

# **Informe de implementación - Sistema de Gestión de Pedidos**

## **1. Introducción**

Este informe detalla el proceso de diseño e implementación del Sistema de Gestión de Pedidos para una cadena de restaurantes, que incluye los microservicios de Pedidos e Inventario. El sistema está desarrollado utilizando C#, PostgreSQL, Docker y emplea una arquitectura de microservicios con enfoque en arquitectura hexagonal.

## **2. Arquitectura implementada**

Se implementó una arquitectura hexagonal en ambos microservicios, asegurando una separación clara entre el dominio, la infraestructura y la aplicación. Se utilizó RabbitMQ para la comunicación asíncrona mediante eventos de dominio, lo que garantiza el desacoplamiento entre servicios.

## **3. Diseño del Dominio**

En el microservicio de pedidos, las entidades clave incluyen Pedido, Cliente e ItemPedido. En el microservicio de Inventario, se modelan las entidades Ingrediente y Receta. A continuación se presentan los diagramas de clases.

## **4. Pruebas**

Se implementaron pruebas unitarias e integración para ambos microservicios. Las pruebas unitarias validan la lógica interna del dominio, mientras que las pruebas de integración verifican la correcta interacción entre capas y servicios.

## **5. Manejo de errores**

Se implementó un manejo de errores estructurado utilizando controladores de excepciones centralizados y respuestas adecuadas de la API REST para mejorar la experiencia del usuario y facilitar la depuración.

## **6. Variables y configuración**

Las variables sensibles como credenciales de base de datos se gestionan mediante variables de entorno y archivos de configuración seguros. El archivo docker-compose permite levantar todos los servicios con sus dependencias.

## 7. Decisiones de diseño

- Uso de RabbitMQ para eventos de dominio asíncronos.
- Separación de lógica de dominio mediante arquitectura hexagonal.
- Empleo de Swagger para documentación de la API.
- PostgreSQL como sistema de base de datos relacional.
- Contenerización con Docker para facilitar el despliegue.

