Resumo Arquitetura de Software💻

✅ O que é Arquitetura de Software?

É a **estrutura organizacional do sistema**, composta pelos **componentes principais**, suas **interfaces visíveis** e os **relacionamentos entre eles**. Ela serve como **espinha dorsal** para a construção de sistemas eficazes.

Por que é importante?

- Permite **comunicação entre os envolvidos** (stakeholders).
- Ajuda na **tomada de decisões** desde o início do projeto.
- **Reutilizável** em outros sistemas semelhantes.

🧩 Conceitos Fundamentais

🔹 Coesão

- Mede o quão bem as **responsabilidades de um módulo estão relacionadas**.
- **Alta coesão = Módulos mais simples, reutilizáveis e fáceis de manter.**

Exemplo: Um módulo que só faz cálculo de impostos tem alta coesão. Já um que faz login, cálculo e envio de e-mail, não.

🔹 Acoplamento

- Mede a **dependência entre módulos**.
- **Baixo acoplamento = menos impacto entre módulos, mais flexibilidade.**

Exemplo: Módulos que se comunicam só por parâmetros têm baixo acoplamento.

🏛️ Tipos de Arquitetura

1. Monolítica

- Tudo em uma única aplicação.
- Dificulta manutenção e escalabilidade.

2. Duas camadas (Two-tier)

- Cliente executa a interface e lógica da aplicação (Fat Client).
- Servidor apenas armazena os dados (SGBD).

3. Três camadas (Three-tier)

- Separação entre:
    - **Apresentação** (interface com o usuário)
    - **Negócio** (regras e lógica da aplicação)
    - **Acesso a Dados** (banco de dados)
- Muito utilizada por facilitar manutenção e segurança.

4. N-Camadas

- Extensão do modelo de três camadas.
- Divide responsabilidades em **várias camadas**, conforme a complexidade do sistema.

📱 Arquitetura MVC (Model-View-Controller)

Muito usada no desenvolvimento web e desktop. Tem **3 camadas**:

- **Modelo**: gerencia dados e regras de negócio.
- **Visão**: interface com o usuário (HTML, XML, etc.).
- **Controlador**: coordena ações entre visão e modelo.

> **Diferente da arquitetura 3-tier**, o MVC tem comunicação **triangular**, onde a **visão pode acessar diretamente o modelo**.

🌐 Arquitetura Web

- O **navegador (browser)** é o cliente universal.
- Reduz a necessidade de instalar software localmente.
- Pode seguir modelo de 3 ou mais camadas.

🌍 Arquitetura Distribuída

- Vários computadores trabalham **juntos como um único sistema**.
- Cada parte (cliente, servidor, banco de dados) pode estar em lugares diferentes.

Vantagens:

- Escalabilidade, tolerância a falhas, compartilhamento de recursos.

Desafios:

- Concorrência, segurança, complexidade, interoperabilidade.

🧩 Middleware

- **"Cola" entre sistemas diferentes.**
- Permite que aplicações em linguagens/plataformas distintas **comuniquem-se**.
- Exemplos: **CORBA**, **RMI (Java)**, **RPC**.

🧱 Arquitetura de Microsserviços

- Sistema dividido em **vários pequenos serviços independentes**.
- Cada um tem sua própria lógica, banco de dados, e pode ser implementado em **linguagens diferentes**.

Vantagens:

- Deploys independentes, escalabilidade, foco em partes específicas do negócio.

Comparação com Arquitetura Tradicional:

Tradicional	Microsserviços
Aplicação única	Aplicações fragmentadas
Deploys acoplados	Deploys independentes
Uma linguagem	Linguagens variadas

🏢 Arquitetura Mainframe

- Computadores de **grande porte**.
- Centraliza tudo: dados, aplicativos, interface.
- **Alta performance e segurança**, mas custo e manutenção elevados.

- Usado em bancos, órgãos públicos, etc.

🔁 Arquitetura P2P (Peer-to-Peer)

- **Todos os computadores são iguais**.
- Sem servidor central — cada máquina pode ser cliente e servidor.
- Usado em torrents, Bitcoin, etc.