PRÁCTICA Final E.D.

EFA
EL CAMPICO

Entornos de desarrollo

Autor:

Pedro Mula Montesinos

Jacarilla, 9 de Junio de 2024

1. Introducción

La Tienda Virtual es un proyecto desarrollado para crear una aplicación web de gestión y venta de productos en línea. Utilizando metodologías ágiles, el proyecto busca asegurar entregas rápidas y de alta calidad, proporcionando a los usuarios finales una experiencia de compra eficiente y segura. A lo largo del ciclo de desarrollo, se han implementado diversas funcionalidades esenciales para el funcionamiento de la tienda.

2. Fase de Ingeniería del Software

2.1. Descripción del Proyecto

Propósito: El propósito de la tienda virtual es proporcionar una plataforma en línea donde los usuarios puedan comprar productos diversos. El objetivo es facilitar la compra de productos mediante un sistema intuitivo y seguro.

Visión General: La tienda virtual permite a los usuarios:

- Ver una lista de productos disponibles.
- Realizar compras de productos.
- Ver y gestionar sus pedidos.

2.2. Diagramas

Diagrama de Clases: El diagrama de clases muestra la estructura del sistema, incluyendo las clases, sus atributos, métodos y las relaciones entre ellas.

Clases Principales:

- Cliente: Representa a los usuarios que pueden hacer compras en la tienda.
    - Atributos: id, name, email, password, address, city, country, registrationDate
    - Métodos: getId(), getName(), getEmail(), getPassword(), getAddress(), getCity(), getCountry(), getRegistrationDate()
- Producto: Representa los productos disponibles para la venta.
    - Atributos: id, name, price, stock, available
    - Métodos: getId(), getName(), getPrice(), getStock(), isAvailable()
- Pedido: Registra los pedidos realizados por los clientes.
    - Atributos: id, clientId, date, total
    - Métodos: getId(), getClientId(), getDate(), getTotal()

- Compra: Registra las compras realizadas por los clientes.
    - Atributos: id, clientId, date, total
    - Métodos: getId(), getClientId(), getDate(), getTotal()
- GestionProducto: Gestiona los productos de la tienda.
    - Atributos: productId, productName, price, stock
    - Métodos: getProductId(), getProductName(), getPrice(), getStock()
- Menu: Clase para mostrar el menú de la aplicación.
    - Métodos: mostrarMenu(), mostrarMensajeFinal()

Relaciones:

- Cliente tiene una relación uno a muchos con Pedido y Compra.
- Producto tiene una relación muchos a muchos con Pedido a través de DetallePedido y con Compra a través de DetalleCompra.
- Albaran y Proveedor están relacionados con Producto.

Enlace al Diagrama de Clases: GitHub - Diagrama de Clases

Diagrama de Actividad: Este diagrama ilustra el flujo de trabajo dentro de la aplicación, desde la autenticación del cliente hasta la finalización de una compra.

Enlace al Diagrama de Actividad: GitHub - Diagrama de Actividad

3. Fase de Diseño de la Base de Datos

3.1. Descripción de las Tablas

Cliente:

- Atributos: id, nombre, email, clave, direccion, ciudad, pais, fecha_registro
- Descripción: Guarda información sobre los clientes.

Producto:

- Atributos: id, nombre, descripcion, precio, stock, fecha_registro
- Descripción: Almacena detalles de los productos disponibles.

Proveedor:

- Atributos: id, nombre, email, direccion, ciudad, pais, telefono, fecha_registro

- Descripción: Contiene información de los proveedores de los productos.

Albarán:

- Atributos: id, proveedor_id, fecha
- Descripción: Registra transacciones con proveedores.

Pedido:

- Atributos: id, cliente_id, fecha, total
- Descripción: Registra los pedidos realizados por los clientes.

Ticket:

- Atributos: id, compra_id, fecha, total
- Descripción: Guarda información de las ventas realizadas.

PedidoProducto:

- Atributos: pedido_id, producto_id, cantidad, precio_unitario
- Descripción: Relaciona pedidos con productos.

Musica, Cine, Videojuego:

- Descripción: Tablas específicas para diferentes tipos de productos.

