Lógica de Negocio

2013

@juanma_ari

Agenda

- Introducción (10 minutos)
- Diseño de la Lógica de Negocio (30 minutos)
- Arquitectura de la Lógica de Negocio (45 minutos)
- Tipos de Arquitectura (25 minutos)

Agenda

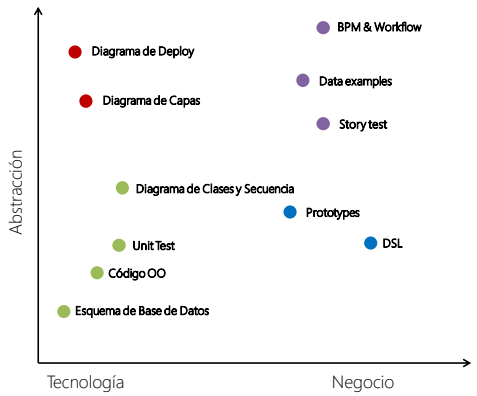
- Introducción (10 minutos)
- Diseño de la Lógica de Negocio (30 minutos)
- Arquitectura de la Lógica de Negocio (45 minutos)
- Tipos de Arquitectura (25 minutos)

Introducción

¿Qué es un Modelo?

Visión simplificada de algo complejo utilizada en el análisis y resolución de problemas

Aspectos de un Modelo



Agenda

- Introducción
- Diseño de la Lógica de Negocio
- Arquitectura de la Lógica de Negocio
- Tipos de Arquitectura

Claves de Diseño

SOLID	DRY	Common Closure	Common Reuse
KISS	YAGNI	Stable Abstraction	Separation of Concerns

SOLID

- Single Responsability Principle
- Open Close Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

DRY

DON'T REPEAT YOURSELF

Evitar duplicaciones

Aplicar abstracciones

KISS

KEEP IT SIMPLE…

Evitar complejizar el problema de forma innecesaria

Un modelo simple es siempre más fácil de mantener

YAGNI

YOU AIN'T GONNA NEED IT

No añadir funcionalidad extra que no vamos a utilizar

Desventajas de implementar algo "a futuro"

- Más tiempo de Testing
- Más tiempo de documentación
- Añadir funcionalidad extra, puede requerir añadir más funcionalidad extra!

Common Closure

Las clases que se usan juntas, se empaquetan juntas

Apunta a la Modificabilidad

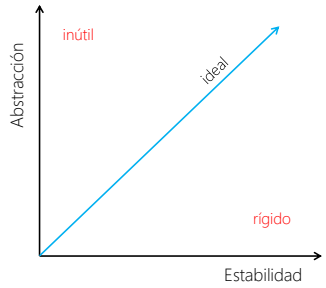
Permite facilidad de distribución y actualización

Common Reuse

Las clases que se usan juntas, se empaquetan juntas

Stable Abstractions

Tener un balance entre lo abstracto y lo rígido



Separation of Concerns

Aplica a paquetes, clases y métodos

Separa las responsabilidades:

- En un nuevo método
- En una nueva dependencia

Mecanismos de Separación

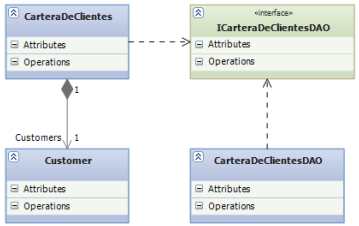
Programática

Declarativa

Introspectiva

Programática

Separación de responsabilidades mediante objetos que se mandan mensajes entre sí



Declarativa

Separa la lógica de negocio de aspectos intrusivos.

Parametriza fuera del código los componentes arquitecturales.

Me concentro en el qué y no en el cómo.

```
<Button  Width="102"
         Height="31"
         Click="OnClick" />
```

```
[Required]
public void Email(string email)
{
     this.Email = email;
}
```

Introspectiva

Puedo manipular la lógica utilizando Reflection

Modifica el comportamiento de mi aplicación

Me permite contar con puntos de extensibilidad

Agenda

- Introducción
- Diseño de la Lógica de Negocio
- Arquitectura de la Lógica de Negocio
- Tipos de Arquitectura

Arquitecturas

DDD	Capas	Aspectos	Transaction Script
Ports and Adapters	CQRS	Workflow	Business Rules

Domain Driven Design

Problemas al construir Software:

- Construir Software complejo sin conocer el Dominio
- Trabajar en conjunto con el experto del negocio

Domain Driven Design

Propone una serie de métodos para construir software trabajando con el experto de domino

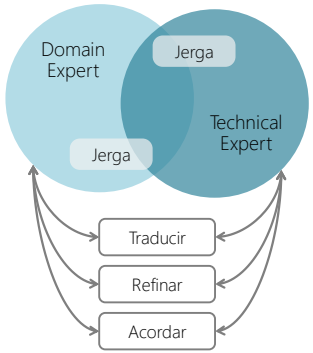
Es el dominio quien guía la solución

Principales conceptos:

- Ubiquitous Language
- Bounded Context
- Aggregates Roots

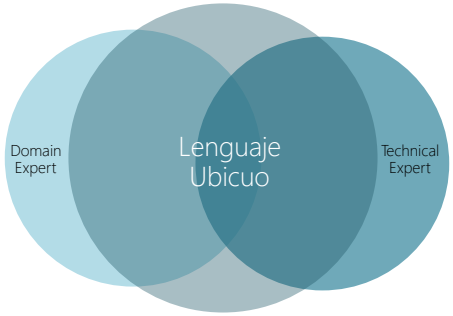
Domain Driven Design

LENGUAJE



Domain Driven Design

LENGUAJE UBICUO



Domain Driven Design

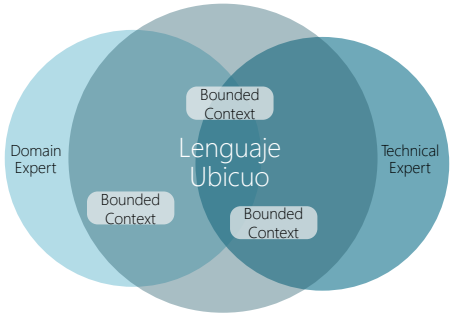
BOUNDED CONTEXT

Cada contexto está optimizado para resolver un problema específico

Permite tener sistemas distribuidos en lugar de un gran sistema monolítico

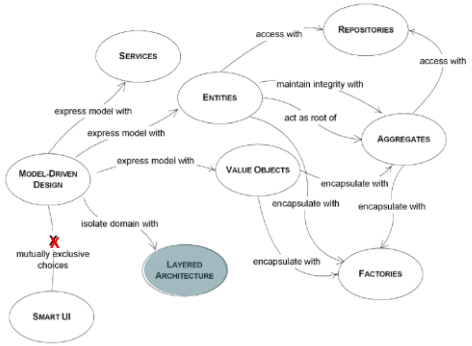
Domain Driven Design

BOUNDED CONTEXT



Domain Driven Design

BUILDING BLOCKS



Domain Driven Design

BUILDING BLOCKS

Entidades:
- Identidad
- Mutables

Value Object
- No tienen identidad
- Inmutables

Aggregates:
- Compuesto por objetos
- Soportan operaciones de negocio
- Aggregate Root

