedido.

Suma los subtotales de todos los detalles de ese pedido.

Actualiza el campo total y cambia el estado del pedido a “confirmado”.

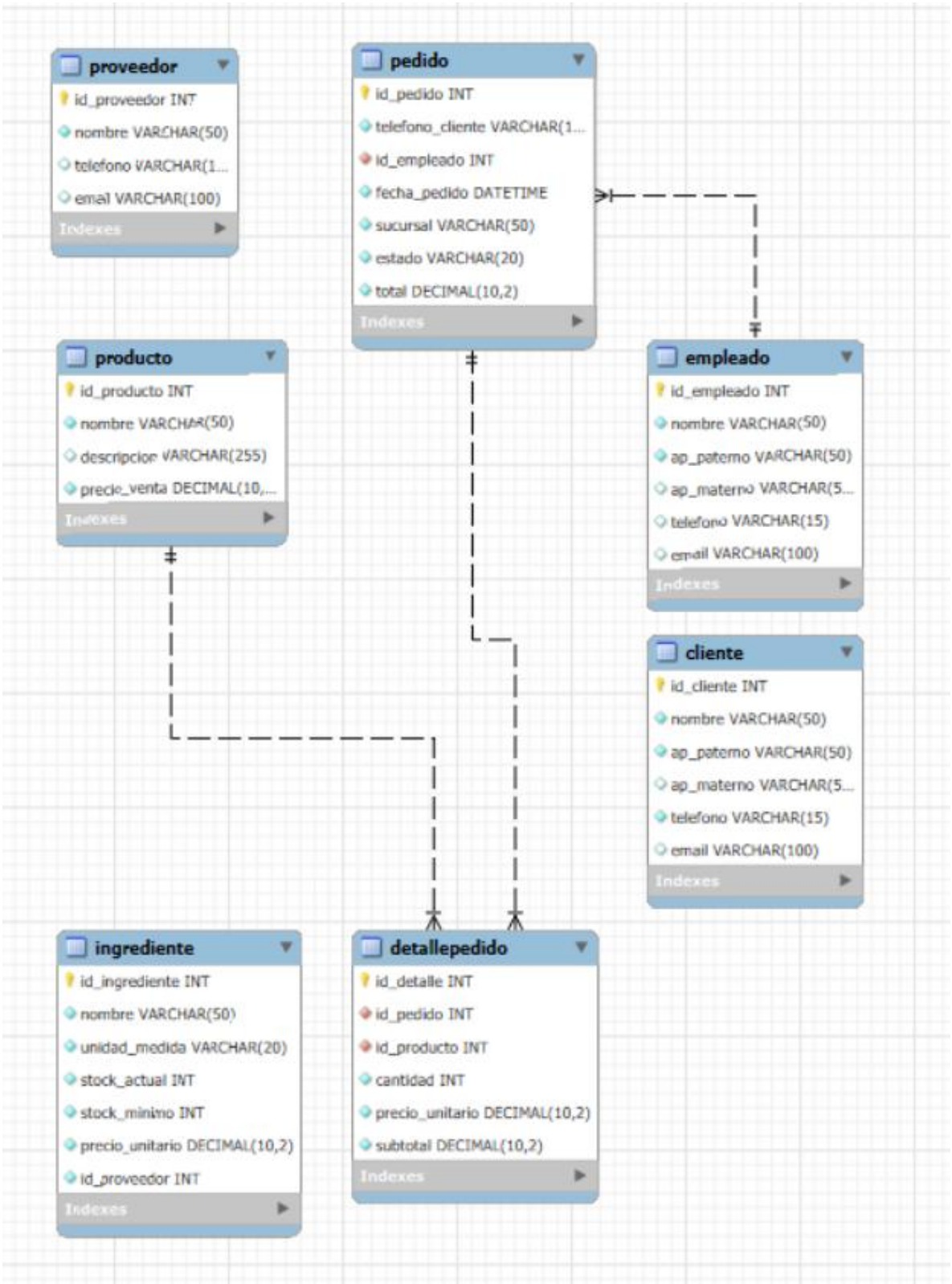
Impacto: este diseño permite una relación clara 1 a N entre pedidos y productos. Al centralizar el cálculo en el SP, se evita que la lógica se repita en el código Java, y se asegura que siempre exista consistencia en los datos.

6. Conclusiones

El proyecto está estructurado de manera simple y funcional. DB.java se encarga únicamente de la conexión con la base de datos, mientras que MainApp.java gestiona tanto la interfaz como la lógica de negocio. El uso de PreparedStatement protege contra inyección SQL, BigDecimal asegura precisión en cálculos monetarios, y el procedimiento almacenado garantiza consistencia en los totales.

Aunque es una estructura básica, es adecuada para un proyecto académico de tercer semestre porque demuestra claramente cómo interactúan Java y SQL. Además, es un ejemplo práctico de cómo dividir la aplicación en componentes que cumplen funciones específicas: conexión, interfaz, lógica de negocio y persistencia de datos.

Diagrama ERR


proveedor
id_proveedor INT
nombre VARCHAR(50)
telefono VARCHAR(1...
email VARCHAR(100)
Indexes
pedido
id_pedido INT
telefono_cliente VARCHAR(1...
id_empleado INT
fecha_pedido DATETIME
sucursal VARCHAR(50)
estado VARCHAR(20)
total DECIMAL(10,2)
Indexes
producto
id_producto INT
nombre VARCHAR(50)
descripcion VARCHAR(255)
precio_venta DECIMAL(10,...
Indexes
empleado
id_empleado INT
nombre VARCHAR(50)
ap_paterno VARCHAR(50)
ap_materno VARCHAR(5...
telefono VARCHAR(15)
email VARCHAR(100)
Indexes
cliente
id_cliente INT
nombre VARCHAR(50)
ap_paterno VARCHAR(50)
ap_materno VARCHAR(5...
telefono VARCHAR(15)
email VARCHAR(100)
Indexes
ingrediente
id_ingrediente INT
nombre VARCHAR(50)
unidad_medida VARCHAR(20)
stock_actual INT
stock_minimo INT
precio_unitario DECIMAL(10,2)
id_proveedor INT
Indexes
detallepedido
id_detalle INT
id_pedido INT
id_producto INT
cantidad INT
precio_unitario DECIMAL(10,2)
subtotal DECIMAL(10,2)
Indexes

Las tablas de ingrediente y proveedor aun no están conectadas porque a futuro se hará una función para que con cada pedido se vaya restando a los ingredientes e inventario.