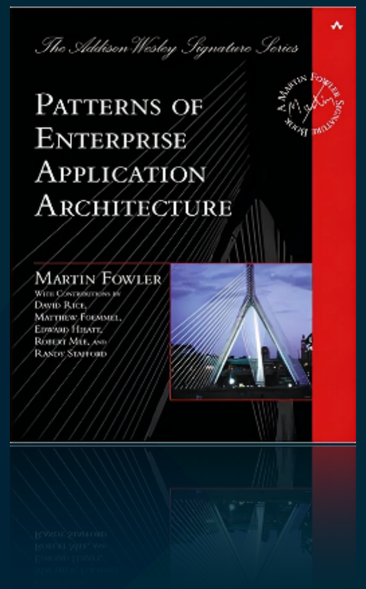
FullCycle

Patterns of Enterprise Application Architecture



FullCycle

Patterns

- Domain Logic
- Data Source Architecture Patterns
- Object-Relational Behavioral
- OO Structural Patterns
- OO Metadata Mapping Patterns
- Web Presentation
- Distribution Patterns
- Offline Concurrency Patterns
- Session State Patterns
- Base Patterns

FullCycle

Layering

- Separação de responsabilidades
- Abstração
- Layer vs Tier

FullCycle

Layer / Lógica

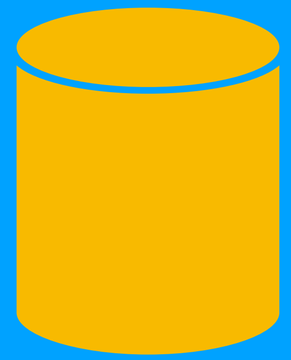
FullCycle

Tier / Física

Client

APP

DB



FullCycle

3 Layers Architecture

- Presentation
  - Display Information
  - CLI
  - HTTP Requests
- Domain
  - Coração da aplicação
  - Regras de negócio
- Data Source
  - Banco de dados
  - Mensageria

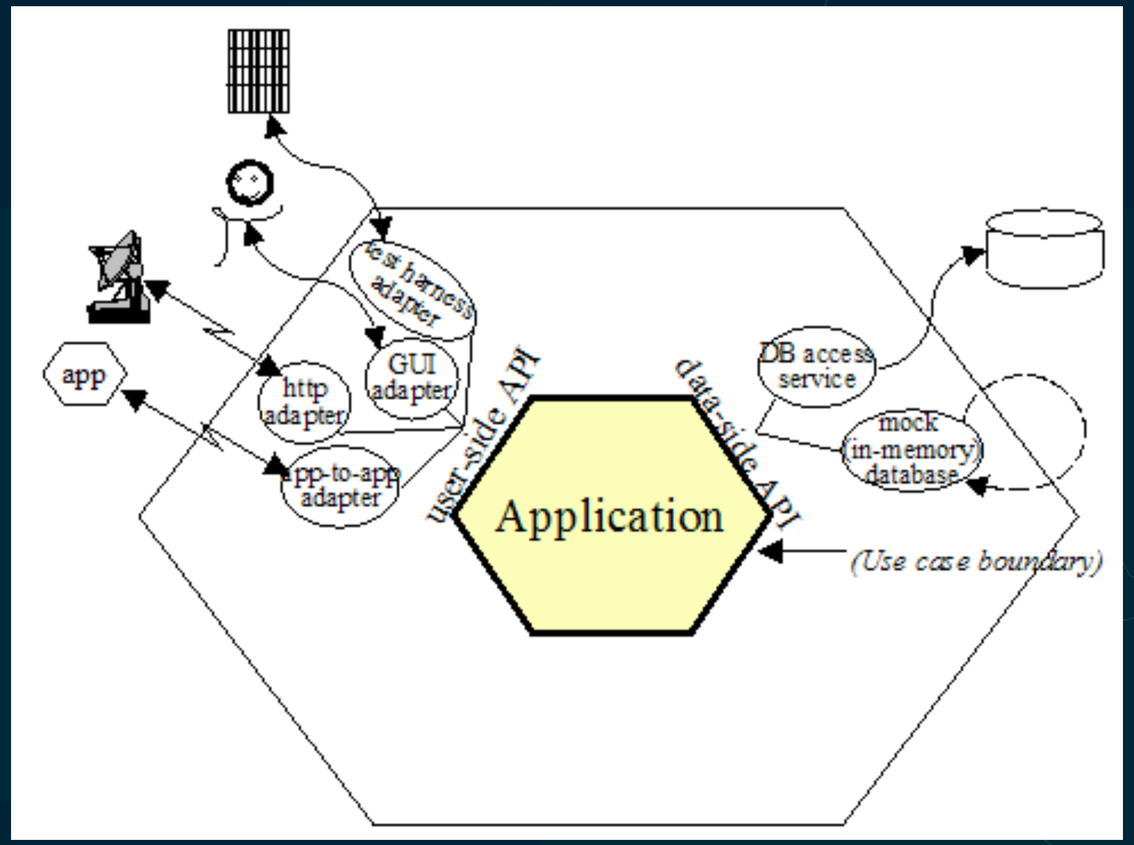
FullCycle

3 Layers Architecture - Regra de Ouro

Domínio e Data Source nunca podem depender da apresentação

FullCycle

3 Layers Architecture - Arquitetura Hexagonal



https://alistair.cockburn.us/hexagonal-architecture/

FullCycle

Domain Logic

- Transaction Script
- Domain Model

FullCycle

Transaction Script

- Regras de negócio em torno de transações
- Segue formato mais “procedural"
- Simples e direto
  - Ex: ProcessarPedido
    - Verifica Estoque
    - Aplica Promoções
    - Fidelidade
    - Cria Pedido no Banco
    - Etc

