Documentación de Software

Carátula

Título: RetoBackend V6

Integrante: Sergio Andre Gomez Vallejos

Mes y año: Agosto 2025

---

1. Introducción

Propósito del Caso

Se solicita el desarrollo de una solución de software orientada a microservicios que permita la gestión y consulta de información de clientes y sus productos financieros asociados. La arquitectura debe implementar un Backend for Frontend (BFF) que integre microservicios especializados para proporcionar una API unificada y segura.

Nombre del Producto de Software

Financia - Sistema de Gestión de Información Financiera de Clientes

---

2. Requisitos Funcionales del Sistema

RF-001: Consulta de Información de Cliente

- **Descripción:** El sistema debe proporcionar una API que permita obtener la información completa de un cliente mediante su código único.
- **Entrada:** Código único del cliente (encriptado)
- **Salida:** Datos del cliente (nombres, apellidos, tipo de documento, número de documento) y lista completa de productos financieros asociados.

RF-002: Gestión de Productos Financieros

- **Descripción:** El sistema debe devolver todos los productos financieros asociados a un cliente específico.
- **Datos incluidos:** Tipo de producto, nombre del producto y saldo actual.
- **Tipos de productos:** Cuentas de ahorro, tarjetas de crédito, entre otros productos financieros.

RF-003: Encriptación de Parámetros

- **Descripción:** El código único enviado como parámetro de entrada debe estar encriptado para garantizar la seguridad de la información.
- **Implementación:** Mecanismo de encriptación/desencriptación para el parámetro codigoUnico.

RF-004: Autenticación de API

- **Descripción:** El acceso a la API debe estar protegido mediante un mecanismo de autenticación robusto.
- **Implementación:** Sistema de autenticación OAuth para controlar el acceso a los servicios.

RF-005: Microservicio de Clientes

- **Descripción:** Implementar un microservicio especializado que contenga y gestione toda la información relacionada con los clientes.
- **Responsabilidad:** Almacenamiento, consulta y gestión de datos de clientes.

RF-006: Microservicio de Productos Financieros

- **Descripción:** Implementar un microservicio especializado que contenga y gestione la información de productos financieros.
- **Responsabilidad:** Almacenamiento, consulta y gestión de productos financieros y sus saldos.

RF-007: Backend for Frontend (BFF)

- **Descripción:** Crear un BFF que actúe como punto de integración entre los microservicios de clientes y productos financieros.
- **Función:** Orquestar las llamadas a los microservicios y presentar una API unificada.

RF-008: Sistema de Tracking

- **Descripción:** Implementar un identificador único para realizar el seguimiento de las transacciones desde el BFF hasta el último microservicio.
- **Propósito:** Trazabilidad completa de las operaciones para auditoría y debugging.

---

3. Requisitos No Funcionales del Sistema

RNF-001: Tecnología de Desarrollo

- **Lenguaje:** Java 17 con implementación de características modernas
- **Características requeridas:** Functional Interface, Predicate, Stream, Optional, Default, Lambda, Date Time
- **Framework principal:** Spring (Security, Boot, Data)
- **Framework reactivo:** WebFlux para programación reactiva

RNF-002: Arquitectura y Patrones

- **Patrones de diseño:** Implementación de patrones de diseño reconocidos
- **Principios SOLID:** Adherencia estricta a los principios SOLID de programación orientada a objetos
- **AOP:** Implementación de Aspect-Oriented Programming para cross-cutting concerns

RNF-003: Testing y Calidad

- **Framework de testing:** JUnit 5 para pruebas unitarias e integración

- **Cobertura:** Pruebas comprehensivas de la funcionalidad implementada

RNF-004: Seguridad

- **Autenticación:** OAuth para la capa de seguridad
- **Encriptación:** Mecanismos seguros de encriptación para parámetros sensibles
- **Autorización:** Control de acceso basado en roles y permisos

RNF-005: Persistencia de Datos

- **Base de datos:** Flexibilidad en la elección de la base de datos según preferencias del desarrollador
- **Gestión de datos:** Implementación eficiente de operaciones CRUD

RNF-006: Containerización

- **Docker:** Dockerización completa de microservicios y BFF
- **Imágenes:** Libertad en la elección de imágenes base para contenedores
- **Orquestación:** Preparación para despliegue en entornos containerizados

RNF-007: Logging y Monitoreo

- **Configuración:** Implementación y configuración de Logback para logging estructurado
- **Trazabilidad:** Logs detallados para seguimiento de operaciones

RNF-008: Utilidades y Productividad

- **Mapeo de objetos:** Implementación con MapStruct para mapeo eficiente
- **Reducción de boilerplate:** Uso de Lombok para código más limpio
- **Serialización:** Implementación con Gson o Jackson para manejo de JSON

RNF-009: Documentación

- **API Documentation:** Documentación completa usando OpenAPI (Swagger)
- **README:** Documentación técnica detallada en archivo README.md
- **Entrega:** Código fuente empaquetado en archivo ZIP o RAR

RNF-010: Starter Personalizado

- **Spring Starter:** Creación de un starter básico personalizado en Spring
- **Configuración:** Facilitar la configuración y uso del sistema

RNF-011: Demostración

- **Postman:** Demostración del uso de la API mediante colección de Postman
- **Testing:** Casos de prueba documentados y ejecutables

RNF-012: Flexibilidad de Nomenclatura

- **Estándares:** No hay restricciones específicas en los nombres de objetos creados
- **Consistencia:** Mantener consistencia interna en la nomenclatura elegida