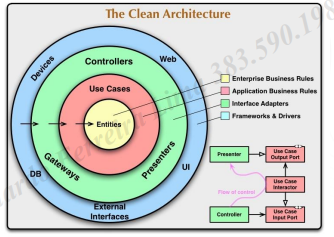
Arquitetura Limpa (Clean Architecture)

Prof. Helder Prado Santos

* A Arquitetura Limpa
  + Termo criado em 2012 por Robert C. Martin.
  + Focada em reduzir o acoplamento entre as camadas arquiteturais.
    - Normalmente dividido em 3 camadas
      * Infra estrutura
      * Use cases
      * Dominio da aplicação
  + Comunicação entre as camadas é realizada a partir de contratos (DTO)
    - Sempre de fora pra dentro nunca de dentro pra fora
  + Orientada aos casos de usos da aplicação.
    - Funcionalidade do sistema
      * Input/output
* 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



* + Enterprise business rules \Dominio da aplicação
    - Entities
      * Aonde vou ter as entidades encapsulando os métodos
      * Também aont se tem as interfaces utilizadas
  + Aplication busines rules
    - Casos de uso acoues ou funcionalidades das entidades
  + Interface adapters
    - Controllers
    - Presenters
      * A forma como quero entregar a apresentação
  + Frameworks e drivers (Mundo externo)
    - Devices
    - Web
    - DB
    - UI
    - External interfaces
  + Fluxo sempre de fora pra dentro
  + 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.
    - São as classes que tem as regras de negocio
    - E o que tem de mais valioso no meu negocio
  + Casos de Uso (Use Cases)
    - Ações e funcionalidades da aplicação
    - Detalhes técnicos **não devem impactar** nos casos de uso
    - Single Responsibility Principle (SRP)
      * **Tem que ser e ter um objetivo único (apenas uma responsabilidade**
      * **Cada caso de uso vai ter o seu input e o seu output**
    - Exemplos: (Se e uma acao eu posso usar um verbo)
      * Registrar Usuário
      * Autenticar Usuário
      * Realizar Pagamento
  + Infraestrutura
    - Camada mais externa do sistema
      * A primeira que eu vou me comunicar
    - Implementação dos **detalhes técnicos da aplicação**
    - **Dependência** para dentro
    - Implementação dos **repositórios**
    - **Persistências** dos dados
    - **Fácil** substituição
  + DTO (Data Transfer Object) – Como nos comunicamos entre as camadas
    - Não possuem regras de negócio
      * Podem apenas ter auto validação básica como de tipo
    - Dados de comunicação entre os limites arquiteturais
    - Dados de Input e Outputs dos casos de uso
      * Todo caso de uso vai ter apenas um método (execute)
        + Esse método vai ter um input e um output
    - 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



* + Desafios e Limitações
    - Complexidade inicial na adoção
    - Impacto em times acostumados com arquiteturas tradicionais
    - Overhead para pequenas aplicações

Fim bloco 1