FullCycle

Transaction Script

- Vantagens
  - Orientado a transações
  - Totalmente direto ao ponto
  - Adequado para requisitos simples

- Desvantagens
  - Alta complexidade quando o sistema cresce
  - Trabalha normalmente de forma síncrona

FullCycle

Domain Model

- Domínio de uma aplicação em “objetos de domínio”
- Encapsula a Lógica de Negócios
- Regras de Negócio em primeiro lugar
- Validações

FullCycle

Domain Model

- Vantagens
  - Clareza e Expressividade
  - Flexibilidade
  - Foco na evolução
  - Alta Testabilidade

- Desvantagens
  - Complexidade Inicial
  - Curva de Aprendizado
  - “Escalabilidade" e “Performance"
  - Fácil se perder com Overengineering

FullCycle

Table Module

- Organização por tabelas do banco de dados
- Regras de negócio segregadas por tabela
- Alta coesão
- Acoplamento forte com o banco de dados

FullCycle

Table Module

- Organização por tabelas do banco de dados
- Regras de negócio segregadas por tabela
- Alta coesão
- Acoplamento forte com o banco de dados

FullCycle

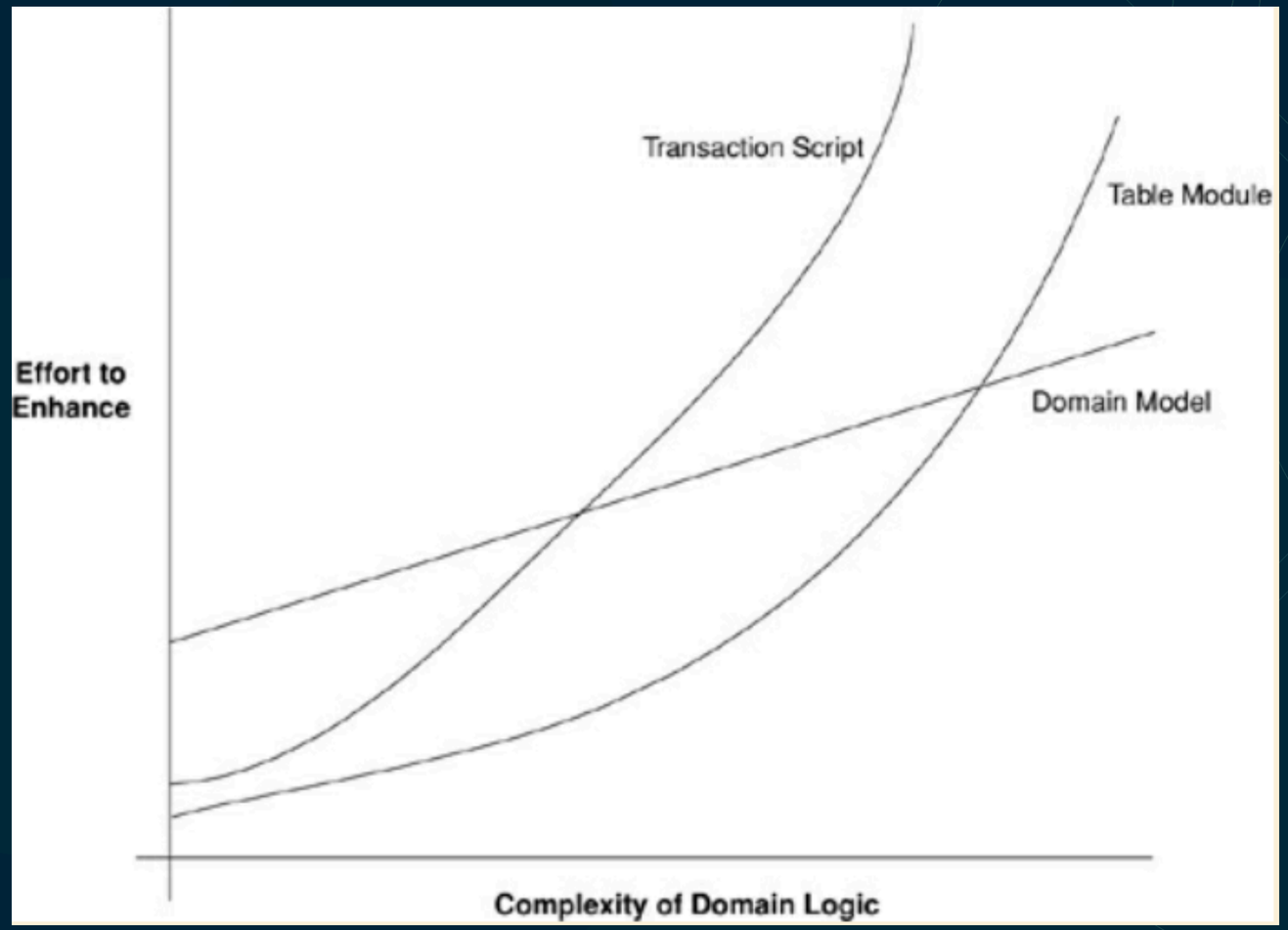
Table Module

- Vantagens
  - Simples
  - Fácil mapeamento
  - Simples manutenção em diversas situações
  - CRUD

- Desvantagens
  - Duplicação de código
  - Baixa reutilização
  - Baixa regra de domínio

FullCycle

Comparativo



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 29). Pearson Education. Kindle Edition.

