21GIIN PROYECTOS DE PROGRAMACIÓN

Actividad UC4

Moisés Sevilla Corrales
PROF. Doctor Héctor Simarro Moncholí

27 de enero de 2025

Índice

Desarrollo de un módulo E-commerce y Diseño de Arquitectura en Paradigma Orientado a Objetos.

Descripción

En esta actividad, los estudiantes deberán añadir un nuevo módulo de e-commerce a un sistema existente utilizando el paradigma de programación orientado a objetos. Además, deberán crear diagramas de clases y de casos de uso, y proponer una solución de arquitectura de tres capas.

Objetivo

1. Desarrollar habilidades en la implementación de módulos utilizando el paradigma orientado a objetos.
2. Diseñar y documentar diagramas de clases y de casos de uso.
3. Proponer y justificar una solución de arquitectura de tres capas.

Instrucciones

1. **Análisis del Sistema Existente:** Revise el sistema actual y documente su estructura y funcionalidades principales.
2. **Añadir Módulo de E-commerce:** Desarrolle e integre un módulo de e-commerce al sistema existente. Asegúrese de que el nuevo módulo sea coherente con el propósito del sistema y mejore su funcionalidad.
3. **Diagramas de Clases y Casos de Uso:** Cree los diagramas de clases y de casos de uso para el nuevo módulo de e-commerce. Asegúrese de que los diagramas sean claros y detallados.
4. **Solución de Arquitectura de Tres Capas:** Proporcione una solución de arquitectura de tres capas (presentación, lógica de negocio y datos) para el sistema mejorado. Justifique su elección de arquitectura y explique cómo cada capa interactúa con las demás.

1. Análisis del Sistema Existente

El sistema existente es una aplicación de **gestión de biblioteca** desarrollada con una arquitectura de **tres capas**, orientada a proporcionar una gestión eficiente y segura de los recursos de la biblioteca. Esta aplicación está diseñada para permitir la administración integral de los libros, usuarios y préstamos, asegurando la integridad y disponibilidad de la información.

1.1 Funcionalidades del Sistema Original

- **Gestión de Libros:** Permite agregar, editar, eliminar y buscar libros mediante criterios como título, autor e ISBN.
- **Gestión de Usuarios:** Registro, modificación y eliminación de usuarios que acceden a los servicios de la biblioteca.
- **Gestión de Préstamos:** Registro de préstamos y devoluciones de libros, con control del límite máximo de préstamos por usuario.
- **Validaciones de Datos:** Validación de datos críticos como el ISBN único para evitar duplicados, y verificación de la disponibilidad de libros antes de asignar préstamos.

1.2 Arquitectura de Tres Capas

- **Capa de Presentación (API REST):**
    - Implementada mediante endpoints accesibles desde Postman.
    - Facilita la interacción de los usuarios con el sistema para gestionar libros, usuarios y préstamos.
    - Los endpoints incluyen operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y filtros de búsqueda.

- **Capa de Lógica de Negocio:**
    - Implementa todas las reglas de negocio.
    - Valida los datos ingresados por los usuarios, como el control del ISBN único y la gestión de préstamos.
    - Define restricciones como el límite máximo de libros prestados por usuario.
- **Capa de Acceso a Datos:**
    - Conexión directa con la base de datos MySQL.
    - Realiza operaciones CRUD asegurando la integridad de los datos.
    - Implementa claves foráneas para mantener la integridad referencial.

1.3 Base de Datos del Sistema Original

- **Tabla usuarios:**
    - Almacena información de los usuarios (nombre, email, teléfono, dirección).
- **Tabla libros:**
    - Contiene información de los libros (ISBN, título, autor, editorial, año de publicación).
- **Tabla prestamos:**
    - Registra los préstamos y devoluciones de libros, vinculando usuarios y libros.

Esta estructura modular permitió un desarrollo ordenado, con operaciones seguras y controladas, lo cual facilitó la posterior integración del módulo de **e-commerce**.

