Arquitetura de Software (A4)

- sistema em camadas
  - criação em camadas tem como função dividir um sistema complexo em partes menores
  - benefícios
    - cada camada tem seu contexto, no qual auxilia na compreensão
    - facilidade de substituição das camadas por implementações alternativas
    - quando uma camada é construída, esta pode ser utilizada por camadas superiores
  - desvantagens
    - alteração em cascata
    - camadas extras prejudicam o desempenho
- arquitetura em camadas - princípios
  - dividir para conquistar: as camadas podem ser projetadas independentemente
  - aumento da coesão: camadas bem definidas
  - redução de acoplamento: camadas bem definidas não conhecem as camadas superiores
  - aumento de abstração: não é necessário conhecer em detalhes a implementação de camadas inferiores
  - aumento da reusabilidade: camadas inferiores podem ser projetadas de forma genérica
  - flexibilidade: adição ou modificação de funcionalidades
  - portabilidade: funcionalidades dependentes podem ser isoladas em camadas de mais baixo nível
  - testabilidade: camadas podem ser testadas independentemente
  - design defensivo: adição de verificações rigorosas de permissões e segurança
- arquitetura em 3 camadas
  - apresentação: interação usuário-software
    - exibir informações para o domínio
    - traduzir comandos de usuário em ações sobre o domínio e a camada de dados
  - camada de dados: comunicação com outros sistemas que executam tarefas no interesse da aplicação
  - lógica de domínio: trabalho que a aplicação tem de fazer para o domínio
  - princípios
    - evitar implementação de lógicas de negócio complexas
    - evitar acesso direto aos dados na camada de apresentação

        - camada de apresentação é responsável pela interação com o usuário
        - camada de apresentação comunica-se diretamente com a lógica de domínio
    - vantagens
        - evoluir, acrescentar ou trocar completamente a camada de apresentação sem necessitar de alterações nas camadas inferiores
        - na camada de dados existe a liberdade de escolha dos mecanismos de persistência
        - manutenção simplificada devido as operações encapsuladas e de responsabilidade distintas
    - desvantagens
        - complexidade alta no desenvolvimento inicial
        - redução do desempenho em operações mais simples, quando operações entre camadas transmitem dados em formatos diferentes
        - divisão de responsabilidades de cada camada é um fator de dificuldade na compreensão durante o desenvolvimento
- arquiteturas comuns
    - padrão Broker: distribuição transparente de aspectos da aplicação por diferentes nós
    - padrão transaction-processing: um processo interpreta várias entradas individualmente de modo que cada entrada represente uma transação, ou seja, estas entradas representam um comando que realizar um cojunto de mudanças nos dados armazenado pela aplicação
    - padrão MVC: auxiliar na separação da camada de interface com o usuário das demais partes da aplicação
        - model: instâncias de classes que devem ser visualizadas e manipuladas
        - view: renderização da aparência dos dados
        - controller: controle e tratamento das interações do usuário com a view e a model
    - padrão orientado a serviços: organiza a aplicação como coleções de serviçoes que comunicam-se através de interfaces bem definidas
        - APIs
        - web services
    - padrão orientado a mensagens (MOM ou Message-Oriented Middleware): comunicação e colaboração entre diferentes subsistemas para realizaçõ de determinada tarefa através de mensagens