FullCycle

Service Layer

- Camada intermediária entre a camada de apresentação e o acesso a dados
- Expõe funcionalidades de alto nível para os “clients"
- Encapsula a lógica de negócios
- Orquestra a ordem das operações
- Tem acesso a camada de dados
- Gerencia transações

FullCycle

Service Layer

- Vantagens
  - Separação de responsabilidades
  - Reutilização
  - Melhor estabilidade do que transaction scripts
  - Flexibilidade para implementar

- Desvantagens
  - Complexidade ao longo do tempo
  - Maior acoplamento
  - Complexidade no trabalho em equipes
  - Tende a trabalhar com modelos de domínio anêmicos

FullCycle

Service Layer vs Transaction Script

- Apesar de terem idéias semelhantes, possuem estruturas diferentes
- Transaction Scripts normalmente trabalham no formato de funções ou conjunto de funções específicas para cada operação. O que tende a gerar muita duplicação de código
- Service Layer tende a ser mais flexível e reutilizável

FullCycle

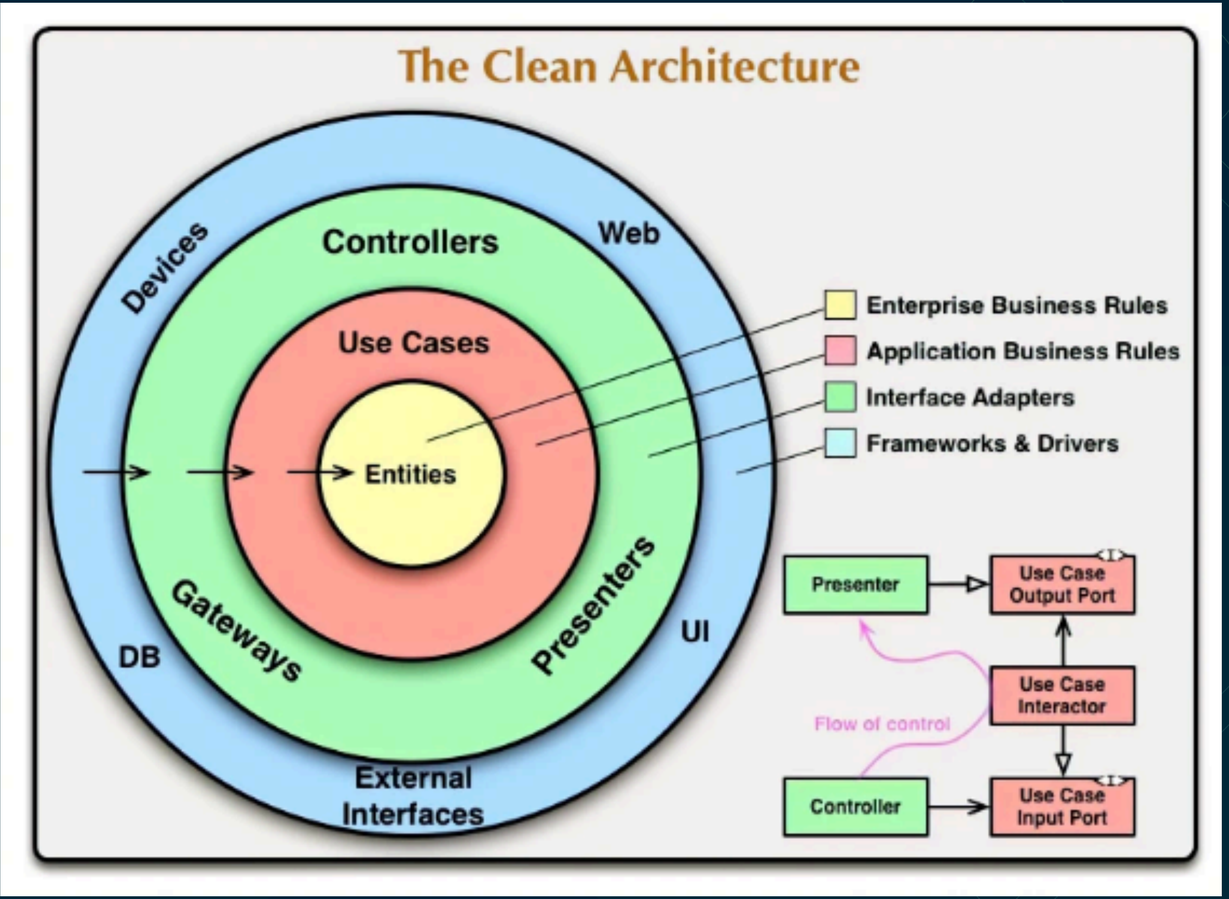
Gateway

Objeto que encapsula e acessa sistemas ou recursos externos

Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 466). Pearson Education. Kindle Edition.

FullCycle

Gateway



https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

FullCycle

Gateway - 2 formas

Uma instância para cada linha retornada por uma consulta (Row Data Gateway)

Figure 3.1.

Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 34). Pearson Education. Kindle Edition.

FullCycle

Gateway - 2 formas

Estrutura genérica de tabelas e linhas que imitam a natureza tabular de um banco (Record Set). Uma classe por tabela. (In Memory Table) -> Table Data Gateway

