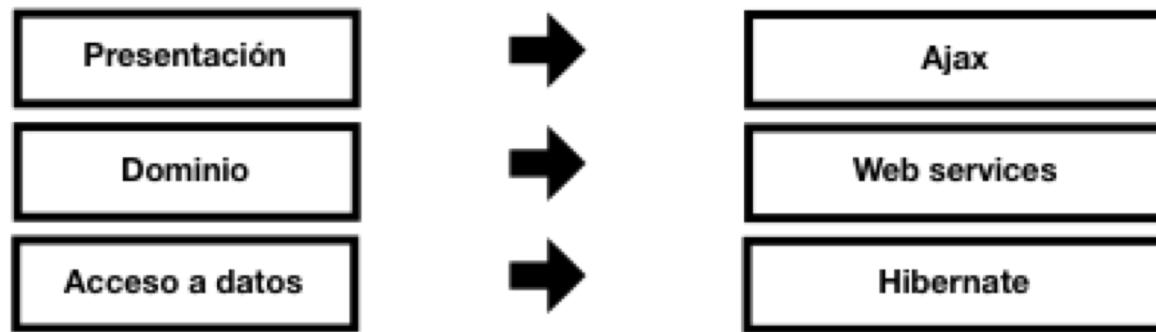


Desarrollo de Aplicaciones Web

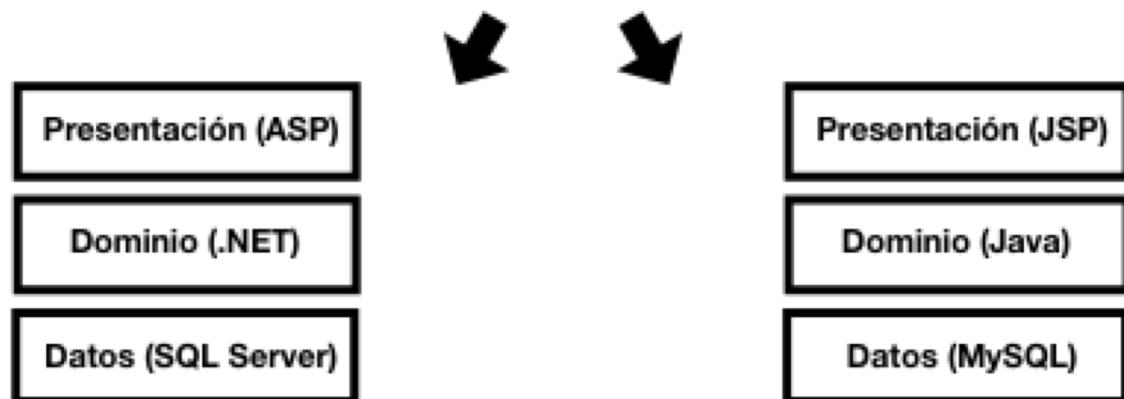
Tema I: Capa de Dominio

Recordemos: Arquitectura en 3 Capas

Se pueden utilizar diferentes tecnologías



El framework depende del lenguaje



Temas a seguir

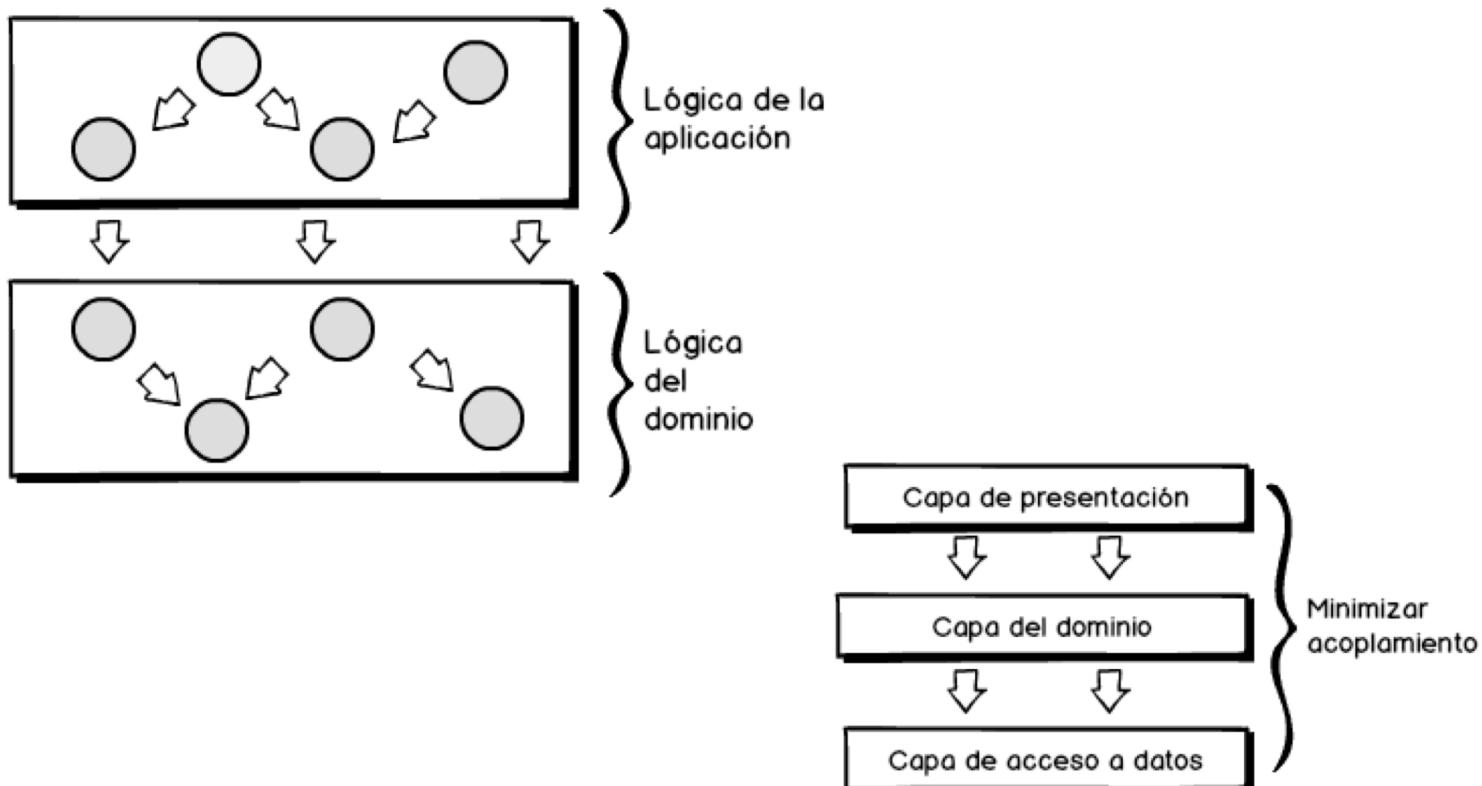
- Capa de Dominio
- Enfoque de organización de la CD
 - Rutina de transacción
 - Módulo de tabla
 - Modelo de dominio
 - Capa de servicio

