Universidad Mariano Gálvez
Facultad de Ingeniería en Sistemas
Centro universitario Boca del Monte
Ingeniería en Sistemas
Programación II
Sección b

Proyecto Final
Administración de juguetería

José Javier Antonio Juárez Payes
7690-22-12482

Introducción

En la era digital, la gestión eficiente de inventarios y ventas se ha vuelto esencial para el éxito de cualquier negocio. El sistema de facturación para juguetería propuesto busca automatizar y optimizar los procesos relacionados con la venta de juguetes, proporcionando una herramienta eficaz para llevar un registro preciso de las transacciones y el inventario disponible.

Administración de Juguetería.

El sistema se ha desarrollado utilizando el lenguaje de programación Java y se basa en el modelo de bases de datos relacional MariaDB para almacenar y recuperar datos de manera eficiente. La conexión entre la aplicación Java y la base de datos se logra mediante el uso de JDBC (Java Database Connectivity).

Estructura de la Base de Datos

El diseño de la base de datos se centra en tres entidades clave: juguete, inventario, y facturación.

1. Tabla juguete:

Almacena información general sobre los juguetes, como su identificador único, nombre y precio unitario.

```sql
USE jugueteria;

CREATE TABLE juguete (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(45) NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    color VARCHAR(15) NOT NULL,
    marca VARCHAR(40) NOT NULL,
    piezas INT NOT NULL
);

SELECT * FROM JUGUETE;

INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES ('Pelota', 12.99, 'Rojo', 'Nike', 1);

-- Insert 2
INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES ('Rompecabezas', 19.99, 'Azul', 'Ravensburger', 500);

-- Insert 3
INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES ('Muñeca', 29.99, 'Rosado', 'Barbie', 1);

-- Insert 4
INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES ('Coche de juguete', 15.99, 'Verde', 'Hot Wheels', 1);

-- Insert 5
INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES ('Peluche', 9.99, 'Marrón', 'Disney', 1);
```

2. Tabla inventario:

Registra la cantidad disponible de cada juguete en el inventario.

```sql
CREATE TABLE inventario (
    idInventario INT AUTO_INCREMENT PRIMARY KEY,
    idJuguete INT NOT NULL,
    cantidad INT NOT NULL,
    clasiJuguete VARCHAR(50) NOT NULL,
    FOREIGN KEY (idJuguete) REFERENCES juguete(id)
);

INSERT INTO juguete (nombre, precio, color, marca, piezas) VALUES
    ('Pelota', 9.99, 'Rojo', 'MarcaA', 10),
    ('Muñeca', 19.99, 'Rosado', 'MarcaB', 5),
    ('Rompecabezas', 14.99, 'Azul', 'MarcaC', 3);


INSERT INTO inventario (idJuguete, cantidad, clasiJuguete) VALUES
    (1, 20, 'Pelota'),
    (2, 15, 'Muñeca'),
    (3, 5, 'Rompecabezas');
```

3. Tabla factura:

Guarda información sobre las transacciones de venta, incluyendo el identificador del juguete, nombre, cantidad vendida y el total.

```sql
USE jugueteria;

CREATE TABLE factura (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATE NOT NULL,
    cliente_nombre VARCHAR(50) NOT NULL,
    total DECIMAL(10,2) NOT NULL
);

INSERT INTO factura (fecha, cliente_nombre, total) VALUES
    ('2023-01-01', 'Cliente1', 39.97),
    ('2023-01-02', 'Cliente2', 29.98);
```

Funcionalidades Principales
El sistema de facturación para juguetería ofrece las siguientes funcionalidades clave:

4. **Realizar Venta:**

Permite al usuario seleccionar un juguete, ingresar la cantidad deseada y completar la transacción. El sistema actualiza automáticamente el inventario y registra la venta en la tabla de facturación.

5. **Consultar Juguetes:**

Proporciona una lista de los juguetes disponibles, permitiendo al usuario revisar la información básica de cada uno.

Manejo de Excepciones y Seguridad

El código ha sido desarrollado teniendo en cuenta las mejores prácticas de seguridad y manejo de excepciones. Se utilizan consultas parametrizadas con PreparedStatement para prevenir la inyección de SQL y se implementa un manejo adecuado de excepciones para garantizar la estabilidad y confiabilidad del sistema.