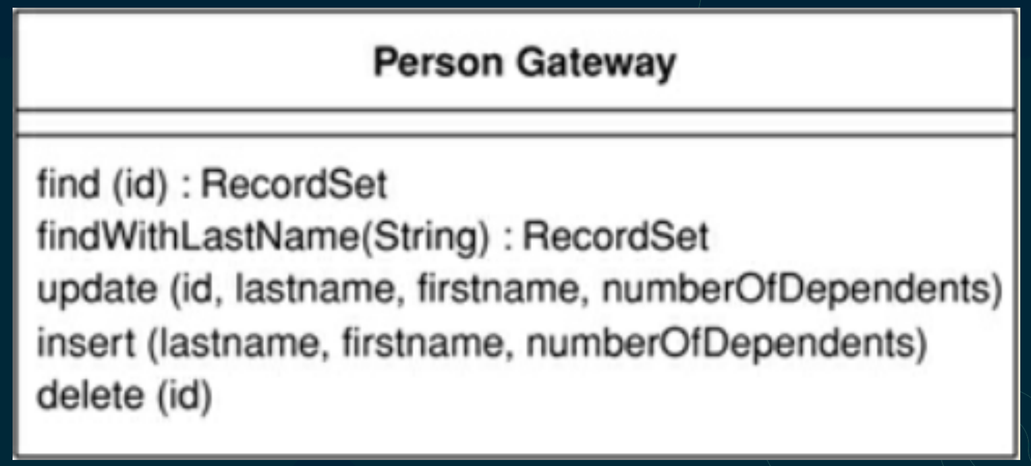


Figure 3.2.

Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 34). Pearson Education. Kindle Edition.

FullCycle

Table Module (Domain Logic)

Talvez faça sentido utilizar Row Data Gateway ou Table Data Gateway

FullCycle

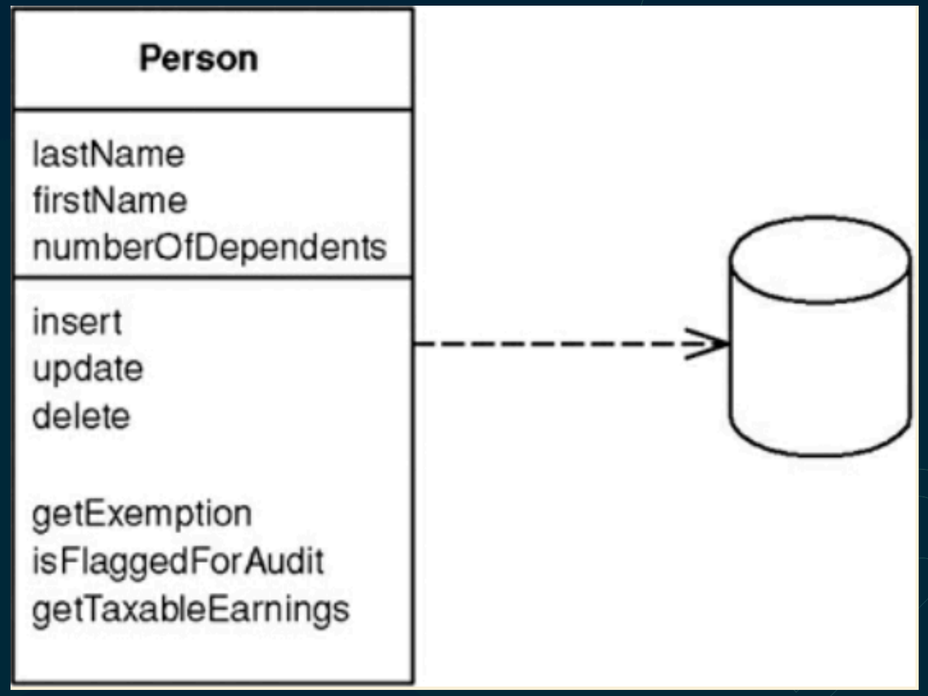
E sobre Domain Model?

Em sistemas simples talvez, pois o domínio por ser 1:1 com o banco.

FullCycle

Active Record

Um objeto que encapsula uma linha, tabela ou view e ainda adiciona lógica de domínio em seus dados.



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 160). Pearson Education. Kindle Edition.

FullCycle

Active Record

- Carrega dados e comportamento
- Lógica de domínio tende a "vazar" para o comportamento do banco de dados
- Extremamente simples de usar
  - Rails
  - Laravel
  - Django
- Recomendação pessoal:
  - Separe o modelo de domínio do modelo do active record
  - Enquanto Data Table ou Row Table apenas é acoplado no banco, o Active Record tende também a ficar acoplado ao domínio.

FullCycle

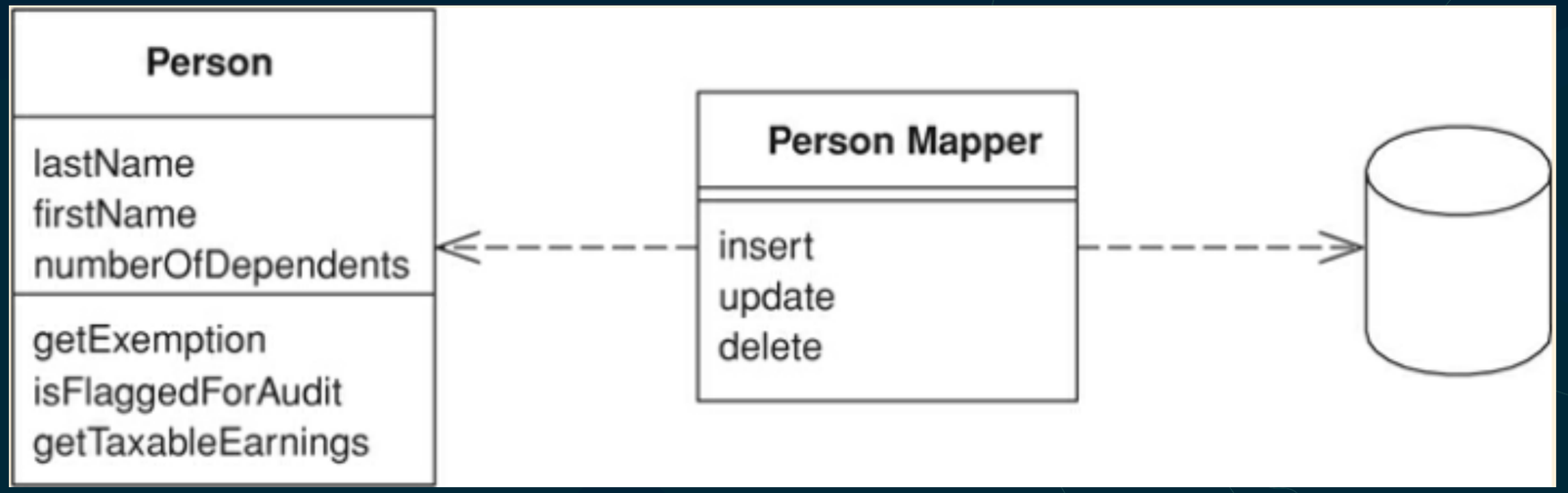
Active Record - "Quando não usar”

