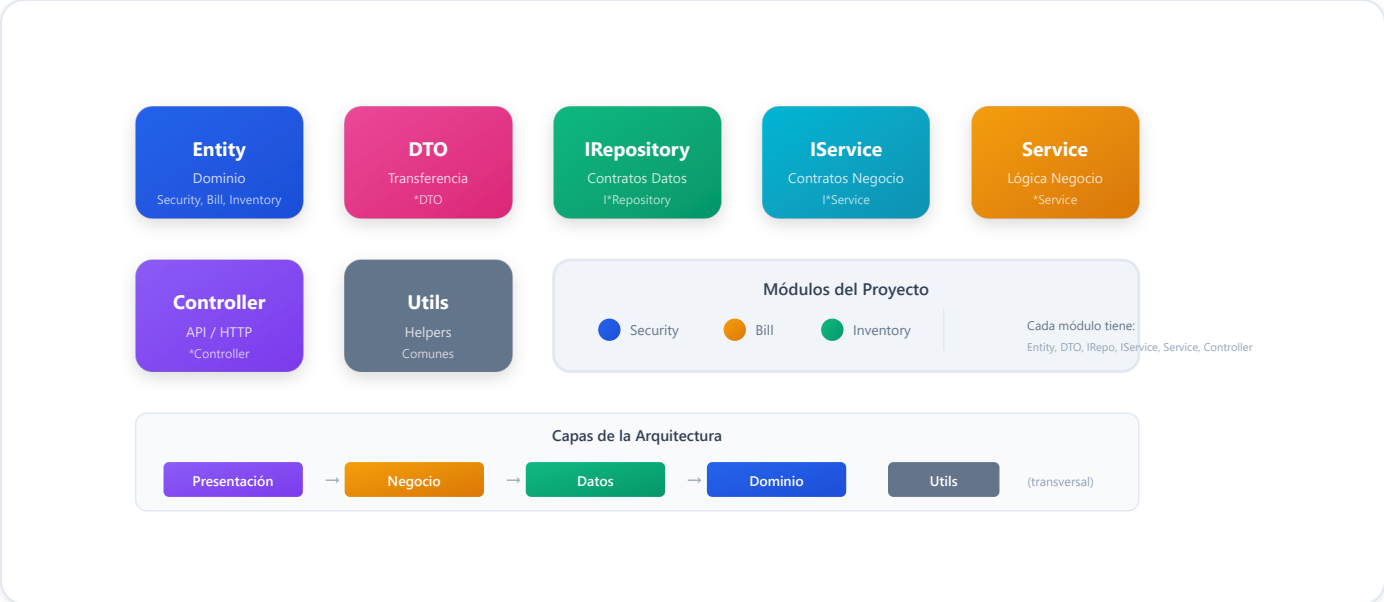
POC IA

Arquitectura del Proyecto aplicando SOLID

Implementación de arquitectura en capas con separación de responsabilidades, bajo acoplamiento y alta mantenibilidad.

Estructura General del Proyecto

Organización modular siguiendo principios de diseño limpio



1 Entity - Modelo de Dominio

Representación del núcleo del negocio

Clases

Security Bill Inventory

Responsabilidades

- → Representar entidades del negocio
- → Definir propiedades del dominio
- → Reglas simples del dominio (si aplica)
- → Cero dependencias de infraestructura

⚠️ **Regla clave:** No invocar repositorios/servicios desde Entity. No mezclar lógica de presentación ni persistencia.

2 DTO - Data Transfer Objects

Objetos de transferencia entre capas

Responsabilidades

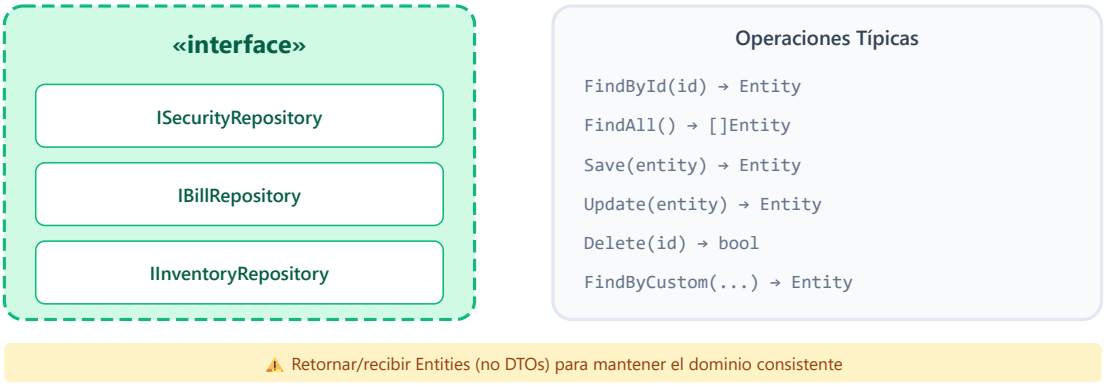
Buenas Prácticas

- → Transportar datos entre capas
- → Controlar exposición de datos
- → Facilitar validaciones de entrada/salida
- → Definir formatos de comunicación