3.2. Relaciones entre Tablas

Las relaciones entre las tablas están visualizadas mediante diagramas de entidad-relación (ERD) para proporcionar una comprensión clara de cómo interactúan las tablas entre sí.

Código SQL para la Base de Datos:

```
-- Crear la base de datos TiendaVirtual

CREATE DATABASE IF NOT EXISTS TiendaVirtual;


-- Usar la base de datos TiendaVirtual

USE TiendaVirtual;
```

```
-- Tabla de Clientes
CREATE TABLE IF NOT EXISTS Clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    clave VARCHAR(100) NOT NULL,
    direccion VARCHAR(200),
    ciudad VARCHAR(100),
    pais VARCHAR(100),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de Compras
CREATE TABLE IF NOT EXISTS Compras (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT,
    fecha_compra TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);

-- Tabla de Detalles de Compra
CREATE TABLE IF NOT EXISTS DetallesCompra (
    id INT AUTO_INCREMENT PRIMARY KEY,
    compra_id INT,
```

```
    producto_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (compra_id) REFERENCES Compras(id)
);

-- Tabla de Tickets de Compra
CREATE TABLE IF NOT EXISTS TicketsCompra (
    id INT AUTO_INCREMENT PRIMARY KEY,
    compra_id INT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total DECIMAL(10, 2),
    FOREIGN KEY (compra_id) REFERENCES Compras(id)
);

-- Tabla de Productos
CREATE TABLE IF NOT EXISTS Productos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    precio DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL,
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Tabla de Proveedores
CREATE TABLE IF NOT EXISTS Proveedores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    direccion VARCHAR(200),
    ciudad VARCHAR(100),
    pais VARCHAR(100),
    telefono VARCHAR(20),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de Albaranes
CREATE TABLE IF NOT EXISTS Albaranes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    proveedor_id INT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (proveedor_id) REFERENCES Proveedores(id)
);

-- Tabla de Detalles de Albaran
CREATE TABLE IF NOT EXISTS DetallesAlbaran (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    albaran_id INT,
    producto_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (albaran_id) REFERENCES Albaranes(id),
    FOREIGN KEY (producto_id) REFERENCES Productos(id)
);

-- Tabla de Pedidos
CREATE TABLE IF NOT EXISTS Pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);

-- Tabla de Detalles de Pedido
CREATE TABLE IF NOT EXISTS DetallesPedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT,
    producto_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
```

```
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(id),
    FOREIGN KEY (producto_id) REFERENCES Productos(id)
);
```

4. Fase de Desarrollo

Estructura del Proyecto:

- Jerarquía de Directorios: Explica la estructura del proyecto, la organización de los paquetes y el propósito de cada directorio.

Funciones Clave:

- guardarFichero: Proporciona una descripción de esta función, incluyendo ejemplos de código y su explicación.

Código de la clase Ficheros:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class Ficheros {
    public void guardarFichero(String contenido, String ruta) {
        try {
            FileWriter writer = new FileWriter(new File(ruta));
            writer.write(contenido);
            writer.close();
        } catch (IOException e) {
            System.err.println("Error al guardar el fichero: " + e.getMessage());
        }
```

```
    }
}
```

cargarProductosDesdeBaseDeDatos: Proporciona una descripción y ejemplos de código para la función que carga los productos desde la base de datos.

Código de la clase GestionProducto:

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashMap;
import java.util.Map;

public class GestionProducto {
    private Map<Integer, Producto> productos = new HashMap<>();

    public void cargarProductosDesdeBaseDeDatos() {
        try (Connection conn = Conexion.conectar();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM productos")) {

            while (rs.next()) {
                String tipo = rs.getString("tipo");
                switch (tipo) {
```

```
                case "Musica":
                    productos.put(rs.getInt("id"), new Musica(rs));
                    break;
                case "Cine":
                    productos.put(rs.getInt("id"), new Cine(rs));
                    break;
                case "Videojuego":
                    productos.put(rs.getInt("id"), new Videojuego(rs));
                    break;
                default:
                    productos.put(rs.getInt("id"), new Producto(rs));
            }
        }
    } catch (SQLException e) {
        System.err.println("Error al cargar productos: " + e.getMessage());
    }
  }
}
```

- mostrarProductos: Proporciona una descripción y ejemplos de código para la función que muestra los productos cargados.