- Domínios complexos
- Active Record precisa de um match exato com as tabelas do banco
- Conforme a complexidade do domínio aumenta, os objetos de domínio deixam de ser 1:1 com o banco

FullCycle

Data Mapper

Uma camada de mapeamento move dados entre os objetos e banco de dados enquanto os mantém independente um do outro.



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 165). Pearson Education. Kindle Edition.

FullCycle

Data Mapper

- Ideal para domínios complexos
- Domínio não fica refém da estrutura do banco de dados
- Separe as entidades (objetos de mapeamento) do modelo de domínio
  - Utilizar as mesmas entidades fará com que seu domínio fique anêmico

FullCycle

Notas

- Domínios simples: Active Record
- Domínios complexos: Data Mapper
- Não há verdade absoluta

FullCycle

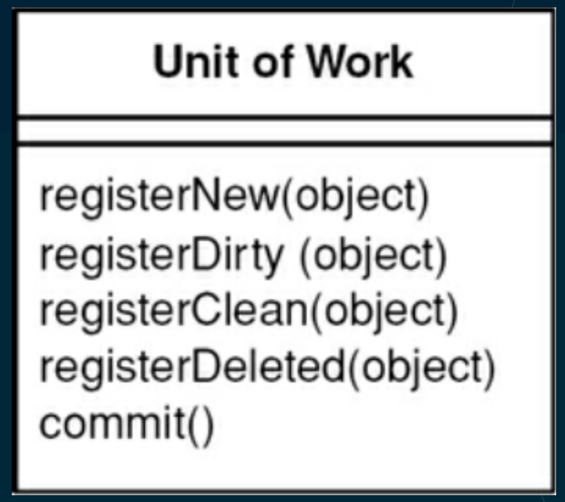
Atomicidade

- Como trabalhar com diversos objetos?
- Como garantir a persistência?
- Como realizar compensação?

FullCycle

Unit of Work

Mantém uma lista de objetos afetados por uma transação de negócios e coordena as mudanças e os problemas de concorrência.



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 184). Pearson Education. Kindle Edition.

FullCycle

Unit of Work

- Register New: Um novo objeto que foi criado durante a execução (Insert)
- Register Dirty: Objeto já existente que foi carregado e modificado (update)
- Register Clean: Objeto recuperado e não modificado
- Register Deleted: Objeto recuperado e marcado para remoção
- Commit: Persiste a transação no banco de dados

FullCycle

Identity Map

Garante que cada objeto é carregado apenas uma vez e o mantém em um mapa de controle. Quando há busca pelo objeto, primeiramente ela é realizada no mapa.

Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 195). Pearson Education. Kindle Edition.

FullCycle

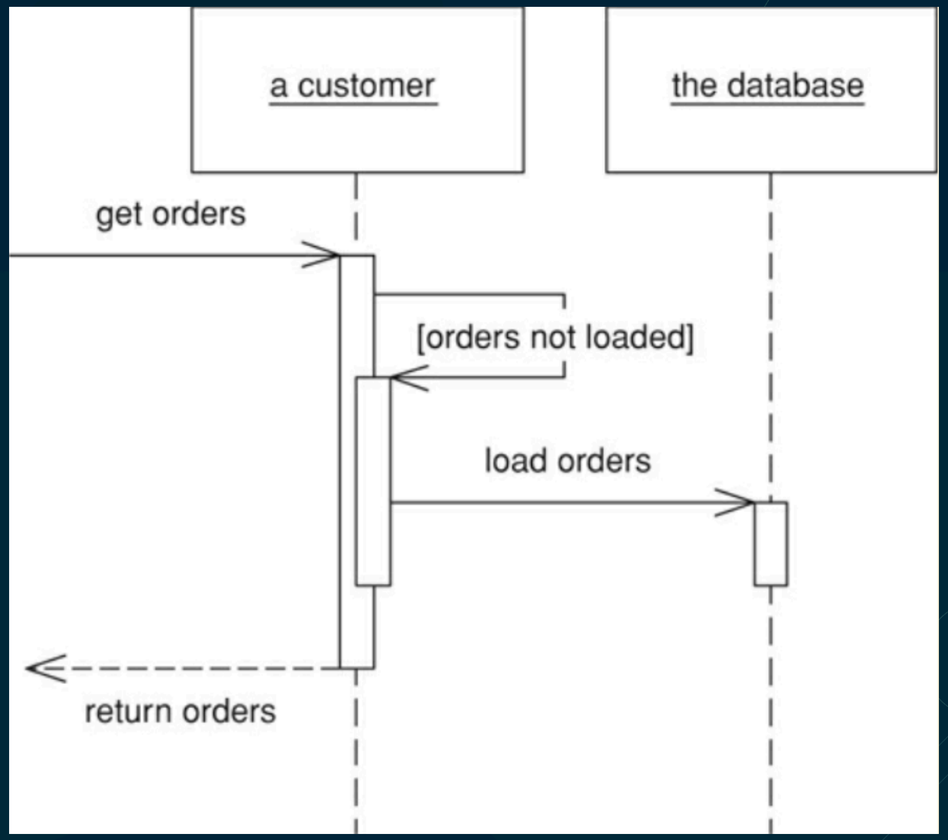
Identity Map

- Estrutura de dados
- Carregamento das entidades em memória
- Garante o carregamento apenas uma vez
- Melhora performance e remove inconsistências
- Mantém o controle dos objetos que foram criados, modificados ou marcados para remoção para ser utilizado em conjunto com o UoW

