MBA
USP
ESALQ

Arquitetura Limpa (Clean Architecture) I-II

Prof. Helder Prado Santos

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.

A Arquitetura Limpa

- Termo criado em **2012** por **Robert C. Martin.**
- Focada em **reduzir o acoplamento** entre as camadas arquiteturais.
- Comunicação entre as camadas é realizada a partir de **contratos**
- **Orientada aos casos de usos** da aplicação.

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

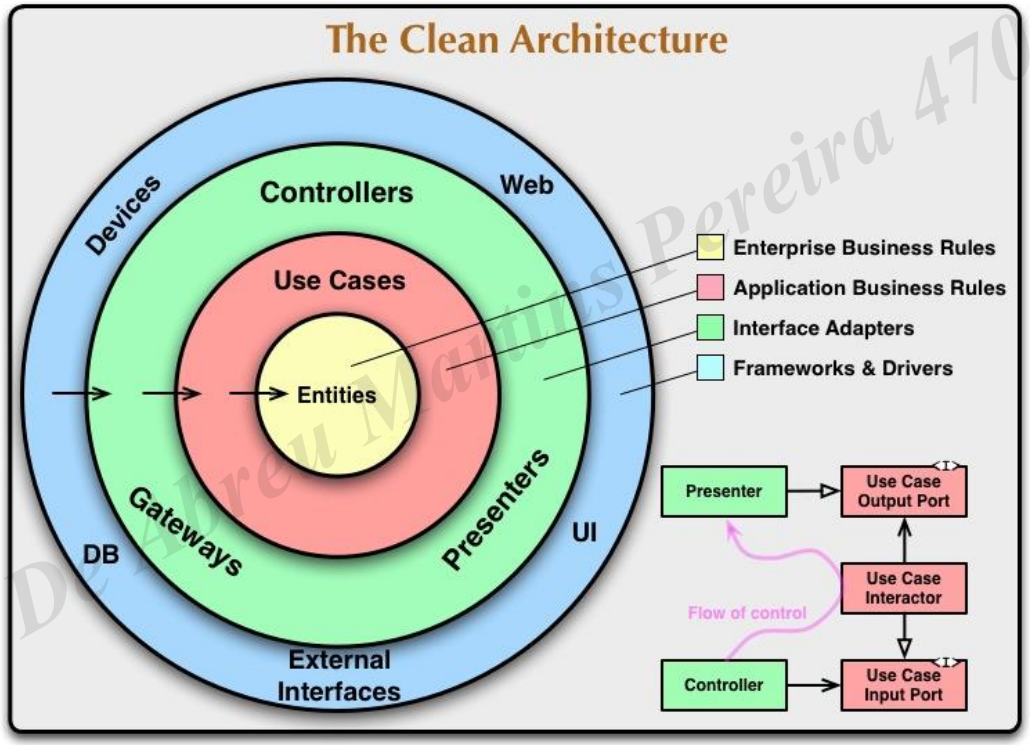
Princípios Fundamentais

- **Independência de Frameworks:** Arquitetura agnóstica a frameworks
- **Testabilidade:** Facilitação de testes unitários
- **Desacoplamento:** Evitar acoplamento entre camadas
- **Separação de Responsabilidades:** Independência entre regras de negócio e detalhes de implementação

O que é uma boa arquitetura?

- Aquela que torna o sistema:
  - Fácil de **entender**
  - Fácil de **desenvolver**
  - Fácil de **manter**
  - Fácil de **implantar**

Camadas da Arquitetura Limpa



Entidades (Entities)

- ❑ Ditam as **regras de negócio** da aplicação;
- ❑ Também conhecido como o **domínio da aplicação;**
- ❑ Detalhes técnicos **não devem impactar** as regras de negócio da aplicação.

Casos de Uso (Use Cases)

- ❑ **Ações e funcionalidades** da aplicação
- ❑ Detalhes técnicos **não devem impactar** nos casos de uso
- ❑ **Single Responsibility Principle** (SRP)
- ❑ Exemplos:
  - ❑ Registrar Usuário
  - ❑ Autenticar Usuário
  - ❑ Realizar Pagamento

Infraestrutura

- ❑ Camada **mais externa** do sistema
- ❑ Implementação dos **detalhes técnicos da aplicação**
- ❑ **Dependência** para dentro
- ❑ Implementação dos **repositórios**
- ❑ **Persistências** dos dados
- ❑ **Fácil** substituição

DTO (Data Transfer Object)

- ❏ **Não possuem** regras de negócio
- ❏ Dados de **comunicação entre os limites arquiteturais**
- ❏ Dados de **Input** e **Outputs** dos **casos de uso**
- ❏ Cada caso de uso possui o seu **DTO** de **input** e seu **DTO** de **output**

Presenters

- Objetos de transformações de dados
- Adequa um **DTO de output** para um **formato específico de entrega**
  - JSON
  - XML
  - GraphQL
  - … entre outros

Exemplo de Presenters

```python
input = CreateUserInputDto(name="João", idade=42)
repository = UserRepository(session=session)
usecase = CreateUserUseCase(repository=repository)
output = usecase.execute(input=input)

# Presenter

# JSON
outputJson = UserPresenter(output).toJson()

# XML
outputXML = UserPresenter(output).toXML()
```

Desafios e Limitações

- ❑ Complexidade **inicial** na adoção
- ❑ Impacto em **times acostumados** com arquiteturas tradicionais
- ❑ Overhead para **pequenas aplicações**

Referências

1. MARTIN, R. C. Arquitetura Limpa. [s.l.] Alta Books Editora, 2019.
2. MARTIN, R. C. Código Limpo. [s.l.] Alta Books, 2019.

OBRIGADO!

linkedin.com/in/helderprado