Capas

Conjunto de componentes reutilizables

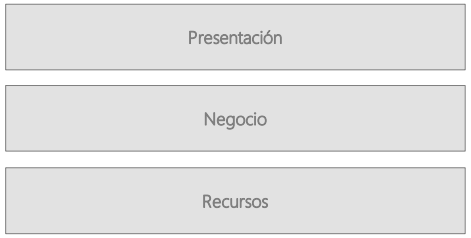
Ayuda a aplicar el principio de SoC

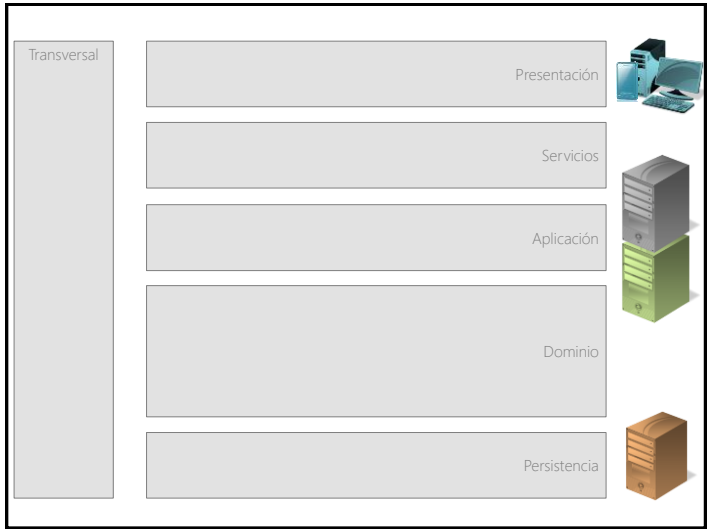
Facilidad para identificar problemas

Elimina duplicación innecesaria

Separación Horizontal

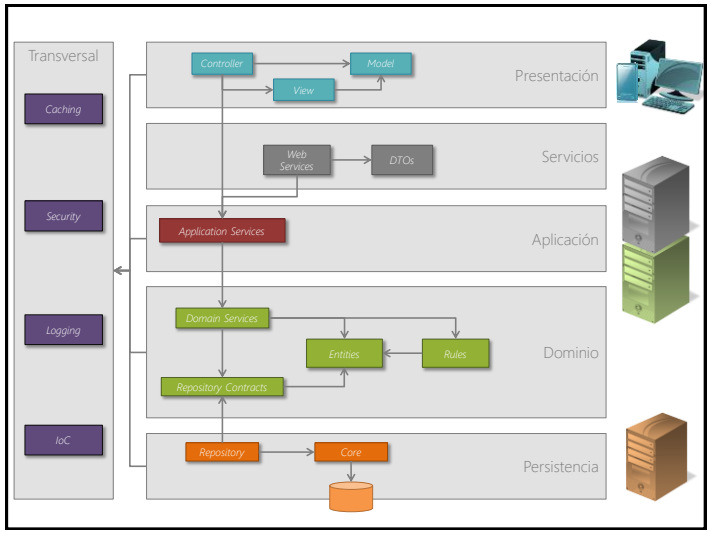
División lógica por funcionalidad

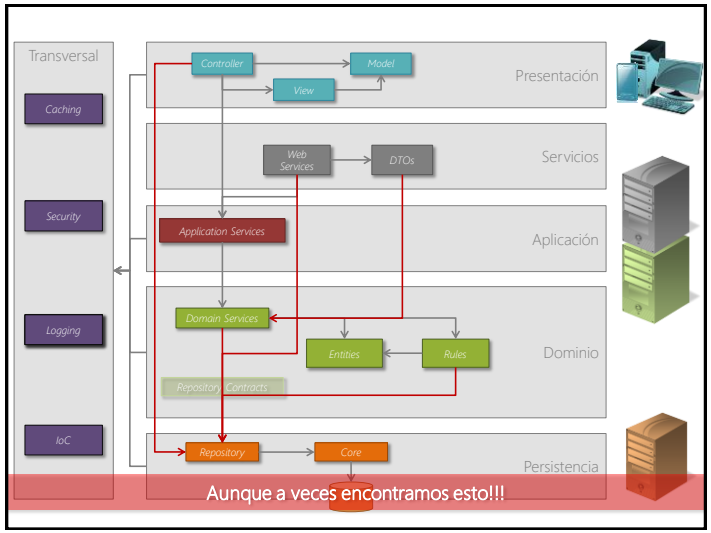





Transversal
- Aspectos Horizontales
- Impactan en toda la App.
- Favorece la reutilización
- DI / AOP
- Interfaz de Usuario
- MVC / MVP / MVVM
Presentación
- Capa de Servicios Distribuidos
- Fachada de nuestra lógica
- REST / SOAP
Servicios
- Coordina actividades de la Aplicación
- No incluye lógica de Negocio
- Coordina servicios de la capa de nivel inferior
Aplicación
- Implementa la funcionalidad principal de nuestro Sistema
- Es quien cuenta con las Entidades de nuestro Negocio
- Recordar que las operaciones nacen del modelo Ubicuo
- Totalmente aislado de los componentes de Infraestructura
Dominio
- Centraliza el acceso a los datos
- Desacopla la tecnología utilizada
- DAOs / Repositorios / ORM / DataMapper / ActiveRecord
Persistencia