Código de la clase GestionProducto:

Fase de Testing

Casos de Prueba

Listado de Casos de Prueba:

1. Caso de Prueba: Mostrar Productos
    - Escenario: Verificar que la lista de productos se muestra correctamente.
    - Datos de Entrada: Lista de productos cargados en la base de datos.
    - Resultado Esperado: La salida debe incluir detalles precisos de cada producto, incluyendo ID, nombre, precio y cantidad.
    - Resultado Obtenido: (Ejemplo: "ID: 1, Nombre: Producto1, Precio: 10.0, Cantidad: 100")
2. Caso de Prueba: Guardar Producto
    - Escenario: Verificar que un nuevo producto puede ser guardado correctamente.
    - Datos de Entrada: Detalles del producto (nombre, precio, cantidad).
    - Resultado Esperado: El producto debe ser añadido a la base de datos y estar disponible en la lista de productos.
    - Resultado Obtenido: (Ejemplo: Producto guardado exitosamente y visible en la lista)
3. Caso de Prueba: Procesar Pedido
    - Escenario: Verificar que un pedido puede ser procesado correctamente.
    - Datos de Entrada: Detalles del pedido (ID del cliente, lista de productos, cantidades).
    - Resultado Esperado: El pedido debe ser registrado en la base de datos y el inventario debe actualizarse.
    - Resultado Obtenido: (Ejemplo: Pedido procesado exitosamente y cantidades actualizadas)
4. Caso de Prueba: Cargar Productos desde la Base de Datos
    - Escenario: Verificar que los productos se cargan correctamente desde la base de datos.
    - Datos de Entrada: Productos almacenados en la base de datos.
    - Resultado Esperado: Los productos deben ser cargados en la aplicación sin errores.
    - Resultado Obtenido: (Ejemplo: Productos cargados exitosamente sin errores)
5. Caso de Prueba: Mostrar Mensaje Final
    - Escenario: Verificar que el mensaje final se muestra correctamente al finalizar una acción.

- Datos de Entrada: Acción completada (por ejemplo, procesar un pedido).
- Resultado Esperado: El mensaje final debe mostrarse correctamente al usuario.
- Resultado Obtenido: (Ejemplo: "Gracias por su compra")

Ejemplos de Pruebas Unitarias:

Prueba de mostrarProductos:

Código de la clase TestMostrarProductos:

```
import static org.junit.Assert.assertTrue;

import java.io.ByteArrayOutputStream;

import java.io.PrintStream;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;


public class TestMostrarProductos {

        private final ByteArrayOutputStream outContent = new ByteArrayOutputStream();

    private final PrintStream originalOut = System.out;


    @Before

    public void setUpStreams() {

        System.setOut(new PrintStream(outContent));

    }


    @After
```

```
    public void restoreStreams() {
        System.setOut(originalOut);
    }

    @Test
    public void testMostrarProductos() {
        GestionProducto gestionProducto = new GestionProducto();
        gestionProducto.cargarProductosDesdeBaseDeDatos(); // Assuming this method works as expected
        gestionProducto.mostrarProductos();
            String expectedOutput = "ID: 1, Nombre: Producto1, Precio: 10.0, Cantidad: 100\n";
        assertTrue(outContent.toString().contains(expectedOutput));
    }
}
```

6. Resumen

Este documento proporciona una guía detallada sobre el desarrollo de la Tienda Virtual, abarcando desde la ingeniería del software hasta el testing final. Cada fase ha sido documentada meticulosamente para asegurar la claridad y la continuidad del proyecto.

7. Objetivos

1. Manejar un volumen de información superior al de prácticas anteriores.
2. Integrar código desarrollado previamente en una nueva aplicación.
3. Mantener estándares de desarrollo y organización de código.

En esta sección, se incluyen los códigos y configuraciones clave utilizados en el desarrollo de la Tienda Virtual.