FullCycle

Lazy Load

Um objeto que não possui todos os dados que você talvez precise, mas sabe onde buscá-lo.



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 198). Pearson Education. Kindle Edition.

FullCycle

Lazy Load

- Carrega os objetos somente quando necessário
- No carregamento inicial ao invés de ter os dados reais, esses dados são substituídos por proxies (apenas representações do objeto real, sem dados)
- Quando os dados relacionados são acessados o proxy carrega os dados do banco
- Cuidado enorme com N+1

FullCycle

Repository

Mediação entre a camada de domínio e a camada de dados usando uma interface para acessar os objetos de domínio.

Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 322). Pearson Education. Kindle Edition.

FullCycle

Repository

- Mediação entre os objetos de domínio e o data mapper
- Recebe objetos de domínio atendendo uma especificação
- Retorna objetos de domínio
- Normalmente faz diferentes combinações de especificações gerando o SQL desejado para atender um critério
- “Promove" o padrão “specification"

FullCycle

Repository

```
interface Specification<T> {
  isSatisfiedBy(item: T): boolean;
}
```

Repository

```
class UserEmailSpecification implements Specification<User> {
  constructor(private email: string) {}

  isSatisfiedBy(user: User): boolean {
    return user.email === this.email;
  }
}
```

FullCycle

Repository

```
interface UserRepository {
  findBySpecification(specification: Specification<User>): User[];
  // Other methods in the UserRepository interface...
}
```

FullCycle

Repository

```
class SqlUserRepository implements UserRepository {
  // Other implementation details...

  findBySpecification(specification: Specification<User>): User[] {
    const users: User[] = [];
    for (const user of this.users) {
      if (specification.isSatisfiedBy(user)) {
        users.push(user);
      }
    }
    return users;
  }
}
```

FullCycle

Repository

```
// Exemplo de classe User
class User {
  constructor(public id: number, public name: string, public email: string) {}
}

// Exemplo de uso
const userRepository: UserRepository = new SqlUserRepository(); // Instanciand

const userSpecification: Specification<User> = new UserEmailSpecification("use

// Especificação de busca por email

const users: User[] = userRepository.findBySpecification(userSpecification);

console.log(users); // Exibindo os usuários encontrados
```