Capa de Dominio

- El software con cierto nivel de complejidad
 - Está organizado en capas
 - Cada capa representa una sección lógica de el
- Los módulos en la capa de dominio tienen
 - Los algoritmos funcionales
 - Los cálculos que hacen que el sistema trabaje
 - La interacción con otras capas como son
 - La Capa de Datos
 - La Capa de Presentación

Capa de Dominio

Capa del dominio



Capa de Dominio (Capa 2)

- La Capa de dominio
 - Es el nervio central del sistema
 - Es el área principal para la lógica del sistema
- Existen algunos enfoques para su organización
 - Rutina de transacción
 - Módulo de tabla
 - Modelo de dominio
 - Capa de servicio

Rutina de Transacción

Rutina de transacción

- Estas rutinas de transacción
 - Reciben los parámetros de entrada
 - Procesa la información
 - Almacena la info en la base de datos
 - Invoca operaciones de otros sistemas
 - Responde con datos
- Todo lo anterior dentro de un único procedimiento

Rutina de transacción (Cont)

- Puede ser tan simple como
 - Desplegar Información
 - Hacer validaciones
 - Generar resultados de cálculos
- La idea es organizar la lógica del negocio
 - En procedimientos
 - Cada uno maneja una solicitud de la Capa 1 (?)

Rutina de transacción (Cont)

- En detalle cada transacción
 - Recibe la solicitud
 - Sigue la info a la base de datos (Capa 3)
 - Modifica los datos
 - Guarda los resultados en la BD
 - Las rutinas no deben invocar la Capa 1

Rutina de transacción (Cont)

- Transforman los componentes del negocio
 - En acciones requeridas por el usuario
- Por cada solicitud del usuario
 - Se debe escribir un método o rutina
- La iniciativa siempre proviene del usuario

Rutina de transacción (Cont)

- Ventajas
 - No necesita ser monolítico
 - Una transacción de alto nivel
 - Se puede separar en varias rutinas más simples
 - Estas rutinas pueden ser reutilizables
 - Como ejemplo está el proceso de persistencia de datos
 - Estos componentes manejan el acceso a los datos
 - Los cuales pueden ser reutilizables

Rutina de transacción (Cont)

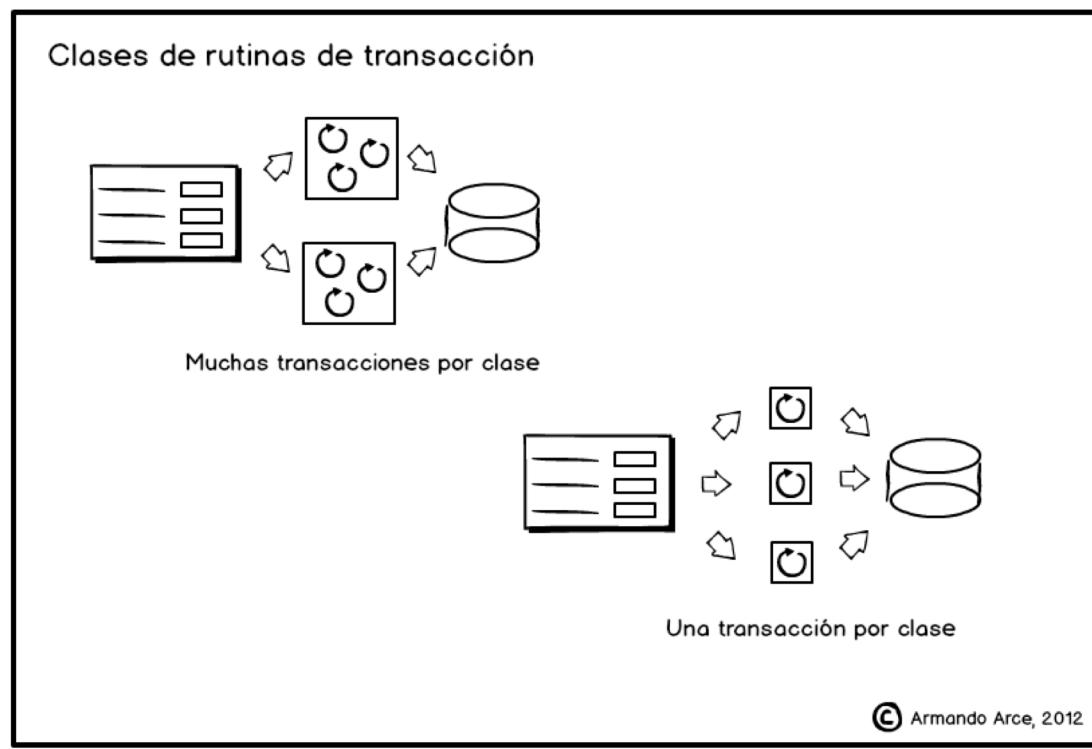
- Ventajas
 - Es un modelo procedimental y muchos entiende
 - Fácil de programar
 - Es fácil de comprender
 - No hay que preocuparse por otras transacciones
 - Varios programadores pueden trabajar a la vez
 - Cada transacción es independiente
 - Son ideales para proyectos con
 - Poca lógica
 - Tiempos de entrega cortos
 - Que utilicen ambientes de desarrollo IDEs

Rutina de transacción (Cont)

- Desventajas
 - Los problemas aparecen cuando se vuelve complejo
 - Las rutinas tienden a duplicar el código
 - Es normal ver transacciones que
 - Terminan ejecutando tareas muy similares
 - Este problema se puede corregir con refactoring
 - Extrae pedazos (chunks)
 - Convierte los chunks en métodos en sí
 - La remodelación tiene limitaciones y no sirve para todo