2. Añadir Módulo de E-commerce

Se ha integrado un módulo de **e-commerce** al sistema, permitiendo a los usuarios comprar libros directamente desde la biblioteca. Este módulo incluye las siguientes funcionalidades:

2.1 Funcionalidades Implementadas

- **Catálogo de Libros en Venta:** Visualización de todos los libros disponibles para la compra, incluyendo detalles como título, autor, precio y stock.
- **Carrito de Compras:** Los usuarios pueden agregar libros al carrito, modificar la cantidad o eliminar productos antes de proceder a la compra.
- **Gestión de Pedidos:** Registro de compras realizadas por los usuarios, donde cada pedido incluye el detalle de los productos adquiridos y su estado de procesamiento.
- **Proceso de Compra:** Validación de stock antes de confirmar la compra y cálculo automático del total a pagar.
- **Gestión de Stock:** Actualización del stock de productos tras la confirmación de compras.
- **Historial de Pedidos:** Los usuarios pueden consultar sus compras pasadas y verificar el estado de sus pedidos.

2.2 Nuevas Tablas en la Base de Datos

1. **productos:** Almacena los libros disponibles para la venta.
    - id: Identificador único del producto.
    - titulo: Título del libro.
    - autor: Autor del libro.
    - precio: Precio de venta.
    - stock: Cantidad disponible.

2. **carrito:** Almacena los productos seleccionados por cada usuario antes de la compra.
    - id: Identificador único del carrito.
    - usuario_id: ID del usuario propietario del carrito.
    - total: Total acumulado del carrito.

3. **detalle_carrito:** Detalla los productos y cantidades en el carrito.
    - id: Identificador único del detalle.
    - carrito_id: ID del carrito asociado.
    - producto_id: ID del producto agregado.
    - cantidad: Número de unidades seleccionadas.

4. **pedidos:** Registra las compras confirmadas.
    - id: Identificador único del pedido.
    - usuario_id: ID del usuario que realizó la compra.
    - fecha_pedido: Fecha y hora del pedido.
    - estado: Estado actual del pedido (pendiente, enviado, entregado).

5. **detalle_pedido:** Detalla los productos incluidos en cada pedido.
    - id: Identificador único del detalle.
    - pedido_id: ID del pedido asociado.
    - producto_id: ID del producto comprado.
    - cantidad: Número de unidades adquiridas.
    - precio: Precio unitario del producto al momento de la compra.

```sql
-- Tabla productos
CREATE TABLE IF NOT EXISTS productos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(100) NOT NULL,
    autor VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL
);

-- Tabla carrito
CREATE TABLE IF NOT EXISTS carrito (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    total DECIMAL(10, 2) DEFAULT 0,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);

-- Tabla detalle_carrito
CREATE TABLE IF NOT EXISTS detalle_carrito (
    id INT AUTO_INCREMENT PRIMARY KEY,
    carrito_id INT NOT NULL,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    FOREIGN KEY (carrito_id) REFERENCES carrito(id),
    FOREIGN KEY (producto_id) REFERENCES productos(id)
);

-- Tabla pedidos
CREATE TABLE IF NOT EXISTS pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    fecha_pedido DATETIME DEFAULT CURRENT_TIMESTAMP,
    estado VARCHAR(50) DEFAULT 'pendiente',
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);

-- Tabla detalle_pedido
CREATE TABLE IF NOT EXISTS detalle_pedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT NOT NULL,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (pedido_id) REFERENCES pedidos(id),
    FOREIGN KEY (producto_id) REFERENCES productos(id)
);
```

- Se implementan los módulos de e-commerce con clases y objetos POO de la lógica de negocio.

- Ficheros Java adjuntos en github: ***https://github.com/moisessevilla/biblioteca***
    - Usuario.java
    - Carrtio.java
    - Producto.java
    - Pedido.java
    - DetalleCarrito.java
    - DetallePedido.java

2.3 Flujo del Proceso de Compra

1. **Selección de Productos:** El usuario navega por el catálogo y agrega productos al carrito.
2. **Gestión del Carrito:** El usuario puede modificar cantidades o eliminar productos.
3. **Confirmación de Compra:** Al confirmar el pedido, el sistema verifica el stock y procesa la compra.
4. **Registro del Pedido:** Se crea un registro en la tabla pedidos y se actualiza el stock en la tabla productos.
5. **Seguimiento del Pedido:** El usuario puede consultar el estado de su pedido hasta su entrega.