Transversal
- Aspectos Horizontales
- Impactan en toda la App.
- Favorece la reutilización
- DI / AOP
- Interfaz de Usuario
- MVC / MVP / MVVM
Presentación
- Capa de Servicios Distribuidos
- Fachada de nuestra lógica
- REST / SOAP
Servicios
- Coordina actividades de la Aplicación
- No incluye lógica de Negocio
- Coordina servicios de la capa de nivel inferior
Aplicación
- Implementa la funcionalidad principal de nuestro Sistema
- Es quien cuenta con las Entidades de nuestro Negocio
- Recordar que las operaciones nacen del modelo Ubicuo
- Totalmente aislado de los componentes de Infraestructura
Dominio
- Centraliza el acceso a los datos
- Desacopla la tecnología utilizada
- DAOs / Repositorios / ORM / DataMapper / ActiveRecord
Persistencia

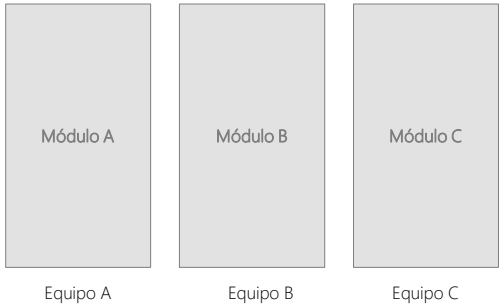

Transversal
Caching
Security
Logging
IoC
Controller
Model
View
Presentación
Web Services
DTOs
Servicios
Application Services
Aplicación
Domain Services
Entities
Rules
Repository Contracts
Dominio
Repository
Core
Persistencia


Transversal
Caching
Security
Logging
IoC
Controller
Model
View
Presentación
Web Services
DTOs
Servicios
Application Services
Aplicación
Domain Services
Entities
Rules
Repository Contracts
Dominio
Repository
Core
Persistencia
Aunque a veces encontramos esto!!!

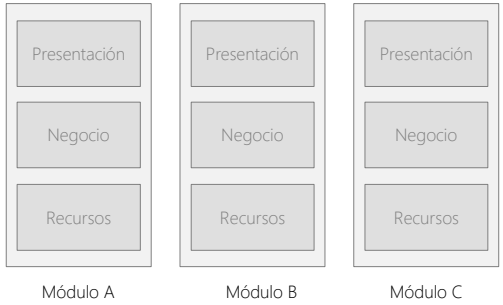
Separación Vertical

División lógica por módulos

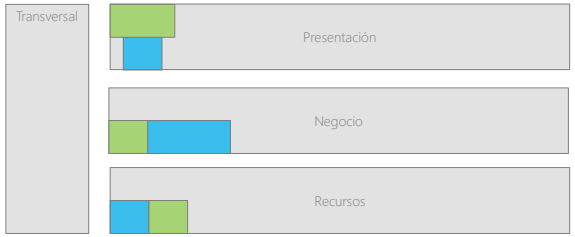
Separa Responsabilidades y Dependencias



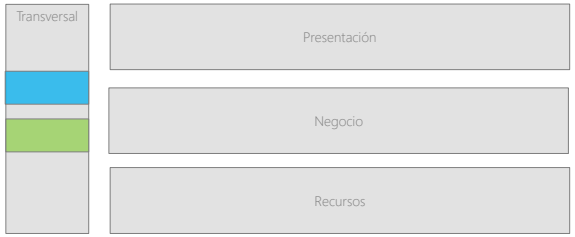
Horizontal - Vertical



Separación por Aspectos



Separación por Aspectos



Separación por Aspectos

Aspectos Típicos

- Seguridad
- Cache
- Gestión de Configuraciones
- Gestión de Excepciones
- Logging