Manual Java Administración de Juguetería

La clase **ProyectoFinalR** en el proyecto Java actúa como la interfaz principal de la aplicación. Proporciona un menú interactivo que permite al usuario realizar operaciones relacionadas con la gestión de juguetes, como consultar, crear y eliminar juguetes. La clase utiliza instancias de las clases Juguete y ConexionDB para llevar a cabo estas operaciones.

Estructura del Código

- ✓ Método main(String[] args):

Punto de entrada principal de la aplicación.
Crea instancias de las clases Juguete y ConexionDB para realizar operaciones de gestión de juguetes.
Presenta un menú interactivo al usuario y realiza la operación seleccionada.

- Menú Interactivo:

- Ofrece opciones numéricas al usuario para seleccionar una operación.
- Muestra las siguientes opciones:
- 1: Consultar juguete.
- 2: Crear nuevo juguete.
- 3: Eliminar juguete.
- 0: Salir del menú.

- ✓ Selección de Operación:

Utiliza una estructura de control switch para ejecutar la operación correspondiente según la elección del usuario.

```
1
2   package proyectofinalr;
3
4   import java.sql.SQLException;
5   import java.util.*;
6   public class ProyectoFinalR {
7       public static void main(String[] args)throws SQLException {
8           Scanner entrada = new Scanner(source: System.in);
9           Juguete juguete= new Juguete();
10          ConexionDB conexiondb= null;
11
12          int opcion;
13          System.out.println(x: "Ingrese el la opcion que necesite");
14          System.out.println(x: "1.Consultar Juguete");
15          System.out.println(x: "2.Crear nuevo Juguete");
16          System.out.println(x: "3.Eliminar Juguete");
17          System.out.println(x: "0.Salir del Menu");
18          opcion= entrada.nextInt();
19
20
21
22          switch(opcion){
23              case 1:
24                  juguete.consultarJuguete();
25                  break;
26              case 2:
27                  juguete.nuevoJuguete();
28                  break;
29              case 3:
30                  juguete.eliminarJuguete();
31                  break;
32          }
33      }
34  }
```

La clase **ConexionDB** en el proyecto Java proporciona funcionalidades esenciales para establecer y gestionar la conexión con una base de datos MariaDB. Utiliza la biblioteca JDBC (Java Database Connectivity) y el controlador JDBC de MariaDB para interactuar con la base de datos. La clase incluye métodos para realizar consultas, obtener la conexión actual y cerrar la conexión.

Estructura del Código

✓ Constructor ConexionDB:

- Inicializa la conexión con la base de datos MariaDB al instanciar un objeto de la clase.
- Carga el controlador JDBC de MariaDB mediante la línea Class.forName.
- Configura la URL de conexión con información como la dirección IP del servidor, el puerto y el nombre de la base de datos.
- Utiliza el método DriverManager.getConnection para establecer la conexión, especificando el nombre de usuario y la contraseña.

✓ Método consultar (String sql):

- Crea una declaración (Statement) a partir de la conexión establecida.
- Ejecuta una consulta SQL especificada por el parámetro sql.
- Retorna un conjunto de resultados (ResultSet) que contiene los datos obtenidos de la consulta.

✓ Método getConnection():

- Retorna la conexión actual, permitiendo que otras clases accedan a la conexión establecida.

```java
package proyectofinalr;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class ConexionDB {
    private Connection conexion;

    public ConexionDB() throws SQLException{
        try{
            Class.forName(className: "org.mariadb.jdbc.Driver");
            String url = "jdbc:mariadb://" + "127.0.0.1" + ":" + "3306" + "/" + "jugueteria";
            this.conexion = DriverManager.getConnection(url, user: "root", password: "1234");

        }catch (ClassNotFoundException e) {
            throw new SQLException("Error al cargar el controlador de MariaDB: " + e.getMessage());
        }
    }

        public ResultSet consultar(String sql) throws SQLException {
        Statement sentencia = conexion.createStatement();
        return sentencia.executeQuery(string: sql);
    }

        public Connection getConnection() {
        return conexion;
    }

        public void cerrar() throws SQLException {
        if (conexion != null) {
            conexion.close();
        }
    }
}
```

La clase **Juguete** en el proyecto Java proporciona métodos para realizar operaciones relacionadas con la gestión de juguetes en la base de datos. Incluye funciones para consultar la información de los juguetes, crear nuevos juguetes y eliminar juguetes existentes. La clase utiliza instancias de la clase ConexionDB para interactuar con la base de datos MariaDB.

