Sistema de Gestión de Clientes – Backend con Spring Boot

Autor: Aura Cristina Garzon Rodriguez

Fecha: 23 Sep 2025

Sustentación de Proyecto

# 1. Introducción

El proyecto consiste en la creación de un backend REST para la gestión de clientes implementado con Java Spring Boot, aplicando buenas prácticas de desarrollo, principios de Clean Code, SOLID, manejo de logs, pruebas unitarias y arquitectura hexagonal.

# 2. Arquitectura del proyecto

Se implementa una arquitectura hexagonal, separando las capas de dominio, aplicación e infraestructura. Esto permite desacoplar la lógica de negocio de la infraestructura y facilita la mantenibilidad y pruebas.

Diagrama conceptual: Controller → Service → Repository → DB

# 3. Principios de Clean Code aplicados

- Nombres de clases y métodos descriptivos.  
- Código limpio, legible y consistente.  
- Comentarios y documentación Javadoc en todas las clases y métodos.  
- Métodos concisos y separación de responsabilidades.

# 4. Principios SOLID aplicados

- Single Responsibility: cada clase tiene una sola responsabilidad.  
- Open/Closed: clases abiertas a extensión pero cerradas a modificación.  
- Liskov Substitution: interfaces permiten reemplazar implementaciones sin romper el sistema.  
- Interface Segregation: interfaces pequeñas y específicas.  
- Dependency Inversion: la capa de aplicación depende de abstracciones.

# 5. Manejo de logs

Se utiliza SLF4J y LoggerFactory en todas las clases importantes (Controller, Service, Repository, Aplicación principal). Se registran diferentes niveles de logs:  
- INFO: operaciones normales exitosas.  
- WARN: condiciones inesperadas o validaciones.  
- ERROR: errores críticos o excepciones.

# 6. Implementación de Controller-Service-DAO-DTOs

- Controller: recibe solicitudes HTTP, valida parámetros y devuelve respuestas con códigos HTTP.  
- Service: lógica de negocio y delegación al Repository.  
- Repository/DAO: acceso a la base de datos mediante JDBC.  
- DTOs: estructuras de transferencia de datos entre capas.

# 7. Manejo de errores y códigos HTTP

- 200 OK → Cliente encontrado.  
- 400 Bad Request → Parámetros inválidos.  
- 404 Not Found → Cliente no encontrado.  
- 500 Internal Server Error → Errores inesperados.

# 8. Documentación de clases con Javadoc

ClienteController: logs de búsqueda y resultados, validaciones de parámetros, manejo de excepciones.  
ClienteService: operaciones de negocio con logs, delega al repository.  
ClienteRepositoryImpl: acceso a datos, mapeo de ResultSet a Cliente, logs y manejo de excepciones.  
GlobalExceptionHandler: manejo centralizado de errores y generación de respuestas HTTP.  
ClientesApp: inicialización de la aplicación, generación de clientes con CommandLineRunner y logs.

# 9. Conclusiones

El proyecto demuestra la aplicación de buenas prácticas de Clean Code y SOLID, una arquitectura desacoplada y modular, y un manejo de logs consistente. El sistema es mantenible, escalable y está listo para pruebas y documentación, cumpliendo todos los objetivos propuestos.