ARQUITECTURA HEXAGONAL - SERVICIUDAD CALI

Ports & Adapters Pattern - Domain-Driven Design Implementation

Diagrama de Arquitectura Hexagonal - ServiCiudad Cali | Universidad Autónoma de Occidente | Ingeniería de Software 2 | Octubre 2025
Equipo: Eduard Criollo, Felipe Charria, Jhonathan Chicaiza, Emmanuel Mena, Juan Sebastian Castillo

INFRASTRUCTURE - INPUT ADAPTERS

REST CONTROLLERS

```
1. DeudaRestController
/api/deuda (2 endpoints)

2. FacturaRestController
/api/facturas (6 endpoints)

3. ConsumoEnergiaRestController
/api/consumos-energia (3 endpoints)
```

HTTP Requests

HTTP CLIENTS

Web Browser / Mobile App
Postman / cURL
Third-party Apps

Invokes (via Port)

APPLICATION LAYER

USE CASES (Casos de Uso)

```
• ConsultarDeuda
UseCaseImpl
• GestionarFactura
UseCaseImpl
• ConsultarConsumo
EnergiaUseCaseImpl
```

DTOs (Data Transfer Objects)

```
Request:
• ConsultarDeudaRequest
• RegistrarPagoRequest
Response:
• DeudaConsolidadaResponse
• FacturaResponse
• ConsumoEnergiaResponse
• EstadisticasResponse
```

MAPPERS (Transformaciones)

```
• DeudaMapper
```

Uses

Maps to/from

DOMAIN LAYER (Núcleo Hexagonal - Business Logic)

VALUE OBJECTS (Objetos Inmutables)

```
• ClienteId
• FacturaId
• Periodo
• ConsumoAgua
• ConsumoEnergia
• Dinero
```

DOMAIN MODELS (POJOs Puros)

```
• FacturaAcueducto
• ConsumoEnergiaModel
• DeudaConsolidada
• EstadisticasDeuda
• EstadoFactura (Enum)
```

PORTS (INPUT) Use Cases Interfaces

```
• ConsultarDeudaUseCase
• GestionarFacturaUseCase
• ConsultarConsumo
EnergiaUseCase
```

PORTS (OUTPUT) Repository Interfaces

```
• FacturaRepositoryPort
• ConsumoEnergia
ReaderPort
```

Repository Port

Reader Port

INFRASTRUCTURE - OUTPUT ADAPTERS

JPA PERSISTENCE ADAPTER

```
FacturaRepositoryAdapter
├─ Implementa: FacturaRepositoryPort
├─ Usa: FacturaJpaRepository
├─ Usa: FacturaJpaMapper
└─ Entities: FacturaJpaEntity

ConsumoEnergiaReaderAdapter
├─ Implementa: ConsumoEnergiaReaderPort
├─ Usa: ConsumoEnergiaJpaRepository
├─ Usa: ConsumoEnergiaJpaMapper
└─ Entities: ConsumoEnergiaJpaEntity
```

JDBC Spring Data JPA

FILE READER ADAPTER (COBOL Legacy Integration)

```
AdaptadorArchivoEnergia
├─ Lee: consumos_energia.txt (78 chars)
├─ Parsea: Formato COBOL ancho fijo
└─ Convierte: String → ConsumoEnergiaModel
```

Files.read AllLines()

SPRING CONFIGURATION

```
HexagonalConfig
└─ @Bean creation for all Use Cases
(Dependency Injection orchestration)
```

EXTERNAL DATA SOURCES

PostgreSQL 15

Database: serviciudad_db
Table: facturas_acueducto
Rows: 20+ registros
Indices: 4 (B-tree)

consumos_energia.txt

Format: COBOL 78 chars
Fixed-width
Legacy Mainframe
Rows: 10+ registros

5 PATRONES DE DISEÑO IMPLEMENTADOS

1. ADAPTER PATTERN
ConsumoEnergiaReaderAdapter
→ Lee archivo COBOL legacy

2. REPOSITORY PATTERN
FacturaRepositoryPort + Adapter
→ Spring Data JPA abstraction

3. DTO PATTERN
Request/Response DTOs
→ Decoupling layers

4. BUILDER PATTERN
Response DTOs con @Builder
→ Immutable object construction

5. IoC/DI PATTERN
Spring @Autowired injection
→ Dependency Inversion Principle

11 ENDPOINTS REST

```
DeudaRestController:
• POST /api/deuda/consultar
• GET /api/deuda/cliente/{id}

FacturaRestController:
• GET /api/facturas/{id}
• GET /api/facturas/cliente/{id}
• GET /api/facturas/cliente/{id}/
periodo/{periodo}
• POST /api/facturas/pagar
• POST /api/facturas/{id}/anular
• POST /api/facturas/marcar-vencidas

ConsumoEnergiaRestController:
• GET /api/consumos-energia/
cliente/{id}
• GET /api/consumos-energia/
cliente/{id}/periodo/{periodo}
• GET /api/consumos-energia/
cliente/{id}/elevados
```

STACK TECNOLÓGICO

```
Backend:
• Spring Boot 3.2.0
• Java 17 LTS
• Maven 3.9.11

Persistence:
• PostgreSQL 15
• Hibernate 6.3 (JPA)
• HikariCP 5.0

Infrastructure:
• Docker + Compose
• OpenAPI 3.0 (SpringDoc)

Testing:
• JUnit 5 + Mockito
• AssertJ 3.24

Observability:
• Spring Actuator
• Prometheus + Grafana
```

MÉTRICAS DE CALIDAD

```
Cobertura de Tests:
• Lines: 85%
• Branches: 78%

Complejidad:
• Cyclomatic avg: 6.2
• Max complexity: 12

Performance:
• P95 response: 1.2s
• Throughput: 120 RPS
• Error rate: 0.3%

Calidad de Código:
• Critical bugs: 0
• Major bugs: 0
• Code smells: 12 (minor)

Vulnerabilities:
• Critical CVEs: 0
• Dependency issues: 0
```