Estructura del Código

- ✓ Constructor Juguete:

- Inicializa una instancia de la clase ConexionDB para permitir la interacción con la base de datos.

- ✓ Método consultarJuguete():

- Consulta y muestra en la consola la información de todos los juguetes almacenados en la base de datos.

- ✓ Método nuevoJuguete():

- Interactúa con el usuario para ingresar información sobre un nuevo juguete.
- Utiliza consultas parametrizadas para insertar el nuevo juguete en la tabla 'juguete'.
- Muestra el ID del nuevo juguete agregado con éxito.

- ✓ Método eliminarJuguete():

- Interactúa con el usuario para ingresar el ID de un juguete que desea eliminar.
- Utiliza una consulta parametrizada para eliminar el juguete de la tabla 'juguete'.
- Informa al usuario sobre el éxito o la falta de éxito en la eliminación.

```java

package proyectofinalr;

import java.sql.PreparedStatement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
public class Juguete {
    private ConexionDB conexiondb;

    public Juguete() throws SQLException {
        conexiondb = new ConexionDB();
    }

     public void consultarJuguete(){
        try{
        String consultas = "SELECT * FROM juguete";
        ResultSet consulta = conexiondb.consultar(sql: consultas);

        while (consulta.next()){
        int id = consulta.getInt(string: "id");
        String nombre = consulta.getString(string: "nombre");
        Double precio = consulta.getDouble(string: "precio");
        String color = consulta.getString(string: "color");
        String marca = consulta.getString(string: "marca");
        int piezas = consulta.getInt(string: "piezas");

        System.out.print("nombre: " + nombre + " - ");
        System.out.print(":precio " + precio + " - ");
        System.out.print("color: " + color + " - ");
        System.out.print("marca: " + marca + " - ");
        System.out.print("piezas: " + piezas + " - ");
            }
    }catch (SQLException e){
        e.printStackTrace();
    }
```

La clase **CONSULTA** en el proyecto Java proporciona métodos para realizar consultas específicas en la base de datos. Esta clase utiliza instancias de la clase ConexionDB para interactuar con la base de datos MariaDB y realizar consultas sobre las tablas 'inventario', 'juguete' y 'pedido'.

✓ consultaInventario():

- Realiza una consulta en las tablas 'inventario' y 'juguete'.
- Muestra en la consola información detallada sobre el inventario, incluyendo el tipo de juguete, clasificación, y cantidad.

✓ consultarInventario():

- Realiza una consulta en las tablas 'inventario' y 'juguete'.
- Muestra en la consola información básica sobre el inventario, incluyendo el tipo de juguete, clasificación, y cantidad.

✓ consultarJuguete():

- Realiza una consulta en la tabla 'juguete'.
- Muestra en la consola información detallada sobre los juguetes, incluyendo el tipo de juguete, tamaño y precio.

✓ consultarPedido():

- Realiza una consulta en la tabla 'pedido'.
- Muestra en la consola información detallada sobre los pedidos, incluyendo el ID, juguete asociado, cantidad y precio.

✓ obtenerTipoJuguetePorId(int idJuguete):

- Obtiene el tipo de juguete asociado a un ID de juguete específico.
- Utiliza una consulta en la tabla 'juguete' para recuperar la información.

✓ obtenerCantidadInventarioPorIdJuguete(int idJuguete):

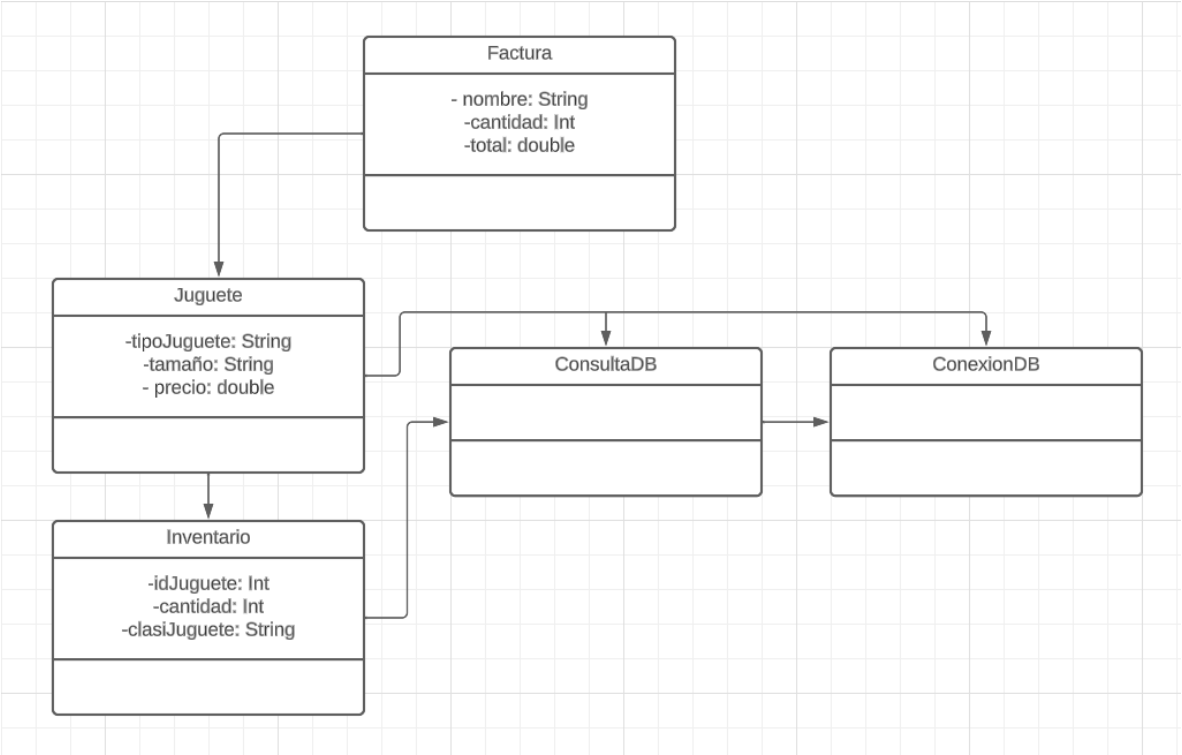
- Obtiene la cantidad disponible en el inventario para un ID de juguete específico.
- Utiliza una consulta en la tabla 'inventario' para recuperar la información.

✓ obtenerPrecioJuguetePorId(int idJuguete):

- Obtiene el precio asociado a un ID de juguete específico.
- Utiliza una consulta en la tabla 'juguete' para recuperar la información.

```java
package proyectofinalr;

import java.sql.ResultSet;
import java.sql.SQLException;

public class CONSULTA {
    private ConexionDB conexiondb;
    public  CONSULTA() throws SQLException{
            conexiondb = new ConexionDB();
    }
    public String consultaInventario(){
        String mensaje ="";
        try{
        String inv = "SELECT * FROM inventario";
        String jug = "SELECT * FROM juguete";
        ResultSet resultados = conexiondb.consultar(sql:inv);
        ResultSet ju = conexiondb.consultar(sql:jug);


        while (resultados.next()){
            int idJuguete = resultados.getInt(string: "idJuguete");
            int cantidad = resultados.getInt(string: "cantidad");
            String clasiJuguete = resultados.getString(string: "clasiJuguete");

            while (ju.next()){
            int idJugueteJug = ju.getInt(string: "idJuguete");
            if (idJugueteJug == idJuguete) {
            String tipoJuguete = ju.getString(string: "tipoJuguete");
                mensaje = "juguete: " + tipoJuguete + " - clasificacion: "
                        + clasiJuguete + " - cantidad: " + cantidad;
                System.out.print("juguete: " + tipoJuguete + " - ");
                System.out.print("clasificacion: " + clasiJuguete + " - ");
                System.out.println("cantidad: " + cantidad);

            }

            }
            ju.beforeFirst();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
        return mensaje;
}
```

Diagrama de Clases



Conclusión

El proyecto Java desarrollado para la gestión de una juguetería demuestra una estructura organizada con clases como Juguete, ConexionDB, CONSULTA, y ProyectoFinalR. La conexión a la base de datos MariaDB se realiza eficientemente mediante la clase ConexionDB. La aplicación ejecuta operaciones CRUD, como inserción y eliminación de juguetes, consulta de inventario, y presentación de datos en la consola. La interactividad con el usuario se logra a través de un menú en ProyectoFinalR. Se observa un manejo básico de excepciones con try-catch. Se sugiere mejorar la robustez del código, validar la entrada del usuario y perfeccionar la presentación de resultados.