JHONY ANDRES MIRA GAVIRIA

C.C 71.263.544

TALLER POO CON C#

Sistema pedidos cafetería

Programación Distribuida

PDN-2025-2-PDI74-3

Martes – jueves 8:00pm

Fecha: 16 -09 -2025

Profesor: Daniel Andrey Villamizar Araque

Facultad de Ingenierías

Instituto Tecnológico Metropolitano

Medellín

2025

Caso de Negocio

Problema

Actualmente, el proceso de toma de pedidos y gestión de inventario en la cafetería "Trigo Dorado" es manual y propenso a errores. Los empleados anotan los pedidos en papel, lo que causa demoras, equivocaciones en los totales y dificulta el seguimiento de las ventas y el control del stock de productos. No hay una forma centralizada de aplicar promociones o de saber qué productos necesitan ser repuestos urgentemente.

Usuarios

- **Personal cafetería/Cajero:** Necesitan una forma rápida y precisa de registrar los pedidos, calcular los totales y procesar pagos para agilizar el servicio al cliente.
- **Administrador/Dueño de la Cafetería:** Requiere un sistema para gestionar los productos, monitorear el inventario, ver el historial de pedidos y analizar las ventas para tomar decisiones de negocio informadas.
- **Cliente:** Espera que su pedido sea tomado correctamente, que el total sea exacto y que el proceso de pago sea eficiente.

Valor Esperado

- **Eficiencia:** Reducir el tiempo de toma de pedidos y el procesamiento de pagos.
- **Precisión:** Eliminar errores humanos en el cálculo de totales y la gestión del inventario.
- **Control de Inventario:** Recibir alertas cuando un producto está por agotarse, evitando la pérdida de ventas.
- **Flexibilidad:** Permitir la fácil implementación de nuevas reglas de negocio, como diferentes estrategias de precio o métodos de pago, sin cambiar la lógica central del sistema.

Historias de Usuario

1. **Como** cajero, **quiero** poder crear un nuevo pedido **para** un cliente.
    - **Dado** que estoy en la pantalla de inicio del sistema de pedidos.
    - **Cuando** ingreso los datos del cliente y presiono "Crear Pedido".
    - **Entonces** el sistema me muestra un nuevo pedido listo para agregar productos.
2. **Como** cajero, **quiero** poder agregar múltiples productos a un pedido **para** calcular el total rápidamente.
    - **Dado** un pedido abierto.
    - **Cuando** selecciono un producto (ej. "Café Americano") y una cantidad.
    - **Entonces** el sistema agrega el producto al pedido y actualiza el subtotal.
3. **Como** cajero, **quiero** poder procesar un pedido con diferentes métodos de pago (efectivo, tarjeta) **para** ofrecer flexibilidad al cliente.
    - **Dado** un pedido con productos y un total calculado.
    - **Cuando** selecciono "Pagar con tarjeta de crédito".
    - **Entonces** el sistema valida el pago y muestra una confirmación "Pago OK".
4. **Como** cajero, **quiero** que el sistema me alerte **cuando** el stock de un producto se agote, **para** informar al cliente y evitar vender algo que no tengo.
    - **Dado** que el stock de "Pastel de Chocolate" es 5.
    - **Cuando** se vende el quinto pastel.
    - **Entonces** el sistema me muestra un mensaje de alerta: " ALERTA: Stock bajo para Pastel de Chocolate".
5. **Como** administrador, **quiero** que el sistema aplique descuentos automáticamente **cuando** el cliente cumple con una condición, **para** incentivar las compras grandes.
    - **Dado** un pedido con 6 productos.
    - **Cuando** el sistema calcula el total.

- **Entonces** el sistema aplica un descuento del 10% al total del pedido.

6. **Como** administrador, **quiero** poder listar todos los pedidos guardados **para** tener un registro de las ventas.
    - **Dado** que se han guardado varios pedidos.
    - **Cuando** accedo a la opción de "Historial de pedidos".
    - **Entonces** el sistema muestra una lista de todos los pedidos, con su ID, fecha, cliente y total.

Requerimientos

Requerimientos de Negocio (RN)

1. El sistema debe permitir la gestión de un catálogo de productos con precio y stock.
2. El sistema debe soportar la creación de pedidos que contienen uno o varios productos.
3. El sistema debe poder procesar pagos con al menos dos métodos: efectivo y tarjeta de crédito.
4. El sistema debe ser capaz de aplicar diferentes estrategias de precio al total del pedido.
5. El sistema debe guardar un registro de todos los pedidos procesados.

Requerimientos Funcionales (RF)

1. El sistema debe permitir crear un Cliente con nombre y documento.
2. El sistema debe permitir crear un Producto con un código, nombre, precio y stock.
3. El sistema debe permitir asociar un Cliente a un Pedido.
4. El sistema debe permitir agregar una LineaPedido a un Pedido, con un producto y una cantidad.

5. El sistema debe descontar la cantidad de producto del stock disponible al momento de agregarlo al pedido.
6. El sistema debe calcular el total de un Pedido sumando los subtotales de todas sus líneas.
7. El sistema debe permitir autorizar un pago usando diferentes implementaciones de IMedioPago.
8. El sistema debe notificar cuando el stock de un Producto cae por debajo de un umbral predefinido.