Transaction Script

Organiza la lógica de negocio en procedimientos

Cada procedimiento maneja una petición de la presentación

Transaction Script

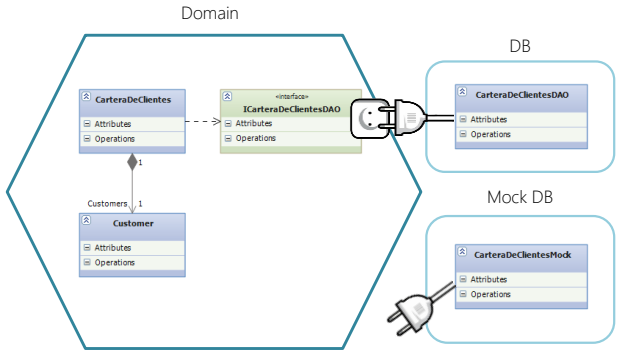
Ports and Adapters

El "core" es el modelo y es centro de la aplicación

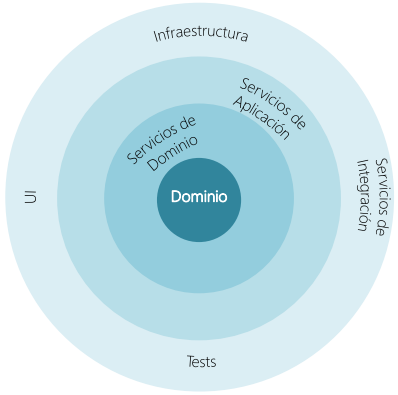
La infraestructura depende del core

La UI depende del core y tiene acceso a la infraestructura

Ports and Adapters



Ports and Adapters



CQS

COMMAND QUERY SEPARATION

Es un principio que indica:

"Un método cambia el estado de un objeto o retorna un valor, pero no ambos"

CQS

COMMAND

- Cambian el estado
- Retornan void

QUERIES

- Devuelven un resultado
- Retornan un tipo

CQRS

COMMAND QUERY RESPONSIBILITY SEGREGATION

Existe para resolver un problema particular:

"Bloquear al usuario cuando se bloquean los datos"

CQRS

COMMAND QUERY RESPONSIBILITY SEGREGATION

Utilizado solamente en dominios colaborativos donde hay gran cantidad de personas modificando un conjunto pequeño de datos

Bloquear los datos es necesario, pero bloquear al usuario no

CQRS

COMMAND QUERY RESPONSIBILITY SEGREGATION

¿Qué pasa si tenemos pocos productores pero muchos consumidores?

¿Por que complejizar y comprometer performance por transformaciones sin sentido?

CQRS

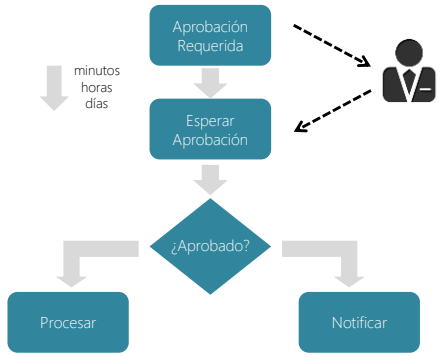
Permite escalar por separado el modelo de Lectura y Escritura

Aplicable a un Bounded Context

UI con respuestas rápidas

Workflow

Escenarios "Reactivos"



Workflow

ACTIVIDADES

Representan acciones o tareas:

- Leer de una Base de Datos
- Generar archivos
- Hacer FTP
- Llamar a un Web Service
- Etc.

Business Rules

MOTOR DE REGLAS

Permite definir las reglas de nuestro negocio de forma centralizada

Puedo cambiar el comportamiento del sistema sin necesidad de modificarlo

Business Rules

UNA REGLA (EN LENGUAJE TÉCNICO)

Ej: "Cuando un cliente es del tipo Gold, entonces se le aplica un descuento del 10%"

```
when Customer( Type == "Gold")
then discount = new Discount(10);
```

Business Rules

UNA REGLA (IMPLEMENTADA COMO DSL)

Define en lenguaje natural el comportamiento de la regla

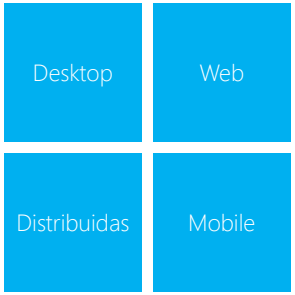
Fácil de comunicar con experto del negocio

Cuando un cliente es Gold
Entonces tiene un descuento del 15%

Agenda

- Introducción
- Diseño de la Lógica de Negocio
- Arquitectura de la Lógica de Negocio
- Tipos de Arquitectura

Aplicación de Arquitectura



Desktop

Abundan los recursos

Aplicaciones pesadas

Atadas al Sistema Operativo

Web

Recursos escasos

Multiplataforma

Basado en estándares

Interfaces fluidas

Comunicaciones asincrónica

Distribuidas

Múltiples servidores

Tecnologías que permitan la distribución

Transparencia en su uso

Cluster

Grid Computing

Mobile

Interfaces Touch

Tiempo de Respuesta muy rápidos

Guidelines de diseño

Recursos más limitados