Rutina de transacción (Cont)

- Las rutinas se pueden estructurar en
 - Muchas transacciones en una clase
 - Una transacción por clase



Rutina de transacción (Cont)

- Muchas transacciones en una clase
 - Es el método más común de organización
 - Contiene múltiples transacciones en 1 clase
 - Cada clase define una temática que las asocia
 - Es adecuado en ciertas situaciones
 - Facilita entender y mantener el código agrupado
 - Una rutina de alto nivel puede ser
 - Separada en subrutinas
 - Estas se pueden reutilizar al ser más simples

Rutina de transacción (Cont)

- Una clase por transacción
 - Utiliza un patrón de diseño llamado comando
 - Se define un supertipo para los comandos
 - Especifican algún método que ejecuta la lógica
 - Permite manipular instancias como objetos
 - En tiempo de ejecución
 - Es posible administrar varias transacciones a la vez
 - Genera problemas al pasar parámetros diferentes
 - Al usar la superclase que ejecuta la lógica en transacciones diferentes
 - Se puede corregir pasando un arreglo de variables de tamaño variable

Módulo de tabla

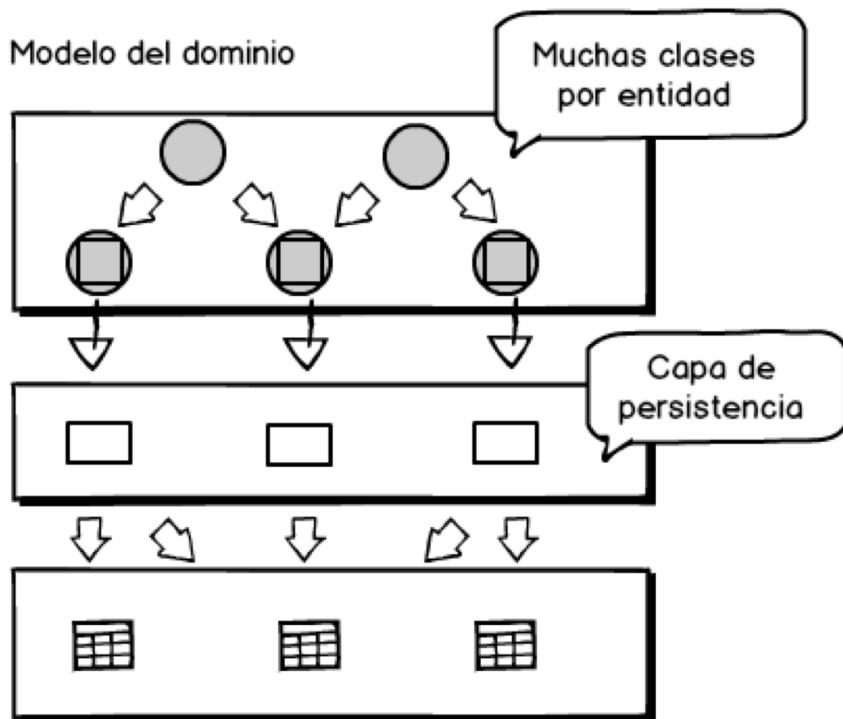
Módulo de Tabla

- Organiza la lógica del dominio
 - Hace 1 clase por cada tabla de la base de datos
 - El modelo de dominio utiliza muchas instancias
 - El módulo de tabla utiliza únicamente 1 instancia por clase
 - La instancia de la clase tiene todos los métodos
 - Estos que actúan sobre los datos

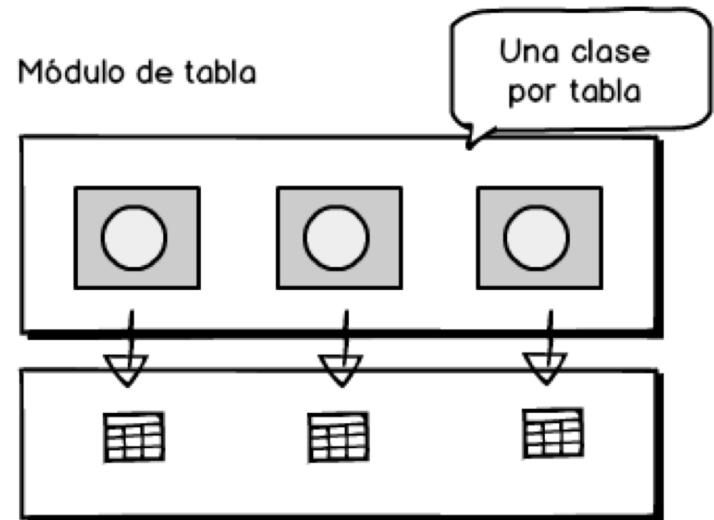
Módulo de Tabla(Cont)

Características del módulo de tabla

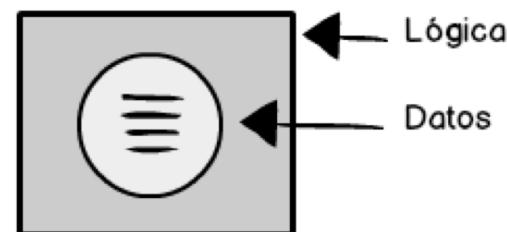
Modelo del dominio



Módulo de tabla



Clase integrada



Módulo de Tabla(Cont)

- Ventajas Generales del Módulo de Tabla
 - Permite empacar la lógica y los datos juntos
 - Mantiene la fortaleza de la BD relacional
 - Aparenta ser un objeto regular
 - Pero no tiene noción de la identidad de los objetos con que trabaja
 - Para ejecutar algo particular a una entidad
 - Se debe pasar algún tipo de referencia de identidad
 - Normalmente es la llave primaria de la BD