FullCycle

Web Presentation Patterns

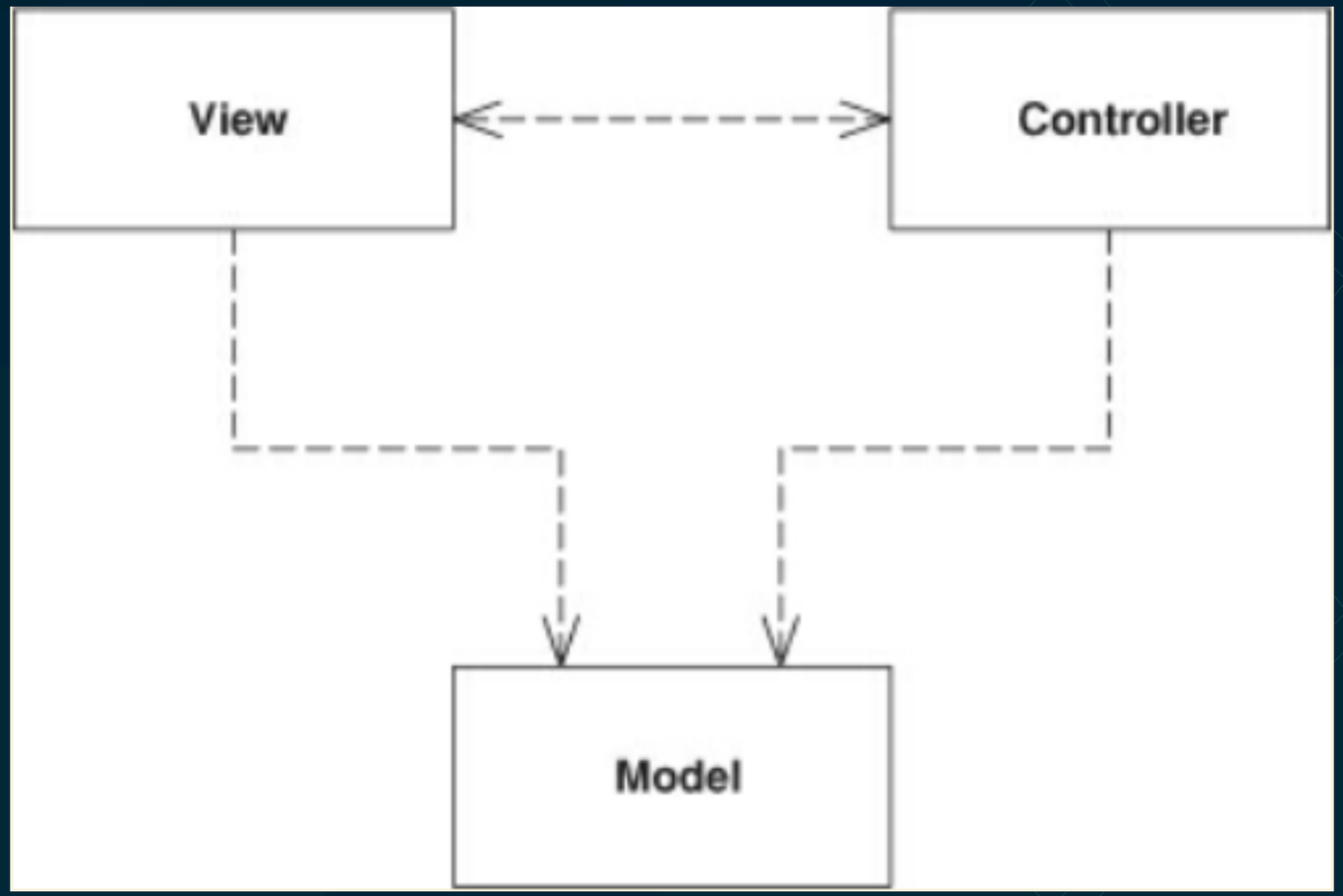
FullCycle

MVC

- M = Model; V = View; C = Controller;
- Criado na década de 70
- Para a Smalltalk
- Frameworks com UI
- Model é um objeto que representa alguma informação do domínio
  - Possui todos os dados que serão utilizados pela UI
- View representa a apresentação dos dados do Model na UI
- Controller recebe a input do usuário, manipula o model e faz com que a view seja atualizada
- UI é a combinação da View e do Controller

FullCycle

MVC



Martin, Fowler. Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series (Fowler)) (p. 330). Pearson Education. Kindle Edition.

FullCycle

MVC

- A View depende do Model, mas o Model não depende da View
- Idealmente se você possui diversas “janelas" em sua UI e o Model é atualizado, espera-se que os dados das “janelas" sejam atualizados automaticamente;
- Para essa atualização, recomenda-se utilizar patterns como o Observer para a propagação de eventos ou mesmo um listener; (Reatividade)

FullCycle

Distribution Patterns

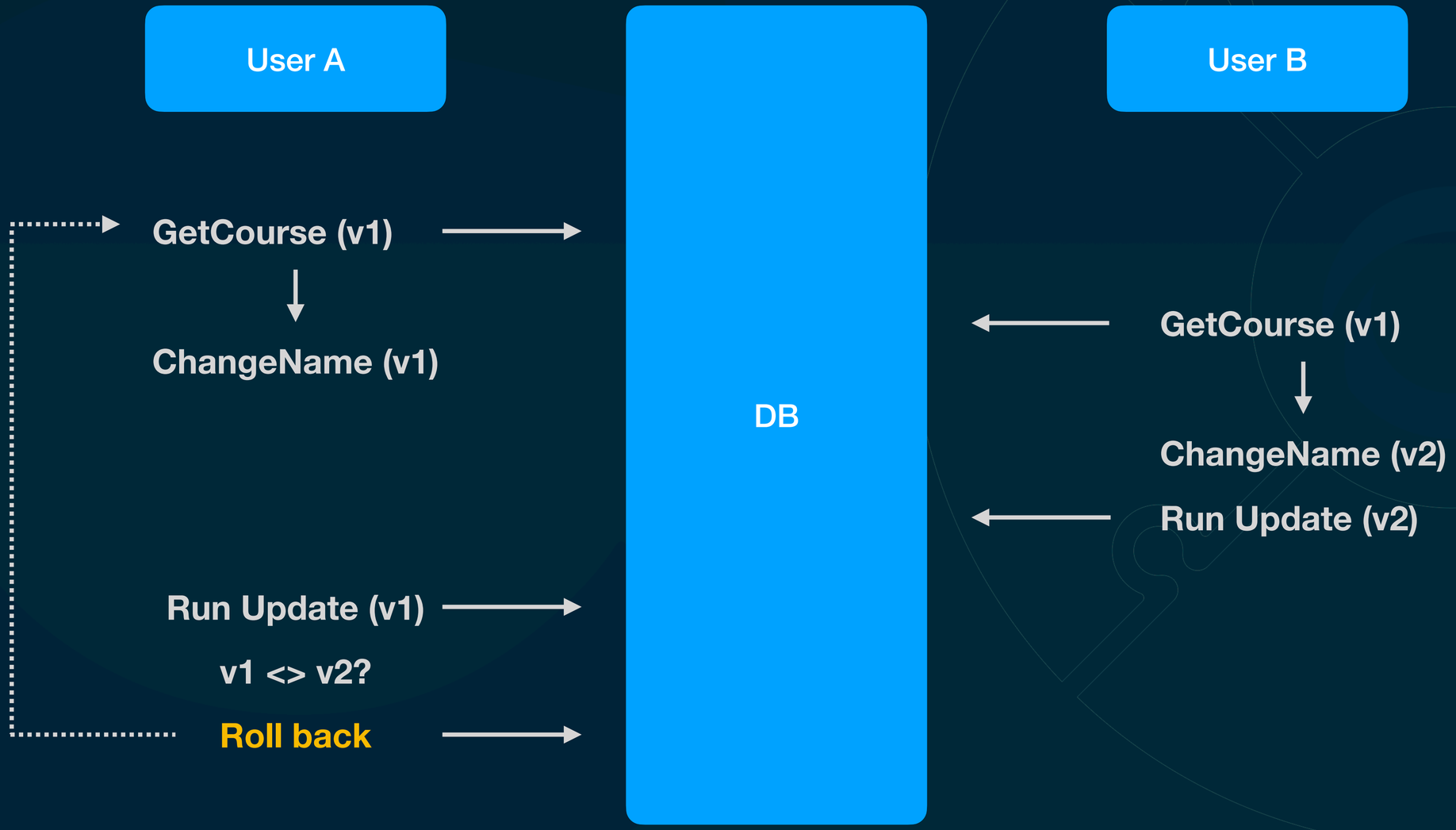
FullCycle

Lock otimista (Optimistic Offline Lock)

Detecta e previne conflitos entre transações concorrentes realizando o roll back da transação.

