

**PROGRAMACION DISTRIBUIDA**

**INGENIERIA DE SISTEMAS**

**Autor(es):**

**Stiven Osorio Roldan**

**Mateo Martinez Franco**



**Instituto Tecnológico Metropolitano - ITM**

**Medellín**

**2025**

## Documento de Análisis – Taller POO en C# (.NET 8 – Consola)

### 1. Caso de negocio

#### Problema:

Una cafetería de barrio necesita tomar pedidos de manera más eficiente. Actualmente los totales se calculan manualmente, lo que genera errores en los precios y retrasos en la atención. Además, no existe un historial de pedidos, lo cual dificulta el control de ventas y la toma de decisiones del negocio.

#### Usuarios principales:

- **Cajero/Barista:** Encargado de registrar los pedidos de los clientes y procesar los pagos.
- **Cliente:** Solicita productos del catálogo y paga por ellos.
- **Administrador:** Revisa el historial de pedidos y los totales de ventas para control interno.

#### Valor esperado:

La aplicación permitirá registrar pedidos de forma digital, calcular automáticamente los totales, reducir errores humanos y conservar un historial básico de transacciones. Esto agilizará la atención y dará soporte al administrador para la toma de decisiones.

---

### 2. Historias de usuario

#### 1. Crear pedido

- *Como cajero, quiero crear un pedido vacío, para iniciar la captura de productos.*
- **Criterios de aceptación:**
  - **Given** que no hay un pedido abierto, **When** selecciono la opción “Nuevo pedido”, **Then** se crea un pedido con un ID único y estado “Abierto”.

#### 2. Agregar producto al pedido

- *Como cajero, quiero agregar productos con su cantidad, para armar el pedido del cliente.*

- **Criterios de aceptación:**
  - **Given** un pedido abierto, **When** agrego un producto válido con cantidad  $> 0$ , **Then** el sistema recalcula el subtotal.

### 3. Eliminar producto del pedido

- *Como cajero, quiero eliminar o corregir un producto del pedido, para reflejar la orden correcta del cliente.*
- **Criterios de aceptación:**
  - **Given** un pedido con líneas, **When** elimino una línea, **Then** el subtotal se actualiza automáticamente.

### 4. Calcular total con propina opcional

- *Como cajero, quiero calcular el total de un pedido con impuestos/propina, para dar el monto final al cliente.*
- **Criterios de aceptación:**
  - **Given** un pedido con ítems, **When** selecciono la opción de calcular con propina del 10%, **Then** el sistema muestra subtotal, propina y total.

### 5. Pagar con diferentes medios

- *Como cajero, quiero procesar el pago en efectivo o tarjeta, para cerrar el pedido.*
- **Criterios de aceptación:**
  - **Given** un pedido listo, **When** selecciono medio de pago y confirmo, **Then** el sistema valida y marca el pedido como “Pagado”.

### 6. Guardar historial de pedidos

- *Como administrador, quiero que los pedidos pagados se guarden, para revisarlos después.*
- **Criterios de aceptación:**
  - **Given** que existe persistencia habilitada, **When** un pedido es cerrado, **Then** el pedido se guarda en memoria o archivo JSON.

---

### 3. Requerimientos de negocio (mín. 5)

- RN-1: Los precios de los productos deben estar definidos en un catálogo central.
- RN-2: Cada pedido debe tener un identificador único.
- RN-3: El sistema debe permitir procesar al menos dos métodos de pago.
- RN-4: El cálculo del total debe incluir subtotal y propina opcional.
- RN-5: El administrador debe poder revisar pedidos históricos del día.

---

### 4. Requerimientos funcionales (mín. 8)

- RF-1: Crear pedido con estado inicial “Abierto”.
- RF-2: Agregar productos a un pedido.
- RF-3: Eliminar productos de un pedido.
- RF-4: Listar productos disponibles en catálogo.
- RF-5: Calcular subtotal y total de un pedido.
- RF-6: Procesar pago con efectivo o tarjeta.
- RF-7: Guardar pedidos pagados en memoria o JSON.
- RF-8: Consultar el historial de pedidos.
- RF-9: Mostrar mensajes de validación en consola ante errores.

---

### 5. Requerimientos no funcionales (mín. 6)

- RNF-1: **Rendimiento:** las operaciones deben ejecutarse en menos de 200 ms.
- RNF-2: **Seguridad básica:** no almacenar datos sensibles de tarjetas.
- RNF-3: **Logging:** registrar operaciones clave en archivo logs/app.log.
- RNF-4: **Mantenibilidad:** el código debe estar organizado por capas (Dominio, Servicios, Infraestructura, UI).