3. Diagramas de Clases y Casos de Uso

3.1 Diagrama de Clases

- El diagrama de clases describe las entidades y sus relaciones dentro del módulo de e-commerce:

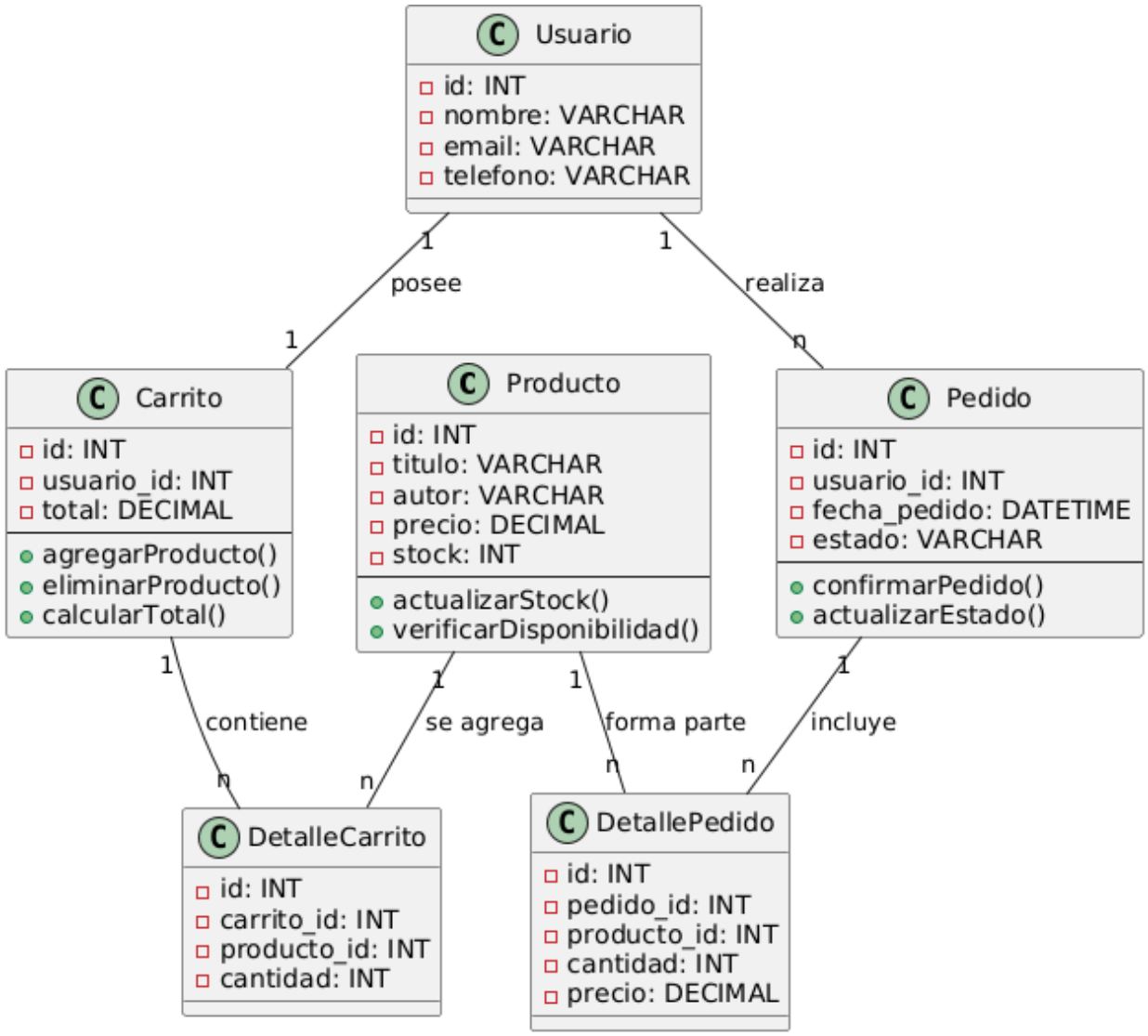
- **Producto:**
  - Atributos: id, titulo, autor, precio, stock
  - Métodos: actualizarStock(), verificarDisponibilidad()

- **Carrito:**
  - Atributos: id, usuario_id, total
  - Métodos: agregarProducto(), eliminarProducto(), calcularTotal()

- **DetalleCarrito:**
  - Atributos: id, carrito_id, producto_id, cantidad

- **Pedido:**
  - Atributos: id, usuario_id, fecha_pedido, estado
  - Métodos: confirmarPedido(), actualizarEstado()