- → Mantener DTOs simples (solo datos)
- → Separar DTOs Request vs Response si el proyecto crece
- → No incluir lógica de negocio

3 IRepository - Contratos de Acceso a Datos

Interfaces que definen operaciones de persistencia



4 IService - Contratos de Lógica de Negocio

Interfaces que definen casos de uso del sistema

ISecurityService

- → Autenticación
- → Autorización
- → Gestión de usuarios

IBillService

- → Creación de facturas
- → Cálculo de totales
- → Validaciones fiscales

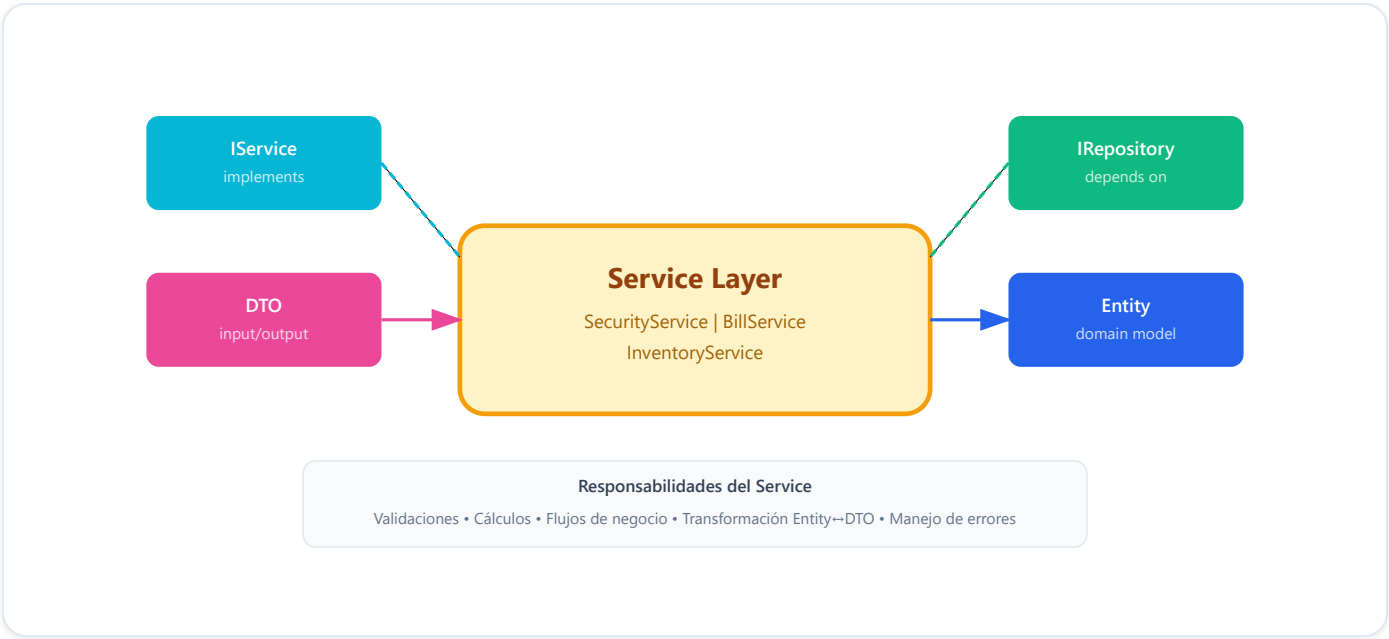
IInventoryService

- → Control de stock
- → Movimientos
- → Alertas de inventario

> **Propósito:** Desacoplar Controllers de implementación concreta. Facilitar pruebas unitarias mediante mocks.