Requerimientos No Funcionales (RNF)

1. **Rendimiento:** Las operaciones de creación y adición a un pedido deben ser casi instantáneas (menos de 500ms).
2. **Seguridad Básica:** Los datos sensibles del pago (ej. número de tarjeta) no deben mostrarse por completo en la consola, solo una versión enmascarada.
3. **Logging:** El sistema debe imprimir en la consola un registro claro de cada paso importante (creación de pedido, pago, etc.).
4. **Mantenibilidad:** La aplicación debe estar organizada en carpetas (Domain, Repositories, Pricing) para que sea fácil de entender y modificar.
5. **Testabilidad:** La lógica de negocio (Pedido, Producto, IMedioPago) debe ser separable de la lógica de presentación (la consola), lo que facilita las pruebas unitarias.
6. **UX Consola:** La interfaz de la consola debe ser clara y legible, con mensajes informativos para el usuario.

Diseño POO (aplicar los 4 pilares)

Abstracción: Definir Contratos y Modelos

La abstracción se logra al centrarse en las ideas esenciales del sistema y ocultar los detalles complejos. En el proyecto, esto se ve en:

- **Interfaces**: La interfaz **IRepositorioEmpleados**. Define el contrato para cualquier clase que necesite manejar los datos de empleados, como el método ObtenerPorDocumento(). La implementación específica (RepositorioEmpleadosMemoria) esconde los detalles de cómo se buscan los datos (por ejemplo, en una lista en memoria), pero el contrato sigue siendo el mismo.
- **Clases de dominio**: Clases como **Producto**, **Pedido** y **Empleado** son abstracciones de entidades del mundo real. No se necesita saber cómo se calcula el subtotal de una línea de pedido para usarla, solo se necesita saber que la propiedad Subtotal dará el valor correcto.

Encapsulamiento: Proteger el Estado de los Objetos

El encapsulamiento se logra al agrupar los datos (propiedades) y los métodos que operan sobre esos datos en una sola unidad (la clase). Esto protege el estado interno de los objetos y asegura que los cambios se realicen de manera controlada.

- **Propiedades privadas**: Las propiedades con un private set como public string Codigo { get; private set; } en la clase Producto impiden que el código externo cambie directamente el código de un producto.
- **Métodos de control**: El método **AgregarLinea()** en la clase Pedido realiza esta función. En lugar de que el código externo manipule el stock o la lista de líneas del pedido, la lógica se centraliza dentro de este método. Al llamar a producto.Descontar(), el Pedido delega la responsabilidad del stock al Producto, asegurando que el proceso de inventario se maneje de forma segura y consistente.
- **Validaciones**: Las validaciones en los constructores y métodos (por ejemplo, en Producto para el stock) garantizan que un objeto nunca pueda estar en un estado inválido.

Herencia: Jerarquías de Clases

La herencia permite crear una jerarquía de clases donde una clase "hija" hereda atributos y comportamientos de una clase "padre".

- **Jerarquía de Persona**: Se tiene una jerarquía con la clase base **Persona** y las clases derivadas **Empleado** y **Cliente**. Ambas clases heredan las propiedades básicas de una persona como Documento y Nombre, evitando la duplicación de código. Esto permite tratar a un Empleado o a un Cliente como una Persona en las partes del código que solo necesitan esa información genérica.

Polimorfismo: Múltiples Formas de un Comportamiento

El polimorfismo permite que objetos de diferentes clases respondan al mismo mensaje (método) de maneras distintas, logrando un código flexible y extensible.

- **Estrategias de Pricing**: Las clases como DescuentoPorCantidadStrategy y PrecioConIvaStrategy implementan la misma interfaz (IPricingStrategy). Esto permite aplicar diferentes lógicas de precios de forma intercambiable. En el código, se puede usar una lista de estas estrategias para aplicar varios cálculos al precio final de un pedido.
- **Medios de pago**: Las clases Efectivo, TarjetaCredito y Transferencia implementan la misma interfaz **IMedioPago**. Cuando se procesa el pago en CrearPedido(), no se necesita saber si el cliente usó efectivo o tarjeta, solo llamar al método **Autorizar()** y el objeto correcto se encarga de la lógica específica de cada medio de pago.
- **Override de ToString()**: Aunque no es una interfaz, el método ToString() se usa polimórficamente en varias clases (Producto, LineaPedido) para proporcionar una representación en texto única y personalizada para cada tipo de objeto.

Modelo Conceptual

- **Cliente:** Una persona que realiza un pedido. Un cliente puede tener muchos pedidos.
- **Producto:** Un artículo disponible en la cafetería (ej. Café, Pastel). Un producto se puede encontrar en muchas líneas de pedido.
- **Pedido:** Una solicitud de compra hecha por un cliente. Un pedido contiene una o más líneas de pedido.
- **Línea de Pedido:** Representa un producto específico y su cantidad dentro de un pedido. Cada línea está relacionada con un solo producto y un solo pedido.
- **Medio de Pago:** Un método para pagar el pedido (ej. Efectivo, Tarjeta de Crédito).

Repositorio :

https://github.com/jhonym90340/Sistema_pedido_cafeter-a.git