- RNF-5: **Testabilidad**: lógica desacoplada de la interfaz de consola para permitir pruebas unitarias.
  - RNF-6: **UX consola**: menús simples y mensajes claros para el usuario.
- 

## 6. Modelo conceptual

### Entidades principales

#### 1. Producto

- **Atributos**: id, nombre, descripción, precio, disponible.
- **Subclases**: **Bebida** (atributos adicionales: tamaño, esFria) y **Comida** (atributos adicionales: tipoComida, requiereCalentar)

#### 2. Pedido

- **Atributos**: id, fechaCreación, fechaPago, estado (Abierto, Pagado), subtotal, propina, total, metodoPago.
- **Relación**: un pedido contiene varias Líneas de Pedido.

#### 3. Línea de Pedido

- **Atributos**: producto, cantidad, precioUnitario, subtotal.
- **Relación**: referencia a un Producto.

#### 4. Historial

- Agrupa los Pedidos registrados (por día y todos).
- Métodos para obtener el total de ventas del día y los productos más vendidos.

#### 5. Pago

- **Estrategia de pago**: Efectivo o Tarjeta.
  - Cada pago tiene información específica (monto recibido, últimos dígitos de tarjeta).
- 

### Relaciones

- Pedido -- contiene --> Línea de Pedido -- referencia --> Producto

- Producto -- hereda --> Bebida
  - Producto -- hereda --> Comida
  - Historial -- agrupa --> Pedidos
  - Pedido -- se paga mediante --> Pago (Efectivo/Tarjeta)
  - Pedido -- se guarda en --> Historial
  - Pago -- utiliza --> Estrategia de pago (patrón estrategia)
- 

### **Diagrama de relaciones (descripción textual)**

- Un Pedido tiene varias Líneas de Pedido.
  - Cada Línea de Pedido está asociada a un Producto.
  - Un Producto puede ser una Bebida (subtipo).
  - El Historial gestiona muchos Pedidos.
  - Un Pedido se asocia a un método de Pago.
  - El sistema soporta diferentes estrategias de pago.
- 

## **7. Diseño POO (4 pilares)**

### **1. Abstracción:**

El sistema define clases que representan los conceptos principales de una cafetería, como Producto, Pedido, Línea de Pedido, Estrategia de Pago, etc. Cada clase abstrae los detalles internos y expone solo lo necesario para interactuar con ella. Por ejemplo, un Producto tiene nombre, precio y descripción, ocultando cómo se calcula el precio o cómo se almacena internamente.

### **2. Encapsulamiento:**

Las clases protegen sus datos internos y solo permiten acceder o modificar su estado mediante métodos públicos controlados. Por ejemplo, la clase Pedido

controla el agregado y eliminado de productos a través de métodos, sin exponer directamente la lista interna de productos. Así se evita que el estado del sistema sea alterado de forma incorrecta o insegura.

### 3. Herencia:

La herencia se utiliza para modelar especializaciones. Producto es una clase abstracta que representa cualquier ítem vendible. De Producto derivan Bebida y Comida, que heredan sus atributos y métodos, pero pueden agregar o modificar comportamientos específicos (por ejemplo, Bebida tiene atributos como tamaño y si es fría).

### 4. Polimorfismo:

El polimorfismo está presente en el manejo de pagos. Las clases PagoEfectivo y PagoTarjeta implementan la misma interfaz (I EstrategiaPago), permitiendo al sistema procesar pagos de distintas formas usando el mismo método. Así, el sistema puede tratar diferentes tipos de pagos de manera uniforme, y es fácil agregar nuevas formas de pago en el futuro sin modificar el resto del sistema.