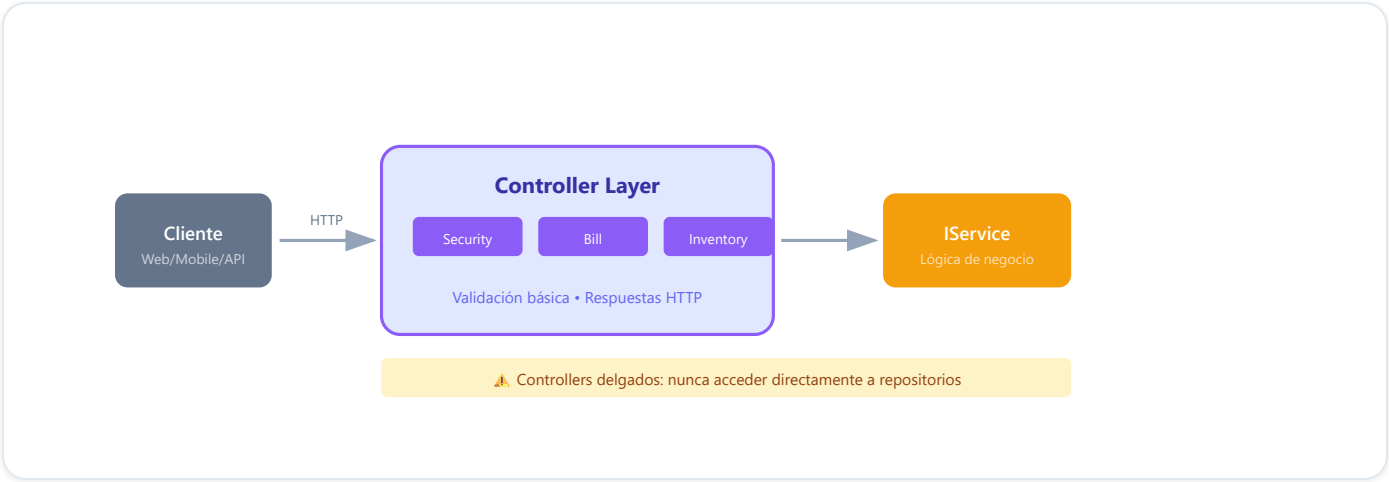
5 Service - Implementación de Lógica de Negocio

Donde vive la lógica "importante" del sistema



6 Controller - Capa de Presentación / API

Endpoints HTTP y orquestación de peticiones



7 Utils - Utilidades Comunes

Helpers y funciones transversales

Contenido Típico

- → Helpers (fechas, strings, formatos)
- → Constantes y configuraciones
- → Validadores reutilizables
- → Mapeos sencillos

⛔ Evitar

- → Lógica de negocio principal
- → Convertirlo en "cajón de sastre"
- → Dependencias circulares

Flujo Típico de una Solicitud

Recorrido completo de una petición HTTP

Principios SOLID Aplicados

Cómo cada principio se manifiesta en la arquitectura

S

Single Responsibility

Cada capa hace una cosa: HTTP, negocio, datos

O

Open/Closed

Extensible sin modificar código existente

L

Liskov Substitution

Implementaciones intercambiables

I

Interface Segregation

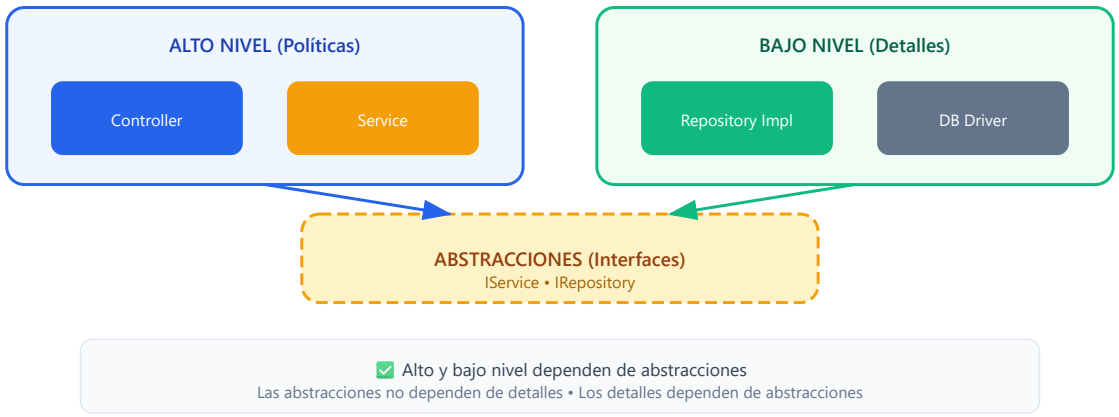
Interfaces separadas por entidad

D

Dependency Inversion

Depender de abstracciones, no concreciones

Inversión de Dependencias en Acción



POC IA - Arquitectura en capas aplicando principios SOLID

Separación de responsabilidades • Bajo acoplamiento • Alta mantenibilidad