- **DetallePedido:**
  - Atributos: id, pedido_id, producto_id, cantidad, precio


Usuario
id: INT
nombre: VARCHAR
email: VARCHAR
telefono: VARCHAR
1
posee
1
1
realiza
n
Carrito
id: INT
usuario_id: INT
total: DECIMAL
agregarProducto()
eliminarProducto()
calcularTotal()
Producto
id: INT
titulo: VARCHAR
autor: VARCHAR
precio: DECIMAL
stock: INT
actualizarStock()
verificarDisponibilidad()
Pedido
id: INT
usuario_id: INT
fecha_pedido: DATETIME
estado: VARCHAR
confirmarPedido()
actualizarEstado()
1
contiene
n
1
se agrega
n
1
forma parte
n
1
incluye
n
DetalleCarrito
id: INT
carrito_id: INT
producto_id: INT
cantidad: INT
DetallePedido
id: INT
pedido_id: INT
producto_id: INT
cantidad: INT
precio: DECIMAL

3.2 Diagrama de Casos de Uso

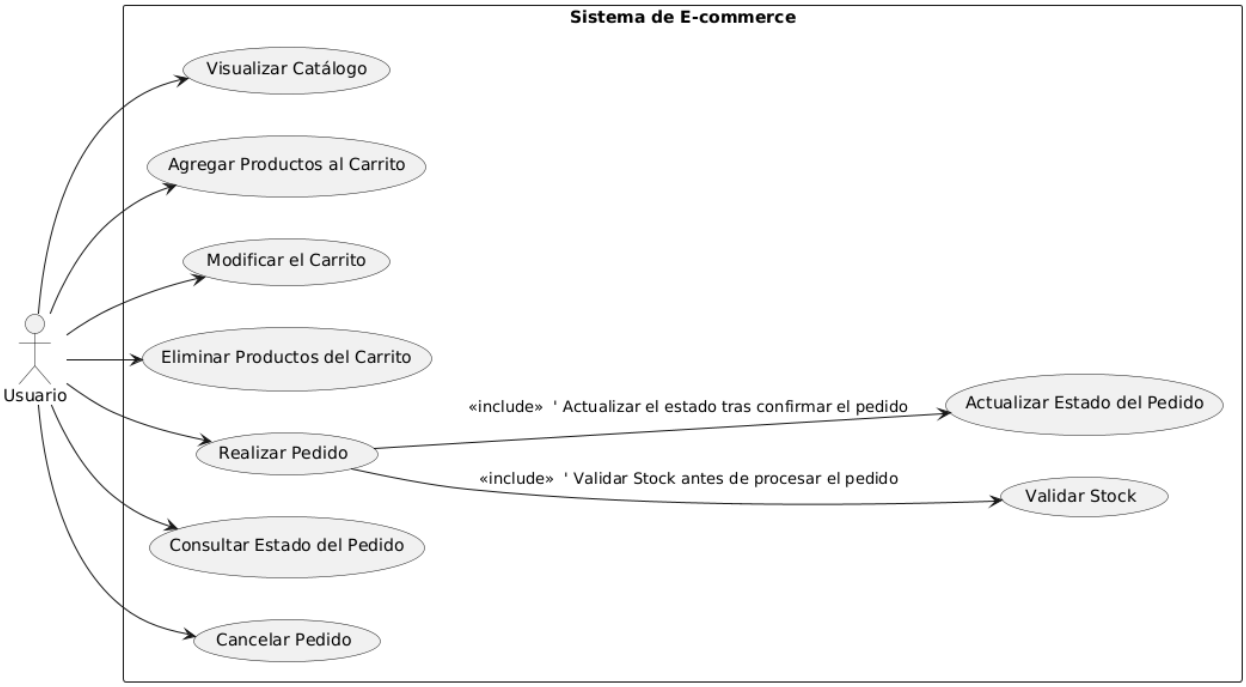
Actor Principal: Usuario

Casos de Uso:

1. **Visualizar Catálogo:** El usuario consulta los productos disponibles.
2. **Agregar Productos al Carrito:** El usuario selecciona productos para compra.
3. **Modificar el Carrito:** Puede actualizar cantidades o eliminar productos.
4. **Realizar Pedido:** El usuario confirma la compra, generando un pedido.
5. **Consultar Estado del Pedido:** El usuario verifica el estado de sus compras.
6. **Cancelar Pedido:** Permite anular pedidos si aún no han sido procesados.

