Taller de Microservicios con Arquitectura Hexagonal: Sistema de Gestión de Pedidos de Restaurante

Caso de Negocio

Una cadena de restaurantes necesita un sistema para gestionar pedidos en línea. El sistema debe manejar la creación de pedidos, gestionar el inventario de ingredientes y facilitar la comunicación entre el área de recepción de pedidos y la cocina.

Microservicios a Implementar

1. Microservicio de Pedidos

Responsable de la gestión del ciclo de vida de los pedidos: creación, actualización, y seguimiento.

2. Microservicio de Inventario

Encargado de gestionar el inventario de ingredientes, validar disponibilidad para nuevos pedidos y actualizar existencias.

Dominio y Contextos Delimitados

Contexto Delimitado: Gestión de Pedidos

- **Entidades**: Pedido, Cliente, Ítem de Pedido
- **Objetos de Valor**: Dirección de Entrega, Total del Pedido

- **Agregados**: Pedido (raíz)
- **Repositorios**: Repositorio de Pedidos
- **Servicios de Dominio**: Servicio de Creación de Pedidos, Servicio de Cálculo de Precios

Contexto Delimitado: Gestión de Inventario

- **Entidades**: Ingrediente, Receta
- **Objetos de Valor**: Cantidad, Unidad de Medida
- **Agregados**: Ingrediente (raíz)
- **Repositorios**: Repositorio de Ingredientes, Repositorio de Recetas
- **Servicios de Dominio**: Servicio de Validación de Disponibilidad, Servicio de Actualización de Inventario

Flujo de Trabajo

1. El cliente realiza un pedido a través de la aplicación
2. El Microservicio de Pedidos recibe la solicitud
3. El Microservicio de Pedidos consulta al Microservicio de Inventario para validar disponibilidad
4. Si hay disponibilidad, se confirma el pedido y se notifica al cliente
5. El Microservicio de Inventario actualiza las existencias de ingredientes
6. La cocina recibe notificación del nuevo pedido a preparar

Integración entre Microservicios

Los microservicios se comunicarán utilizando un patrón de comunicación asíncrona a través de eventos de dominio:

- Evento: `PedidoCreado`
- Evento: `DisponibilidadValidada`
- Evento: `InventarioActualizado`

Tareas para los Estudiantes

Fase 1: Diseño del Dominio

1. Identificar y definir las entidades, objetos de valor y agregados para cada contexto
2. Diseñar los repositorios necesarios
3. Definir los servicios de dominio requeridos

Fase 2: Implementación del Microservicio de Pedidos

1. Implementar la API REST para la gestión de pedidos
2. Implementar la lógica de negocio para la creación y actualización de pedidos
3. Implementar la publicación de eventos de dominio

Fase 3: Implementación del Microservicio de Inventario

1. Implementar la API REST para consulta de inventario (Sync)
2. Implementar la lógica de validación de disponibilidad
3. Implementar la suscripción a eventos de dominio
4. Implementar la actualización de inventario (Async)

Fase 4: Pruebas y Validación

1. Implementar pruebas unitarias para cada microservicio
2. Implementar pruebas de integración
3. Validar el flujo completo del caso de uso

Estructura Sugerida para los Microservicios (Obligatorio usar Arquitectura Hexagonal)

Microservicio de Pedidos

```
pedidos-service/
├── src/
```

```
│   ├── main/
│   │   ├── java/
│   │   │   └── com/restaurante/pedidos/
│   │   │       ├── application/
│   │   │       │   ├── controllers/
│   │   │       │   └── services/
│   │   │       ├── domain/
│   │   │       │   ├── entities/
│   │   │       │   ├── repositories/
│   │   │       │   ├── services/
│   │   │       │   └── valueobjects/
│   │   │       ├── infrastructure/
│   │   │       │   ├── persistence/
│   │   │       │   └── messaging/
│   │   │       └── PedidosApplication.java
│   │   └── resources/
│   └── test/
└── pom.xml
```

Microservicio de Inventario

```
inventario-service/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/restaurante/inventario/
│   │   │       ├── application/
│   │   │       │   ├── controllers/
│   │   │       │   └── services/
│   │   │       ├── domain/
│   │   │       │   ├── entities/
│   │   │       │   ├── repositories/
│   │   │       │   ├── services/
│   │   │       │   └── valueobjects/
│   │   │       ├── infrastructure/
```

```
│   │   │       │   ├── persistence/
│   │   │       │   └── messaging/
│   │   │       └── InventarioApplication.java
│   │   └── resources/
│   └── test/
└── pom.xml
```

Tecnologías Sugeridas

- **Lenguaje**: De su preferencia (Java, Python, NodeJS, C#)
- **Framework**: De su preferencia (Springboot, Flask (Poetry), Express, Nest (Node), ASP .Net Core)
- **Base de Datos**: PostgreSQL o MongoDB (Base de Datos por cada Microservicio)
- **Mensajería**: Kafka o RabbitMQ o ActiveMQ
- **Testing**: Depende del lenguaje
- **Documentación API**: Swagger/OpenAPI

Entregables Esperados

1. Código fuente de ambos microservicios
2. README con instrucciones de ejecución
3. Documentación del diseño del dominio (2 diagramas UML)
4. Pruebas unitarias y de integración
5. Documentación de la API (Swagger)
6. Instrucciones para ejecutar el sistema (README)
7. Containers (Docker - Docker-compose)
8. Informe de implementación con decisiones de diseño y desafíos encontrados

Criterios de Evaluación

- Correcta implementación de los principios de DDD
- Adecuada separación de responsabilidades
- Calidad y cobertura de las pruebas (75%)
- Manejo adecuado de errores y excepciones
- Correcta implementación de la comunicación entre microservicios
- Claridad y organización del código
- Cumplimiento de los requisitos funcionales

Calendario de Entregas

1. 30 de Marzo - Microservicio de Pedidos (Sin Eventos)
2. 5 de Abril - Microservicio de Inventario (Sin Eventos)
3. 5 de Abril - Pruebas Unitarias
4. 12 de Abril - Eventos (Aqui implementan los eventos)
5. 27 de Abril - Docker e Informe final
6. 11 de Mayo - Despliegue en Azure o AWS