## Prueba Xunit:

Como ejecutar: “dotnet test”

```
PowerShell para desarrolladores
+ PowerShell para desarrolladores - [?] [?] [?]
PS C:\Users\mateo\source\repos\CafeteriaApp\cafeteriaapp> dotnet test
Restauración completada (0,3s)
CafeteriaApp prueba-error (0,2s) - bin\Debug\net8.0\CafeteriaApp.dll
Error:
An assembly specified in the application dependencies manifest (testhost.deps.json) was not found:
package: 'Newtonsoft.Json', version: '13.0.3'
path: 'lib\net8.0\Newtonsoft.Json.dll'
El proceso del host de prueba para fuentes "C:\Users\mateo\source\repos\CafeteriaApp\cafeteriaapp\bin\Debug\net8.0\CafeteriaApp.dll" finalizó con el siguiente error: Error:
An assembly specified in the application dependencies manifest (testhost.deps.json) was not found:
package: 'Newtonsoft.Json', version: '13.0.3'
path: 'lib\net8.0\Newtonsoft.Json.dll'
Consulte los registros de diagnóstico para obtener más información.
CafeteriaApp prueba-error con 1 errores (0,3s)
C:\Users\mateo\source\repos\CafeteriaApp\bin\Debug\net8.0\CafeteriaApp.dll : error TESTRUNABORT: Serie de pruebas anulada.
Compilación error con 1 errores en 1,2s
PS C:\Users\mateo\source\repos\CafeteriaApp\cafeteriaapp>
```

```
PowerShell para desarrolladores
+ PowerShell para desarrolladores - [?] [?] [?]
PS C:\Users\mateo\source\repos\CafeteriaApp> dotnet test
Restauración completada (0,7s)
CafeteriaApp prueba-error con 16 advertencias (0,4s) - CafeteriaApp\bin\Debug\net8.0\CafeteriaApp.dll
C:\Users\mateo\source\repos\CafeteriaApp\Infrastructure\Logging\FileLogger.cs(42,61): warning CS8625: No se puede convertir un literal NULL en un tipo de referencia que no acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\PedidoService.cs(20,16): warning CS8618: El elemento campo "pedidoActual" que no acepta valores NULL debe contener un valor distinto de NULL al salir del constructor. Considere la posibilidad de agregar el modificador "required" o declarar el campo como un valor que acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\CatalogService.cs(15,16): warning CS8618: El elemento campo "productos" que no acepta valores NULL debe contener un valor distinto de NULL al salir del constructor. Considere la posibilidad de agregar el modificador "required" o declarar el campo como un valor que acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\Infrastructure\Logging\FileLogger.cs(22,32): warning CS8600: Se va a convertir un literal nulo o un posible valor nulo en un tipo que no acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Infrastructure\Logging\FileLogger.cs(25,43): warning CS8604: Posible argumento de referencia nulo para el parámetro "path" en "DirectoryInfo.Directory.CreateDirectory(string path)".
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\CatalogService.cs(39,20): warning CS8604: Posible tipo de valor devuelto de referencia nulo.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Domain\Models\Pedido.cs(21,16): warning CS8618: El elemento propiedad "metodoPago" que no acepta valores NULL debe contener un valor distinto de NULL al salir del constructor. Considere la posibilidad de agregar el modificador "required" o declarar el campo como un valor que acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\Infrastructure\Pagos\PagoTarjeta.cs(19,30): warning CS8601: Posible asignación de referencia nula.
C:\Users\mateo\source\repos\CafeteriaApp\Infrastructure\Pagos\PagoTarjeta.cs(14,24): warning CS8618: El elemento campo "ultimoDigito" que no acepta valores NULL debe contener un valor distinto de NULL al salir del constructor. Considere la posibilidad de agregar el modificador "required" o declarar el campo como un valor que acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\Infrastructure\Persistence\IItemRepository.cs(47,37): warning CS8600: Se va a convertir un literal nulo o un posible valor nulo en un tipo que no acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Infrastructure\Persistence\ItemRepository.cs(50,47): warning CS8604: Posible argumento de referencia nulo para el parámetro "path" en "DirectoryInfo.Directory.CreateDirectory(string path)".
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Infrastructure\Persistence\ItemRepository.cs(52,20): warning CS8604: Posible tipo de valor devuelto de referencia nulo.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Presentation\ConsoleHelper.cs(76,20): warning CS8600: Posible tipo de valor devuelto de referencia nulo.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\PedidoService.cs(92,13): warning CS8625: No se puede convertir un literal NULL en un tipo de referencia que no acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\PedidoService.cs(106,23): warning CS8625: No se puede convertir un literal NULL en un tipo de referencia que no acepta valores NULL.
C:\Users\mateo\source\repos\CafeteriaApp\CafeteriaApp\Services\MenuPrincipalTests.cs(155,21): warning xUnit1013: Public method 'Dispose' on test class 'MenuPrincipalTests' should be marked as a Fact. Reduce the visibility of the method, or add a Fact attribute to the method. (https://xunit.net/xunit.analyzers/rules/xunit1013)
Error:
An assembly specified in the application dependencies manifest (testhost.deps.json) was not found:
package: 'Newtonsoft.Json', version: '13.0.3'
path: 'lib\net8.0\Newtonsoft.Json.dll'
El proceso del host de prueba para fuentes "C:\Users\mateo\source\repos\CafeteriaApp\cafeteriaapp\bin\Debug\net8.0\CafeteriaApp.dll" finalizó con el siguiente error: Error:
An assembly specified in the application dependencies manifest (testhost.deps.json) was not found:
package: 'Newtonsoft.Json', version: '13.0.3'
path: 'lib\net8.0\Newtonsoft.Json.dll'
Consulte los registros de diagnóstico para obtener más información.
CafeteriaApp prueba-error con 1 errores (0,3s)
C:\Users\mateo\source\repos\CafeteriaApp\bin\Debug\net8.0\CafeteriaApp.dll : error TESTRUNABORT: Serie de pruebas anulada.
Compilación error con 1 errores y 16 advertencias en 1,9s
PS C:\Users\mateo\source\repos\CafeteriaApp>
```