TRABAJO PRACTICO - IMPLEMENTACIÓN DE CRUD CON PATRÓN CQRS

1. OBJETIVOS

Objetivo General

Implementar un sistema CRUD completo aplicando el patrón arquitectural CQRS (Command Query Responsibility Segregation) utilizando Spring Boot, demonstrando la separación de responsabilidades entre operaciones de lectura y escritura.

Objetivos Específicos

- Comprender y aplicar los principios fundamentales del patrón CQRS
- Diseñar una arquitectura que separe claramente las operaciones de comando y consulta
- Implementar servicios especializados para cada tipo de operación
- Desarrollar controladores REST que respeten la filosofía CQRS
- Aplicar buenas prácticas de validación y manejo de errores
- Documentar adecuadamente la arquitectura implementada

---

2. MARCO TEÓRICO

¿Qué es CQRS?

CQRS (Command Query Responsibility Segregation) es un patrón arquitectural que propone separar las operaciones que modifican el estado del sistema (Commands) de aquellas que únicamente consultan información (Queries). Este patrón se basa en el principio CQS (Command Query Separation) de Bertrand Meyer.

Ventajas del Patrón CQRS:

- **Escalabilidad independiente:** Los modelos de lectura y escritura pueden escalarse por separado
- **Optimización específica:** Cada lado puede optimizarse para su propósito específico
- **Complejidad reducida:** Separación clara de responsabilidades
- **Flexibilidad:** Posibilidad de usar diferentes tecnologías para cada lado

Componentes Principales:

- **Commands:** Representan intenciones de cambio en el sistema
- **Queries:** Representan solicitudes de información

- **Command Handlers:** Procesan los comandos y modifican el estado
- **Query Handlers:** Procesan las consultas y retornan información

---

3. DESCRIPCIÓN DEL PROBLEMA

Se requiere desarrollar un **Sistema de Gestión de Productos** para una empresa de retail que permita:

- Gestionar un catálogo de productos con sus respectivas categorías
- Realizar operaciones CRUD sobre productos y categorías
- Consultar productos por diferentes criterios (nombre, categoría, precio, stock)

Modelo de Dominio:

Entidad Producto:

- ID (Long): Identificador único
- Nombre (String): Nombre del producto (obligatorio, máx. 100 caracteres)
- Descripción (String): Descripción detallada (opcional, máx. 500 caracteres)
- Precio (Double): Precio unitario (obligatorio, mayor a 0)
- Stock (Integer): Cantidad disponible (obligatorio, no negativo)
- Categoría (Categoria): Referencia a la categoría (obligatorio)

Entidad Categoría:

- ID (Long): Identificador único
- Nombre (String): Nombre de la categoría (obligatorio, único)
- Descripción (String): Descripción de la categoría (opcional)

---

4. REQUERIMIENTOS TÉCNICOS

4.1 Tecnologías Obligatorias

- **Framework:** Spring Boot 3.x
- **Base de Datos:** MySQL
- **ORM:** Spring Data JPA
- **Validación:** Bean Validation (JSR-303)
- **Documentación:** Spring Boot Actuator + Swagger/OpenAPI

5. ESPECIFICACIONES FUNCIONALES

5.1 Comandos (Operaciones de Escritura)

Productos:

- **Crear Producto:** Validar datos, verificar existencia de categoría, evitar nombres duplicados
- **Actualizar Producto:** Modificar todos los campos excepto ID y fechas de auditoría
- **Eliminar Producto:** Eliminación lógica o física (a elección del estudiante)

Categorías:

- **Crear Categoría:** Validar unicidad del nombre
- **Actualizar Categoría:** Permitir modificación de nombre y descripción
- **Eliminar Categoría:** Solo si no tiene productos asociados

5.2 Consultas (Operaciones de Lectura)

Productos:

- Obtener todos los productos (con información de categoría)
- Buscar productos por nombre (búsqueda parcial, case-insensitive)
- Filtrar productos por categoría

Categorías:

- Obtener categorías con la cantidad de productos asociados

6. RECURSOS ADICIONALES

Bibliografia Recomendada:

- Evans, Eric. "Domain-Driven Design: Tackling Complexity in the Heart of Software"
- Vernon, Vaughn. "Implementing Domain-Driven Design"
- Fowler, Martin. "Patterns of Enterprise Application Architecture"

Enlaces Útiles:

- Spring Boot Documentation
- CQRS Pattern by Microsoft
- Martin Fowler on CQRS