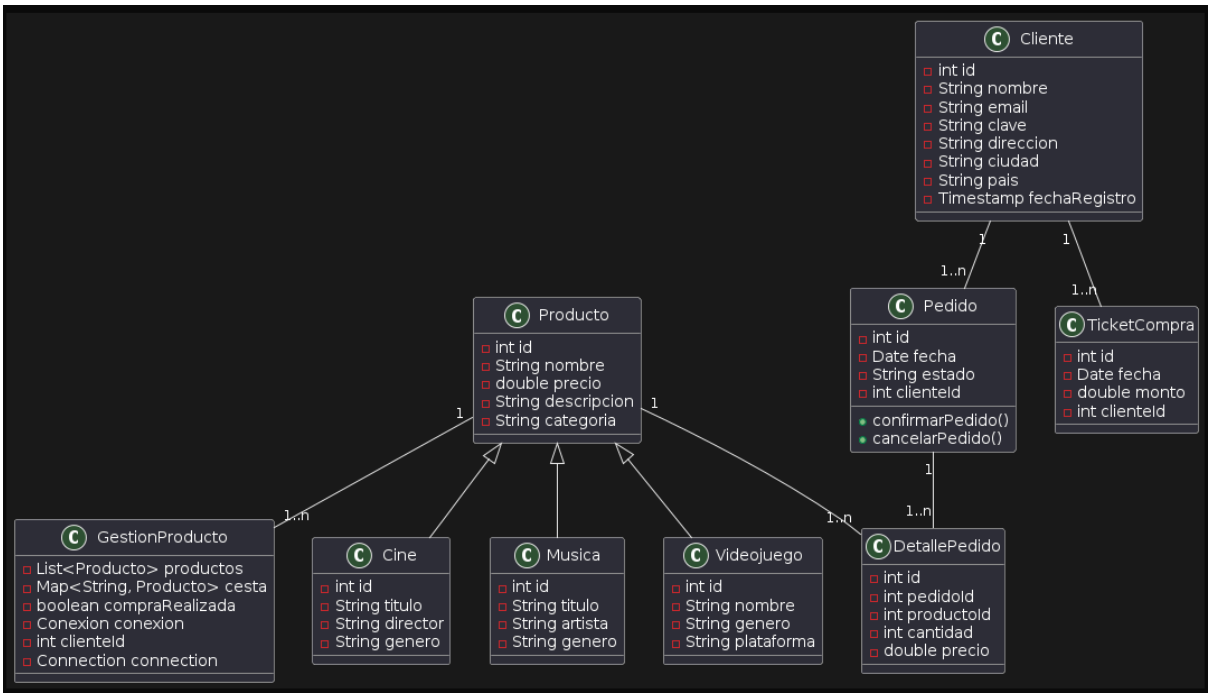
Configuración de la Base de Datos: El código SQL necesario para crear las tablas y relaciones de la base de datos está incluido en la sección 3.2.

Conexión a la Base de Datos:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;


public class Conexion {

    private static final String URL = "jdbc:mysql://localhost:3306/TiendaVirtual";

    private static final String USUARIO = "root";

    private static final String CLAVE = "root";


    public static Connection conectar() {

        Connection conexion = null;

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            conexion = DriverManager.getConnection(URL, USUARIO, CLAVE);

        } catch (ClassNotFoundException | SQLException e) {

            e.printStackTrace();

        }

        return conexion;

    }

}
```

8.Diagrama data y logic

A continuación va un diagrama de clases de los package data y logic.



9. Cierre de la Documentación

Conclusiones

El desarrollo de la Tienda Virtual ha sido un esfuerzo meticuloso que abarcó todas las fases cruciales del ciclo de vida del software. Desde la ingeniería inicial del software y el diseño detallado de la base de datos, hasta la implementación de funcionalidades clave y la realización de pruebas exhaustivas, cada paso ha sido cuidadosamente planeado y ejecutado. Esta documentación proporciona una guía clara para el desarrollo y la configuración del proyecto, asegurando que el sistema final es robusto, eficiente y cumple con los requisitos del usuario.