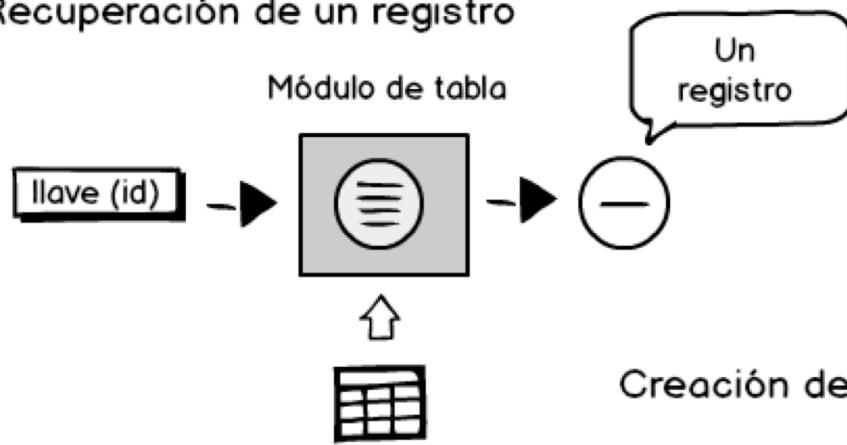
Módulo de Tabla: Estructuración

- Se puede estructurar como
 - Una única instancia
 - Una colección de métodos estáticos

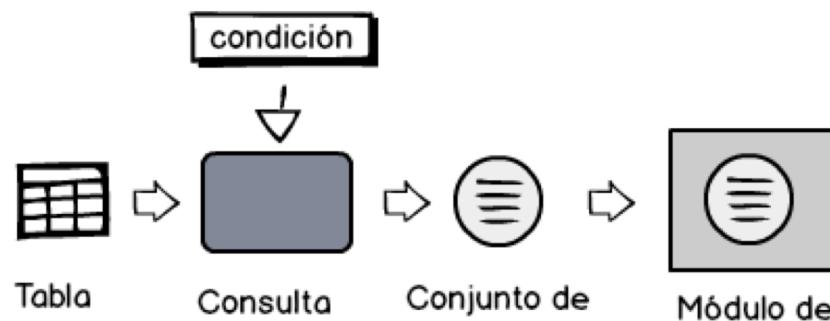
Módulo de Tabla: Estructuración (Cont)

Utilización del módulo de tabla

Recuperación de un registro



Creación de un módulo de tabla



Módulo de Tabla: Estructuración (Cont)

- Una única instancia
 - Permite inicializar el módulo con un conjunto de registros existentes
 - Normalmente el resultado de una consulta SQL
 - Luego se usa la instancia para manipular las filas
 - Modificando el conjunto de registros (Record Set)
 - Las instancias también permiten utilizar herencia
 - Esto permite escribir módulos especializados
 - Estos heredan de un módulo regular

Módulo de Tabla: Estructuración (Cont)

- Colección de Métodos Estáticos
 - AbstRAE conjuntos de datos usados con frecuencia
 - Se pueden unir múltiples tablas
 - Usando vistas o consultas
 - Este método facilita el manejo de datos
 - Cuando la colección de datos es complicada
 - Es más fácil para el desarrollador

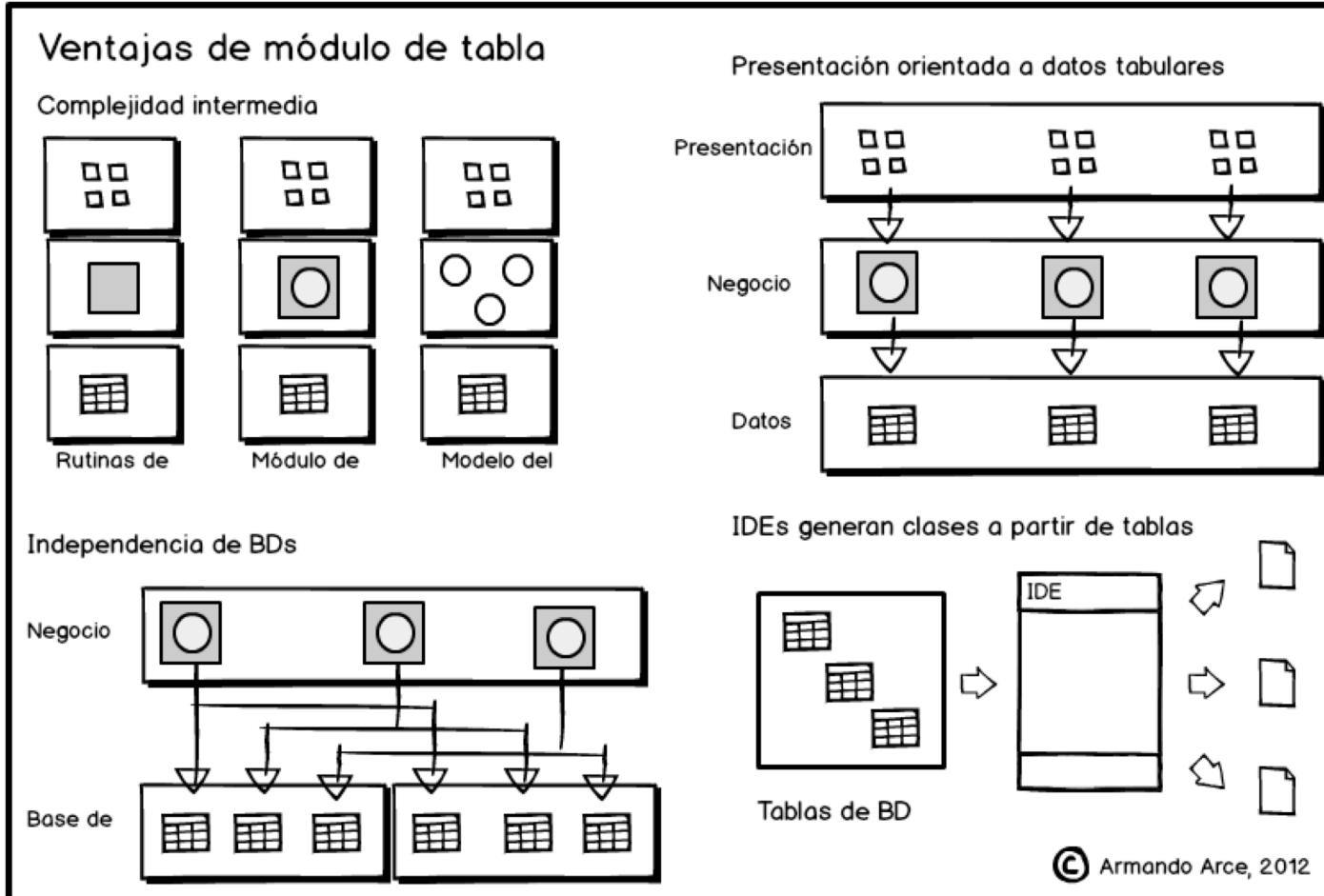
Módulo de Tabla: Ventajas

- Ofrece un punto medio en complejidad
 - Comparado con otras rutinas y modelos
- Se acopla bien a ambientes con BD
 - Las consultas regresan conjuntos de datos
 - Simplifica la comunicación cuando los datos vienen de la BD
 - Resulta más estructurado