Relaciones:

- Asociación directa entre el usuario y las acciones sobre el carrito y pedidos.
- Inclusión de validaciones automáticas de stock y cálculo de totales.



4. Propuesta de Arquitectura de Tres Capas

Capa de Presentación (API REST)

- **Endpoints para el Módulo E-commerce:**
    - GET /productos: Listar productos disponibles.
    - POST /carrito: Agregar producto al carrito.
    - PUT /carrito/:id: Modificar cantidad de un producto.
    - DELETE /carrito/:id: Eliminar producto del carrito.
    - POST /pedidos: Confirmar compra.
    - GET /pedidos/:usuarioId: Consultar pedidos.
    - GET /pedidos/:id/detalle: Consultar detalle de un pedido específico.
    - PUT /pedidos/:id/estado: Actualizar el estado de un pedido.

Capa de Lógica de Negocio

- **Validaciones:** Verificar stock disponible antes de confirmar compra, validación de datos de productos y usuarios.
- **Cálculo de Totales:** Cálculo automático del total en el carrito al agregar o eliminar productos.
- **Gestor de Pedidos:** Gestión del flujo de los pedidos (pendiente, procesando, enviado, entregado, cancelado).
- **Control de Stock:** Disminución automática de stock tras la confirmación de compra y restricción de compra si no hay suficiente stock.

Capa de Acceso a Datos

- **Integridad Referencial:** Uso de claves foráneas para asegurar la relación entre tablas (usuario_id, producto_id).

- **CRUD Completo:** Implementación de todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para los productos, carrito, pedidos y sus detalles.

- **Optimización de Consultas:** Índices en campos de búsqueda frecuente como producto_id, usuario_id y estado de pedidos para mejorar el rendimiento.

- **Transacciones:** Uso de transacciones para garantizar la coherencia de los datos durante la compra.

5. Justificación de la Arquitectura

La arquitectura de **tres capas** garantiza:

- **Modularidad:** Permite dividir la aplicación en componentes independientes, facilitando el desarrollo y mantenimiento.

- **Escalabilidad:** La separación de capas permite agregar nuevas funcionalidades sin afectar el sistema existente.

- **Mantenibilidad:** Los cambios en una capa no afectan directamente a las otras, mejorando la capacidad de mantenimiento.

- **Seguridad:** El aislamiento de las capas permite implementar medidas de seguridad específicas en cada una, protegiendo los datos sensibles.

- **Rendimiento:** La optimización de consultas SQL y el uso de transacciones aseguran una rápida respuesta del sistema.

- **Flexibilidad:** Facilita la integración con otros sistemas o módulos adicionales en el futuro.

- **Facilidad de Pruebas:** La estructura modular facilita la implementación de pruebas unitarias e integrales.

Conclusión

- La integración del módulo de e-commerce ha ampliado significativamente las capacidades del sistema de gestión de biblioteca, transformándolo en una plataforma más versátil y funcional. Este avance permite no solo la gestión eficiente de los recursos bibliográficos, sino también la posibilidad de comercializar los libros, brindando una experiencia más completa y satisfactoria para los usuarios.

- La aplicación mantiene la separación de responsabilidades mediante la arquitectura de tres capas, garantizando flexibilidad, seguridad y escalabilidad para futuras ampliaciones. La correcta implementación de validaciones, reglas de negocio y la optimización de las consultas en la base de datos asegura un rendimiento óptimo y una gestión eficiente de los datos.

- La modularidad del sistema facilita la incorporación de nuevas funcionalidades sin comprometer la estabilidad del sistema existente. Esta estructura permite adaptarse rápidamente a las nuevas necesidades y tecnologías, posicionando al sistema como una solución integral y sostenible a largo plazo.

- El desarrollo del módulo de e-commerce no solo cumple con los objetivos planteados, sino que también sienta las bases para la evolución continua del sistema, permitiendo futuras integraciones y mejoras orientadas a ofrecer un servicio cada vez más completo y eficiente.

Bibliografía

Universidad Internacional de Valencia (VIU).

Libro Proyectos de programación, por Dr. Roger Clotet Martínez.

El código utilizado para la realización de esta actividad se encuentra en GitHub.

https://github.com/moisessevilla/biblioteca