## 8. Diagramas e Imágenes del Proyecto

### Arquitectura Hexagonal

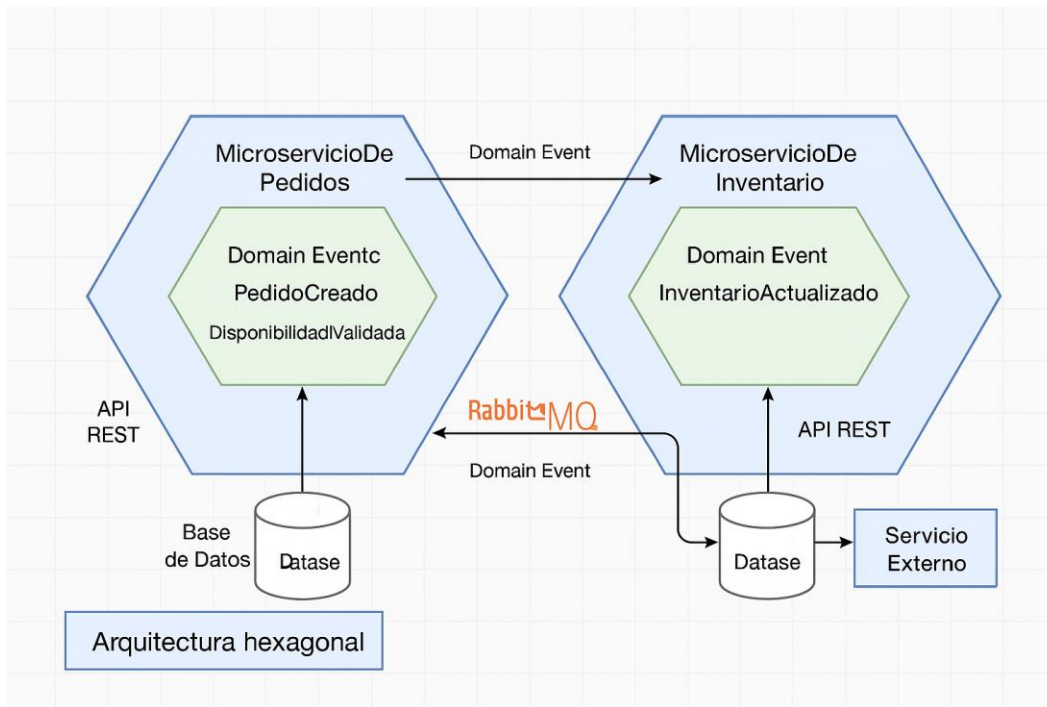


Figura: Arquitectura Hexagonal

```

classDiagram
    class ItemPedido {
        ProductId: Guid
        Cantidad: int
    }
    class Pedido {
        ID: Guid
        Cliente: Cliente
        Items: List<ItemPedido>
        DireccionEntrega: DireccionEntrega
        Confirmar()
        AgregaItem(Guid productId, int cantidad)
        CalcularTotal()
        Confirmar()
    }
    class Cliente {
        ID: Guid
        Nombre: string
    }
    class PedidoRepository {
        Guardar(Pedido pedido)
    }
    class PedidoService {
        CrearPedido(Guid productId, int cantidad)
    }
    class DireccionEntrega {
        Calle: string
        CodigoPostal: string
    }
    class PedidoTotal {
        Monto: decimal
    }
    class PedidoEventPublisher {
        PedidoCreado(Guid pedidoId)
        DisponibilidadValidada(Guid pedidoId)
    }
    class IPedidoEventPublisher {
        PedidoCreado(Guid pedido)
        DisponibilidadValidada(Guid pedidoId)
    }
    class PedidoController {
        CreatePedido(Guid clientId, Guid productId, int cantidad)
        UpdatePedido(Guid pedidoId)
    }
    class PedidoDbContext {
    }

    ItemPedido "1" -- "*" Pedido
    Pedido "1" *-- "*" ItemPedido
    Cliente "1" -- "1" Pedido
    PedidoRepository "1" -- "1" Pedido
    PedidoService "1" -- "1" Pedido
    DireccionEntrega "1" -- "1" Pedido
    PedidoTotal "1" -- "1" Pedido
    PedidoEventPublisher "1" -- "1" Pedido
    IPedidoEventPublisher "1" -- "1" Pedido
    PedidoController "1" -- "1" Pedido
    PedidoDbContext "1" -- "1" Pedido

```

## Diagrama de Clases - Microservicio de Inventario

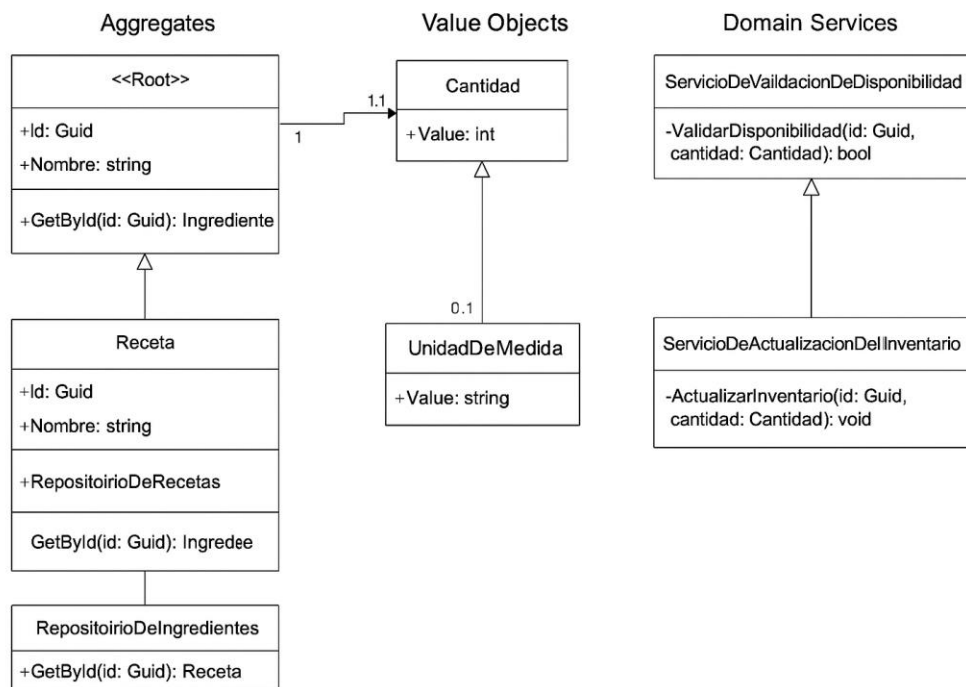


Figura: Diagrama de Clases - Microservicio de Inventario

## **9. Conclusión**

El sistema de gestión de pedidos diseñado es escalable, mantenible y desacoplado. Las decisiones de diseño tomadas permiten una evolución futura del sistema y una mejor adaptabilidad a distintos entornos tecnológicos.