Módulo de Tabla: Ventajas(Cont)

- Si la Capa 1 está basada en datos tabulares
 - Resulta una excelente opción usar módulo de tabla
- El enfoque no depende de un motor de BD
 - La lógica permanece en las clases
 - A diferencia de los Procedimientos Almacenados

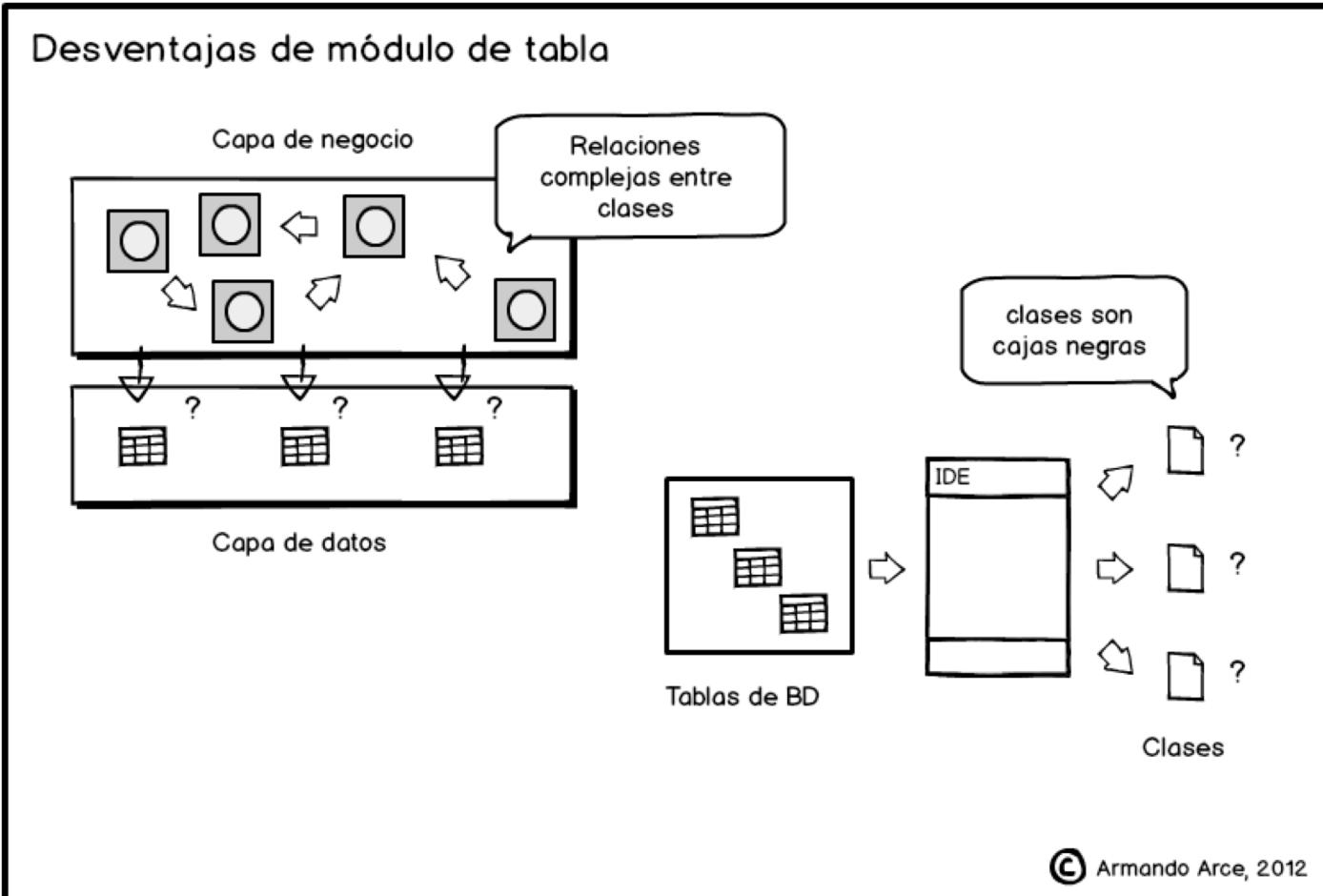
Módulo de Tabla: Ventajas



Módulo de Tabla: Desventajas

- No es fácil expresar relaciones complejas
 - Más si el modelo de objetos y las tablas no se relacionan con facilidad
- Cuando se crea clases a partir de tablas
 - Puede ser muy dependiente del ambiente

Módulo de Tabla: Desventajas



Modelo de Dominio

Modelo de Dominio

- Se crea una red de objetos interrelacionados
- Cada objeto representa un elemento significativo
 - Tan grande como una corporación
 - Tan pequeño como un ítem en una orden de compra
- El modelo inserta una capa completa de objetos
 - Se modela el negocio con que se trabaja

Modelo de Dominio (Cont)

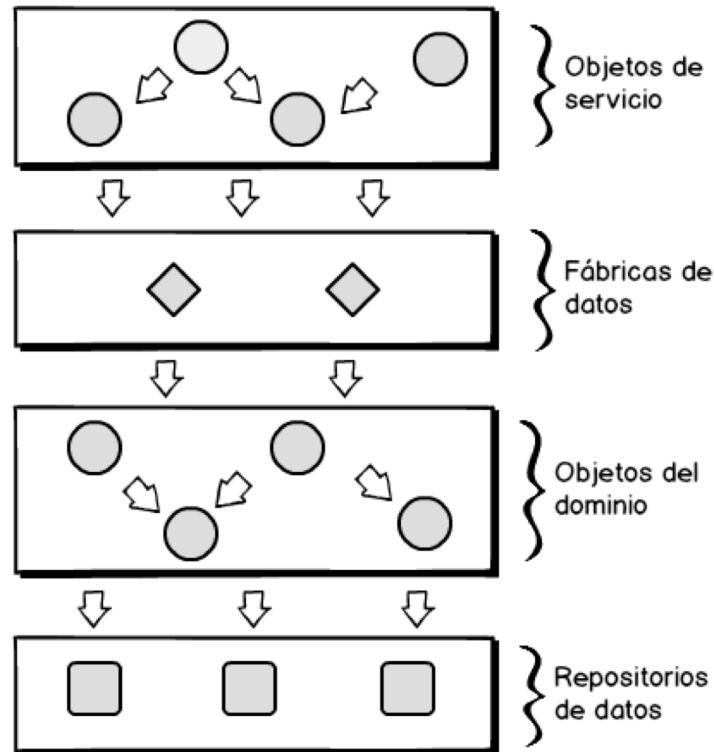
- Lógica del dominio
 - Objetos que simulan datos del negocio
- Lógica de la aplicación
 - Objetos que capturan las reglas del negocio
- A menudo se agrupan datos y proceso
 - Esto agrupa procesos más cercanos a los datos con los que se trabaja

Modelo de Dominio (Cont)

- Debe ser fácil de
 - Construir, modificar y probar
 - Debido a que el negocio cambia constantemente
 - Por eso se debe minimizar el acoplamiento con las otras capas del sistema

Componentes de un Modelo de Dominio

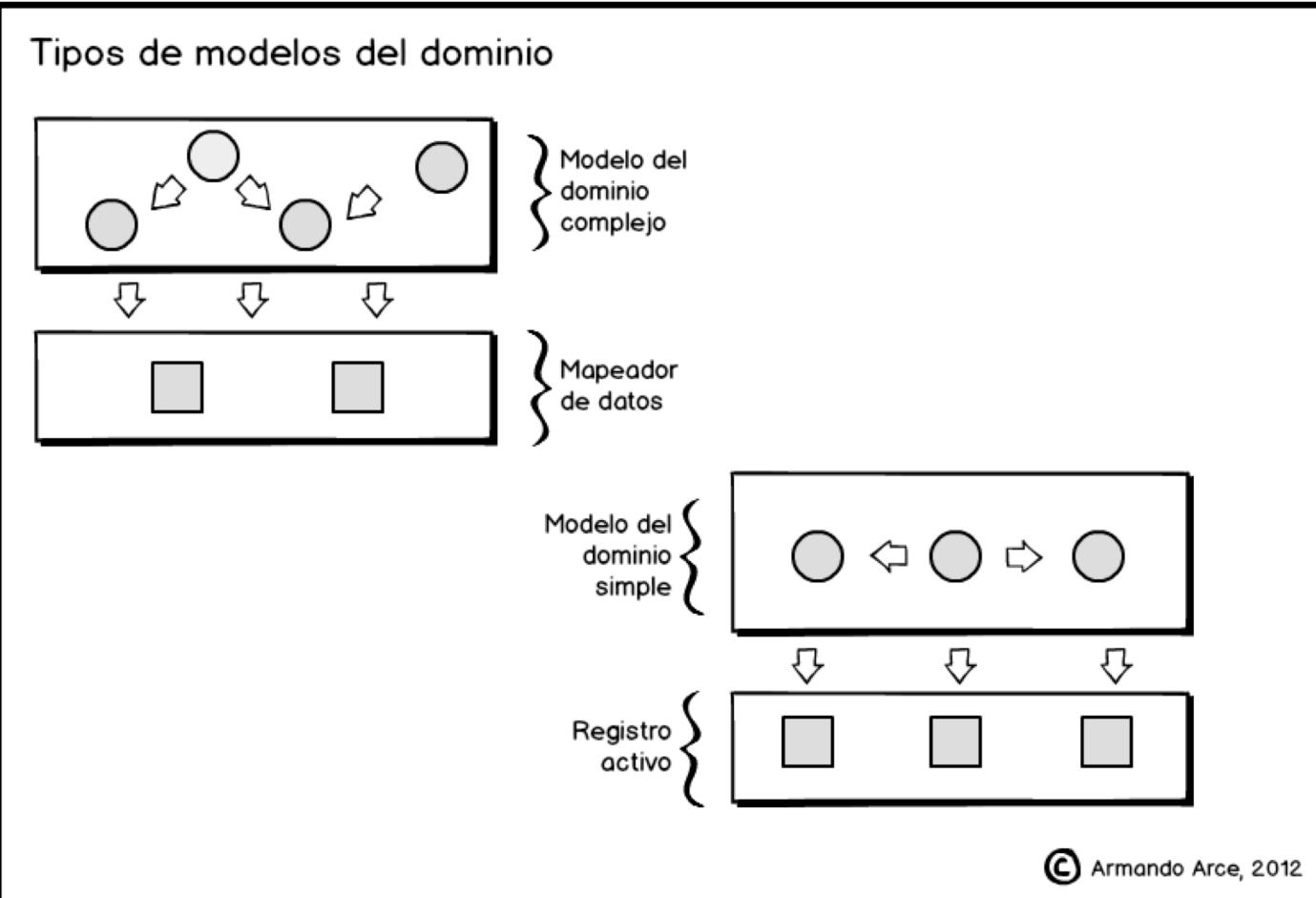
Componentes de un modelo del dominio



Modelo de Dominio: Estructuración

- Cuando es orientado a objetos
 - Luce similar a un modelo de BD
 - Aún cuando la BD sea muy diferente
- En el modelo se agrupan datos y procesos
 - Hay atributos multivaluados
 - Hay una compleja red de asociaciones
 - Se utiliza normalmente jerarquías

Tipos de Modelo de Dominio



Modelo del Dominio Simple

- Más parecido al diseño de una BD
- Tiene un objeto del dominio por cada tabla
- El modelo puede usar la técnica del registro activo
 - Facilita la conexión entre el modelo y la BD
 - Encapsula la complejidad del motor de BD
 - No importa la base de datos ya que maneja el
 - Agregar, consultar, modificar, eliminar

Modelo del Dominio Complejo

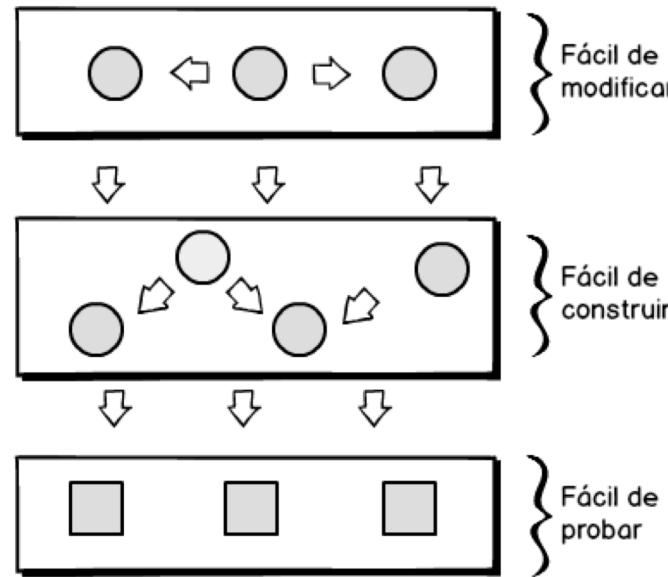
- Luce diferente al diseño de la BD
- Tiene patrones de diseño como
 - Jerarquías y estrategias
 - Complejas relaciones de objetos interconectados
- Maneja mayor la lógica compleja del negocio
 - Es difícil de mapear con una base de datos
- Este puede requerir la técnica de
 - Mapeador de datos (Object-Relational Mapping)

Modelo de Dominio: Ventajas

- Es útil cuando las reglas del negocio cambian constantemente e involucran
 - Cálculos
 - Validaciones
 - Derivación de datos

Modelo de Dominio: Ventajas

Ventajas

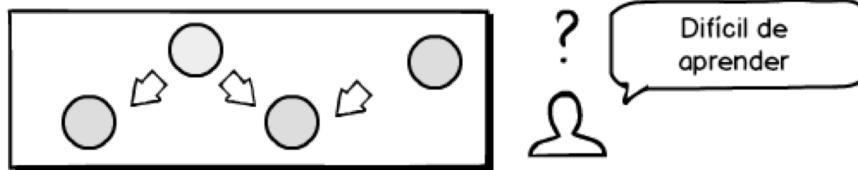


Modelo de Dominio: Desventajas

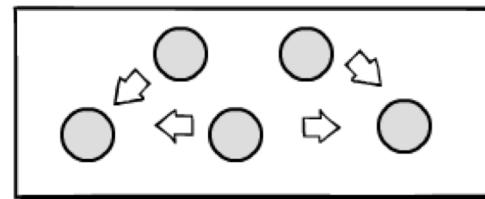
- Aprender a diseñar y usar el modelo es difícil
- Requiere mucha práctica y asistencia
 - Para dominar correctamente los principios
 - Una vez dominados, es difícil dejar de aplicarlos
- Existe un problema de explosión de clases
 - Puede ser que el modelo lleve a una construcción desmedida de clases
 - Esto puede ser señal que se debe utilizar otro modelo

Modelo de Dominio: Desventajas

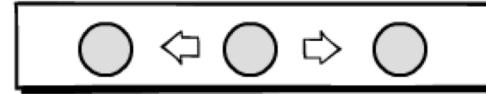
Desventajas



Explosión de
clases



No es útil en
aplicaciones
simples



Capa de servicios

Capa de Servicio

- Define las fronteras de la aplicación
- Encapsula la lógica de aplicación
- Controla las transacciones
- Coordina las respuestas
 - Y la implementación de sus operaciones
- Se puede construir de diferentes formas
 - La diferencia está en la ubicación de las responsabilidades

Capa de Servicio (Cont)

- Responde a la necesidad que la aplicación esté disponible en red.
- La funcionalidad debe estar disponible para varios tipos de sistema
 - La interoperabilidad es clave en el diseño
- Se puede requerir múltiples protocolos de comunicación
- Permitir requerimientos operacionales variantes

Capa de Servicio (Cont)

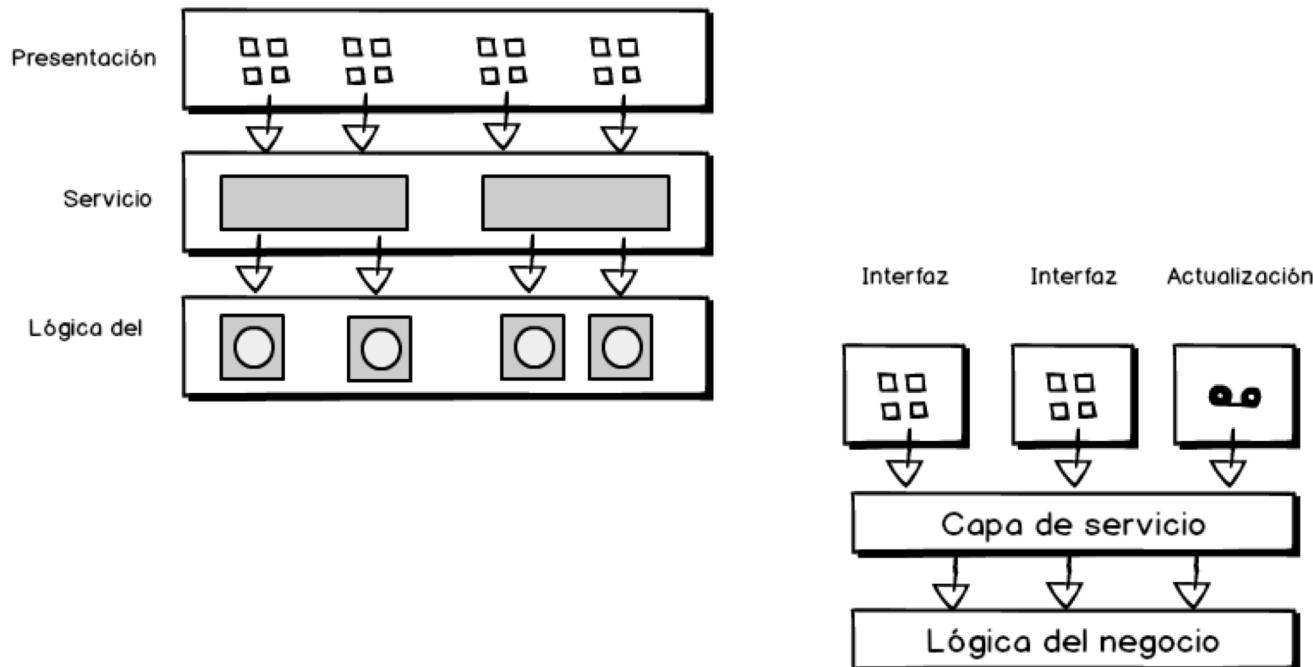
- Es importante mantener el app desacoplado
 - Teniendo disponibles las funciones
 - Para diversas aplicaciones
- Es deseable mantener separado los elementos
 - Encargados de la lógica del negocio
 - Encargados de los protocolos de comunicación
 - Encargados de la transformación de datos

Capa de Servicio (Cont)

- Es importante también tener el ambiente optimizado
 - Mejora el servicio al cliente y la respuesta a este
- La comunicación puede ser variada
 - Un cliente externo puede requerir SOAP
 - Un cliente interno puede requerir API

Capa de Servicio (Cont)

Características de la capa de servicios



Servicios y Operaciones

- Para identificar operaciones se usa
 - El modelo de casos de uso
 - El diseño de la interfaz de usuario
- Con los casos de uso del sistema definidos
 - Se puede crear la lista de métodos de la interfaz
 - Esto permite exponer un contrato y los alcances

Servicios y Operaciones

- La capa de servicios evoluciona diferente al resto
- Es la única interfaz para la Capa 1
 - Y la interacción con los procesos del negocio
- Se recomienda que no incluyan CRUD
 - Salvo los casos de uso que su función es la persistencia de los datos

Tipos de Lógica

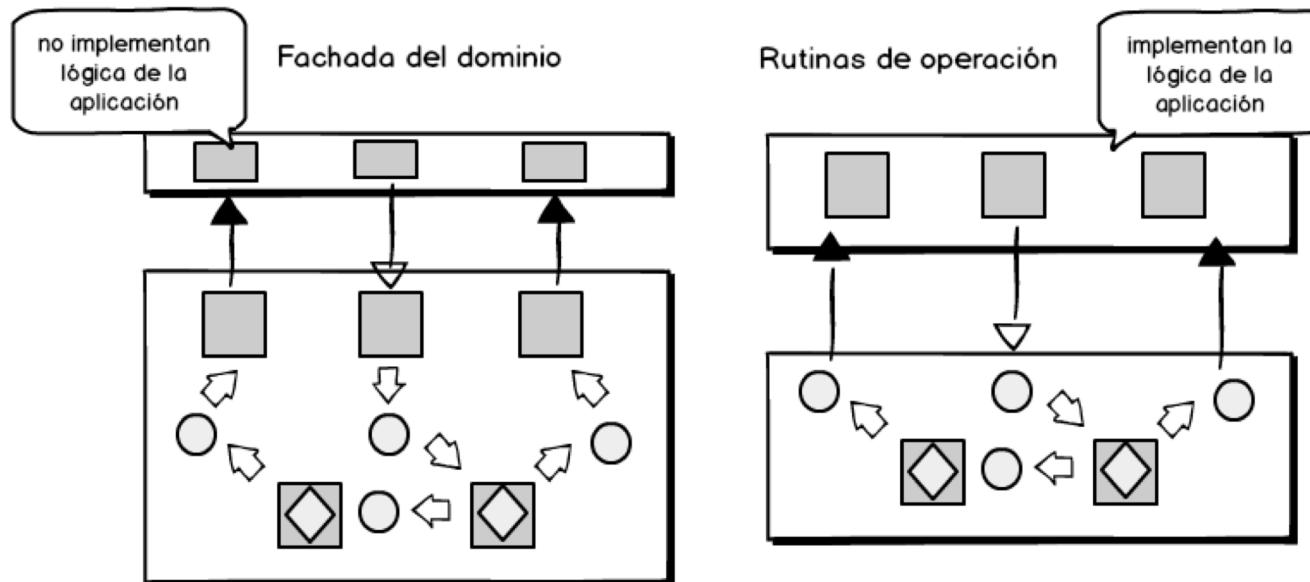
- La lógica del dominio
 - Tiene que ver con el dominio del problema
- La lógica de la aplicación
 - Se le conoce como la lógica del flujo de trabajo
- La mezcla puede tener efectos no deseados
 - Los objetos del dominio son menos reutilizables
 - Dependen de la lógica y paquetes específicos
 - La mezcla hace difícil reimplementar la misma lógica en otra herramienta

Variantes de la implementación

- Fachada del dominio
 - Implementan capas livianas sobre el modelo
 - No implementan ninguna lógica del negocio
 - Frontera y conjunto de operaciones de interacción
- Rutinas de operación
 - La capa de operación implementa clases delegadas
 - Estas implementan la lógica del dominio
 - Las operaciones son implementadas como rutinas
 - Cada una de las clases forma un servicio de aplicación

Variantes de la implementación

Implementación de la capa de servicios



Acceso: Remoto y No Remoto

- Las capas son de granularidad gruesa
 - Las hace óptimas para invocaciones remotas
 - Conlleva el costo de lidiar con distribución de objetos
 - Esto representa una gran cantidad de trabajo extra
 - Lo ideal es trabajar inicialmente de forma local
 - Las llamadas remotas se agregar luego
 - Utilizando fachadas remotas sobre la capa de servicios
 - Haciendo que los objetos implementen interfaces remotas

Ventajas

- Define un conjunto común de operaciones
 - Disponible a múltiples clientes
 - Mediante múltiples tipos de conexiones
- La aplicación es más reutilizable
 - Con la capa de servicio se tiene la lógica de presentación
 - Esta actúa como un API que encapsula el App

Ventajas

- Permite controlar y validar solicitudes
 - Permite poner control de transacciones
 - Se puede verificar la seguridad
 - El mecanismo de servicioestá desacoplado de la lógica
 - Permite agregar nuevas interfaces
 - Permite cambiar la implementación de la aplicación
 - Genera un impacto mínimo en los usuarios
 - Desacopla el código de la interfaz con el código del servicio
 - Maneja ambos componentes en tiers separados
 - Esto aumenta la flexibilidad

Desventajas

- No es útil con rutinas de transacción
 - Las rutinas son poco complejas para su uso
- No es útil si solo existe un tipo de cliente
 - La capa de servicio separada pierde sentido
- Cuando la granularidad es muy fina
 - Puede generar demasiados pasos para ejecutar una sola operación