FullCycle

Lock otimista (Optimistic Offline Lock)



FullCycle

Lock otimista (Optimistic Offline Lock)

- Courses Table (ID, Name, Version)

- Select * from courses where id = 1;
  - Name -> My Course
  - Version -> 1

- Update from courses set
  name = ‘New Name Y’, version=2
  where id = 1 and version = 1;
- Zero registros alterados. Refaz processo.

- Select * from courses where id = 1;
  - Name -> My Course
  - Version -> 1

- Update from courses set
  name = ‘New Name X’ , version=2
  where id = 1 and version = 1;

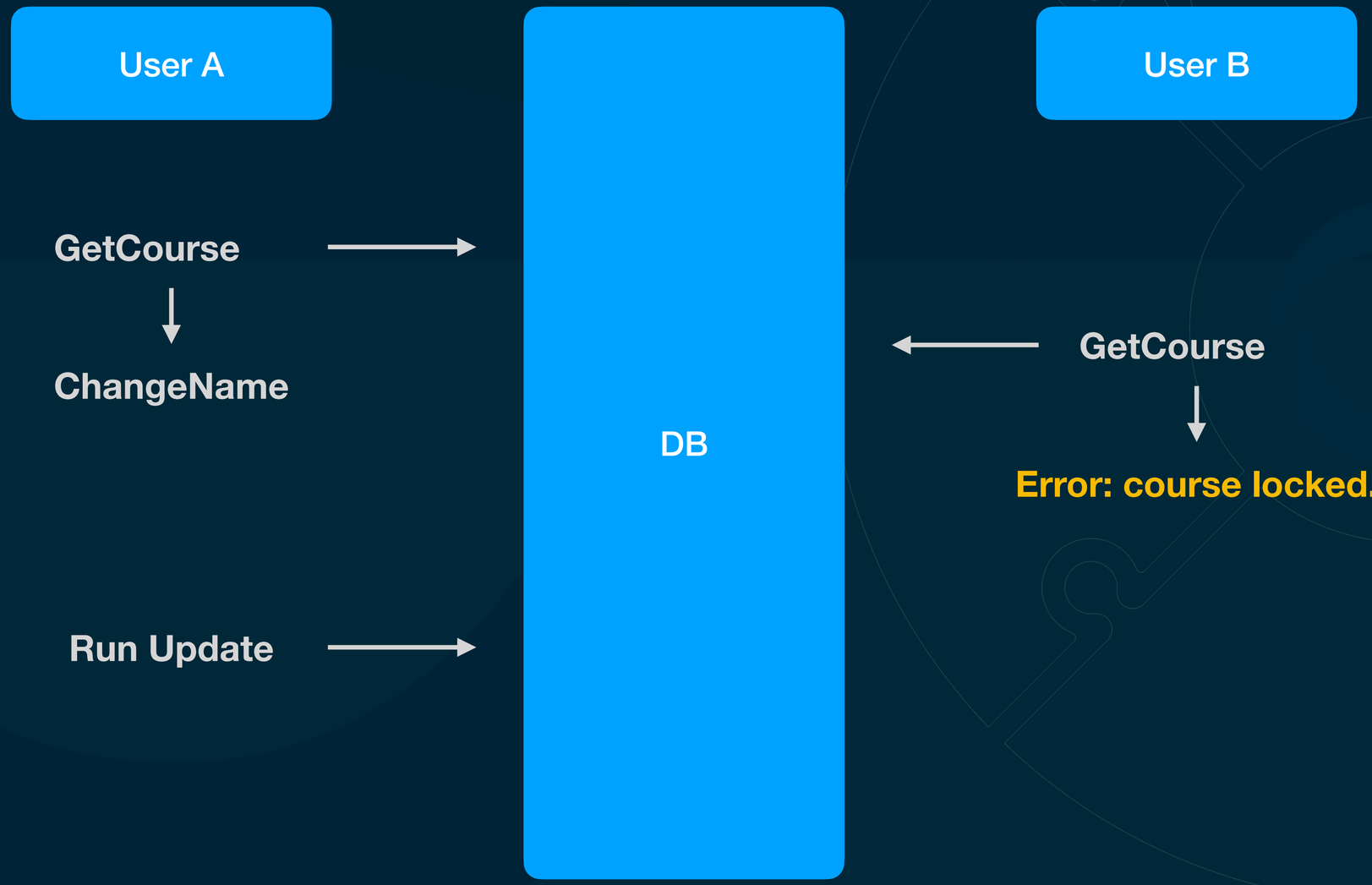
FullCycle

Lock pessimista (Pessimistic Offline Lock)

Evita conflito entre transações concorrentes permitindo apenas uma transação de cada vez.

FullCycle

Lock pessimista (Pessimistic Offline Lock)



FullCycle

Lock pessimista (Pessimistic Offline Lock)

```
BEGIN;
SELECT * from courses where id=1 for update;
UPDATE courses set name='New Name' where id=1;
COMMIT;

BEGIN;
SELECT * from courses where id=1 for share;
UPDATE courses set name='New Name' where id=1;
COMMIT;
```