# Documento de Arquitectura Técnica - Módulo Backend Core

## 1. Propósito del Módulo

El módulo `backend` constituye el núcleo funcional y arquitectónico del sistema, sirviendo como base para los módulos específicos de negocio (por ejemplo, usuarios, pedidos, pagos, etc.). Está construido con Kotlin utilizando el framework Ktor, y soporta despliegue tanto embebido como serverless.

## 2. Tecnologías y Frameworks

- Ktor: Framework principal para la construcción del servidor HTTP y APIs.

- Kodein DI: Sistema de inyección de dependencias utilizado para registrar e inyectar funciones de negocio.

- Gson: Serialización/deserialización de objetos JSON.

- AWS Cognito: Utilizado para validación de tokens JWT.

- AWS Lambda: Permite ejecutar el backend como funciones serverless.

- RateLimiter (TokenBucket): Límite de tasa de peticiones por usuario.

## 3. Componentes Principales

- Application.kt:

- Inicializa el servidor embebido (Netty).

- Instala RateLimiting para prevenir abuso.

- Configura la inyección de dependencias con Kodein.

- Expone la ruta POST /business/function que despacha llamadas a funciones registradas en el contenedor DI.

- Ruteo Dinámico de Funciones:

- Las funciones se registran con un tag dentro de Kodein.

- Se determina la función desde la URL y se instancia mediante DI.

- Function & SecuredFunction:

- Function es una interfaz genérica.

- SecuredFunction extiende Function y valida tokens JWT.

- LambdaRequestHandler:

- Permite reutilizar la lógica de funciones en AWS Lambda.

- Request y Response:

- Clases genéricas base para entrada y salida de datos.

- Excepciones Personalizadas:

- RequestValidationException, UnauthorizedException, ExceptionResponse, etc.

## 4. Ejecución en Ambientes

- Modo Local / Embebido:

- Utiliza embeddedServer(Netty).

- Modo AWS Lambda:

- Ejecuta LambdaRequestHandler con DI.

- Decodifica el body Base64 y expone cabeceras para CORS.

## 5. Patrón de Modularización

Este backend actúa como una superclase funcional. Los módulos de negocio importan este backend y registran sus propias funciones bajo nombres específicos, permitiendo:

- Reutilización de lógica común.

- Separación de responsabilidades.

- Escalabilidad horizontal por servicio.

## 6. Diagrama de interacción de componentes

La siguiente imagen ilustra el flujo de interacción entre los componentes clave del backend:

