Desmistificando Arquitetura de Software

...

TDC Transformation - Trilha Design de Código e XP - 26/08/2021

Quem sou eu?

Haldny Santos

Engenheiro Android - XP Inc

Desenvolvimento Android +7 Anos

Desenvolvimento de Software +11 Anos

Projeto Compartilhar

Onde me encontrar?

Agenda

- O Início da Arquitetura de Software
- Estilos Arquiteturais
- Arquiteturas de Apresentação
- Arquiteturas de Comunicação
- Design Patterns
- Conclusão

O Início da Arquitetura de Software

- Definição de Arquitetura
- Definição de Arquitetura de Software
- Como iniciou a Arquitetura de Software?
- Porquê pensar em Arquitetura desde o Início do Software?

Definição de Arquitetura

"Essa arte é composta pelo conjunto dos princípios, normas, técnicas e materiais, para criar um espaço arquitetônico."

Fonte: https://www.significados.com.br/arquitetura/

Definição de Arquitetura de Software

"Conjunto dos princípios, normas, técnicas e materiais, para criar um Software."

Fonte: https://www.youtube.com/watch?v=kYx1QC1XZSo

Como iniciou a Arquitetura de Software?

- Edsger W. Dijkstra (1968)

- David Parnas (Início dos Anos 70)

Porquê pensar em Arquitetura desde o Início do Software?

- Produtividade

Porquê pensar em Arquitetura desde o Início do Software?

- Objetivo de negócio
- Atributos de Qualidade
- Restrições de Alto Nível

Porquê pensar em Arquitetura desde o Início do Software?

Componentes +

Responsabilidades +

Relacionamentos +

Estratégia

Estilos Arquiteturais

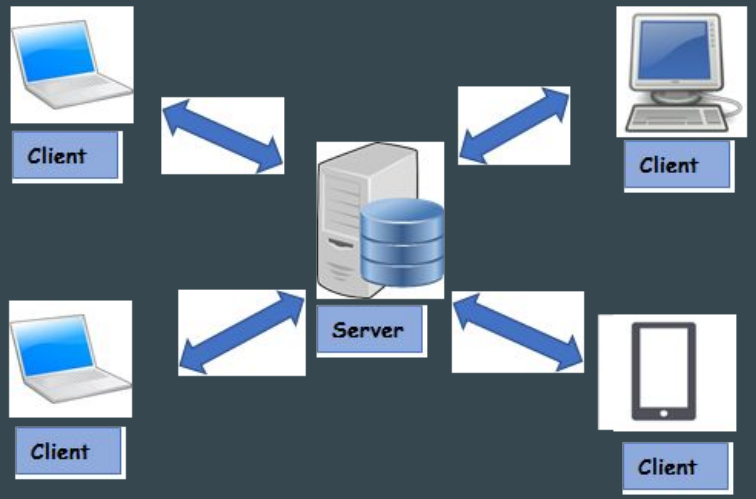
- Layers (Camadas)
- Client-Server (Cliente-Servidor)
- Microservices (microsserviços)
- Arquitetura baseada em Componentes
- Peer-to-Peer (P2P)
- Service-Oriented Architecture (SOA)
- Clean Architecture (Arquitetura Limpa)

Layers (Camadas)

Os módulos e componentes do software são organizados em camadas de funcionalidades, que podem ser desconstruídas em diferentes serviços.

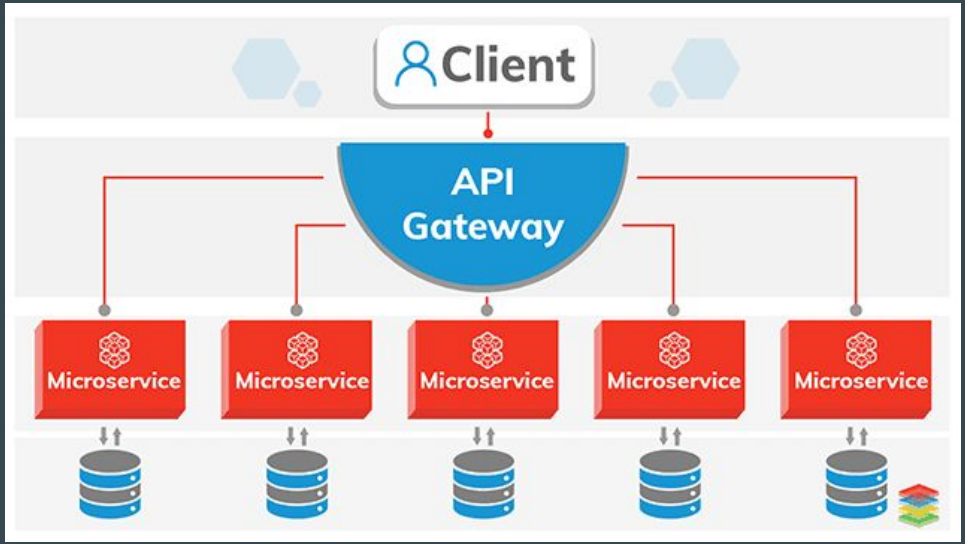
Client-Server (Cliente-Servidor)

Segrega o sistema em duas aplicação, onde o cliente faz uma requisição de serviço ao servidor.



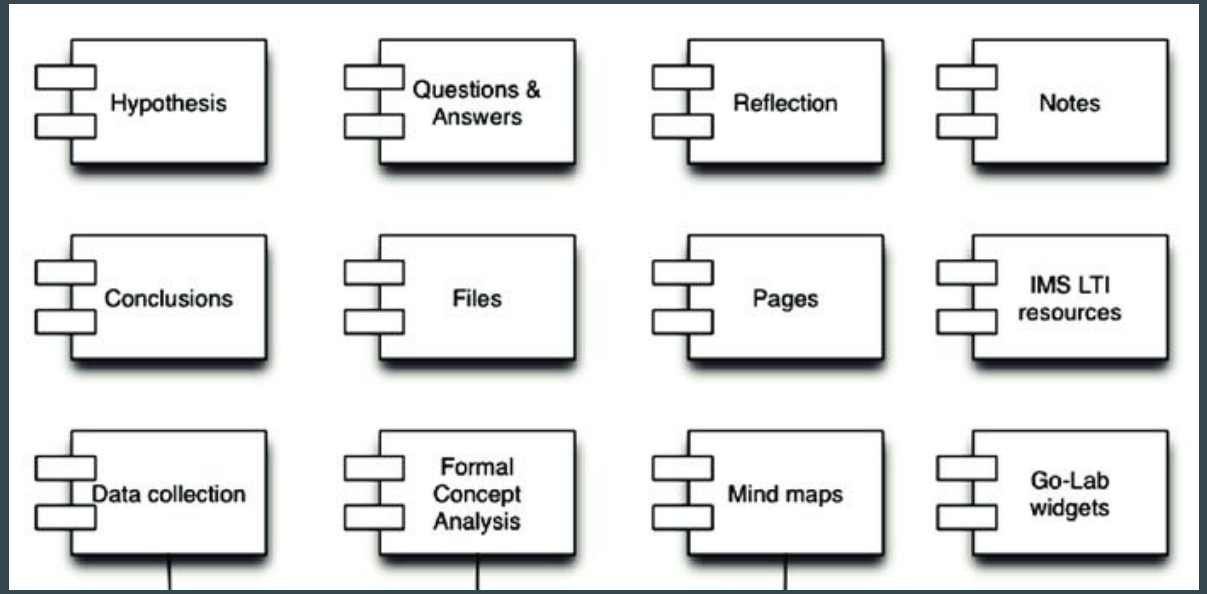
Microservices (microsserviços)

O padrão se baseia em múltiplos serviços e componentes para desenvolver uma estrutura modular. Permite escalabilidade e independência dos módulos, que podem usar diferentes linguagens.



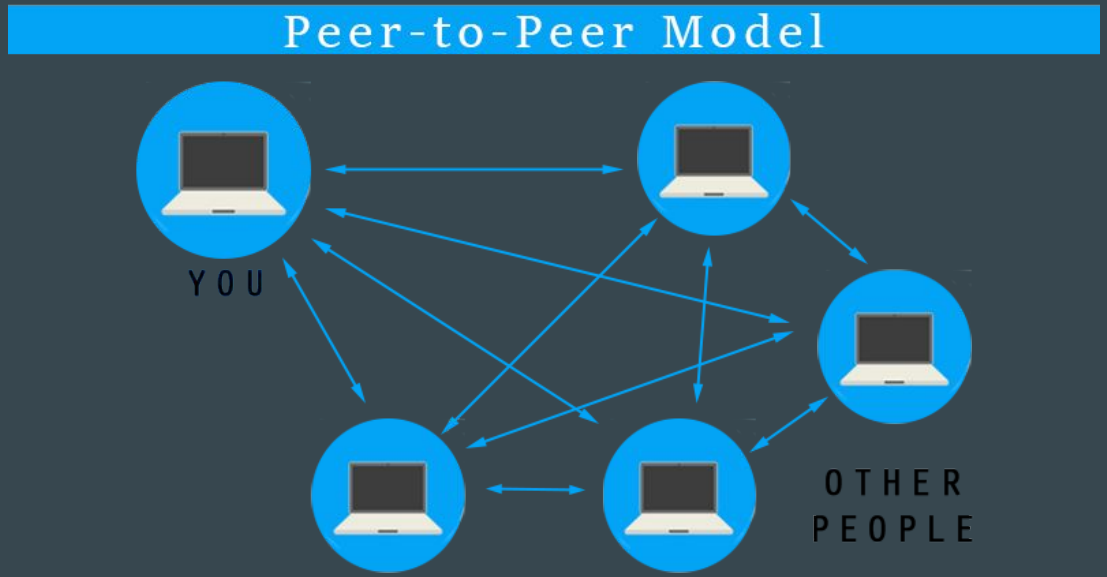
Arquitetura baseada em Componente

Decompõe o design da aplicação em componentes lógicos e funcionais que são independentes de local e expõe interfaces de comunicação bem definidas.



Peer-to-Peer (P2P)

No Peer-to-Peer, todos os pares são clientes e servidores, ou seja, cada computador é um provedor de serviços independente de um servidor central.

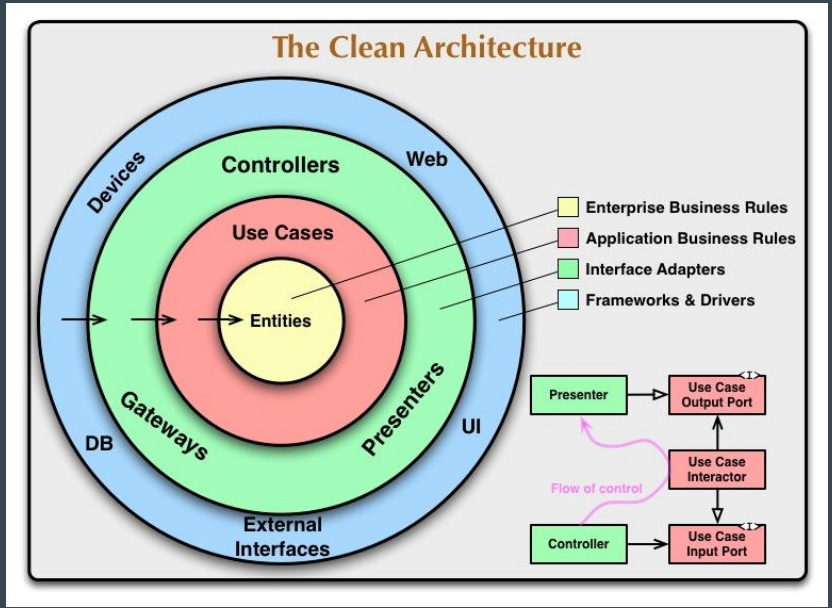


Service-Oriented Architecture (SOA)

Refere-se a aplicações que expõe e consome funcionalidades como um serviço usando contratos e mensagens no nível Enterprise.

Clean Architecture (Arquitetura Limpa)

No Clean Architecture as camadas do mundo externo (Banco de Dados, WEB, UI) são separadas das camadas mais atreladas ao negócio.



Categorias

Categoria	Estilos de Arquitetura
Comunicação	Service-Oriented Architecture (SOA), Message Bus, Pipes e Filtros
Deployment	Client/server, 3-Tier, N-Tier
Domain	Domain Model, Gateway
Interação	Apresentação Separada
Estrutura	Baseado em Componentes, Orientado a Objetos, Arquitetura em Camadas

Qual o melhor estilo arquitetural para o meu software?

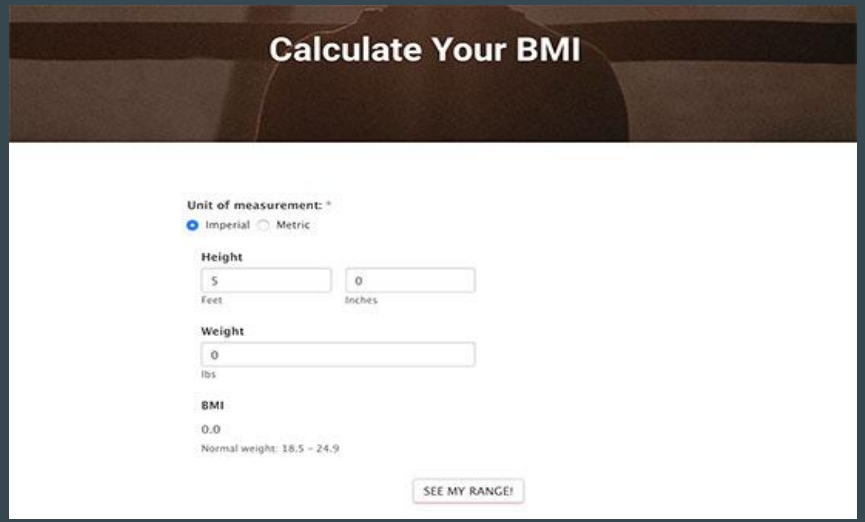
Se você está em busca de uma resposta exata para esta pergunta, saiba que ela não existe. Cada projeto é único e pode combinar diferentes tipos de estilos arquiteturais de software, dependendo do objetivo do sistema e do problema que precisa ser solucionado. O que ajudará nesta definição é a experiência do profissional e o conhecimento dos mais diferentes cases – positivos e negativos.

Arquiteturas de Apresentação

- Forms and Controls
- Model-View-Controller
- Model-View-Presenter
- Model-View-ViewModel
- Model-View-Intent

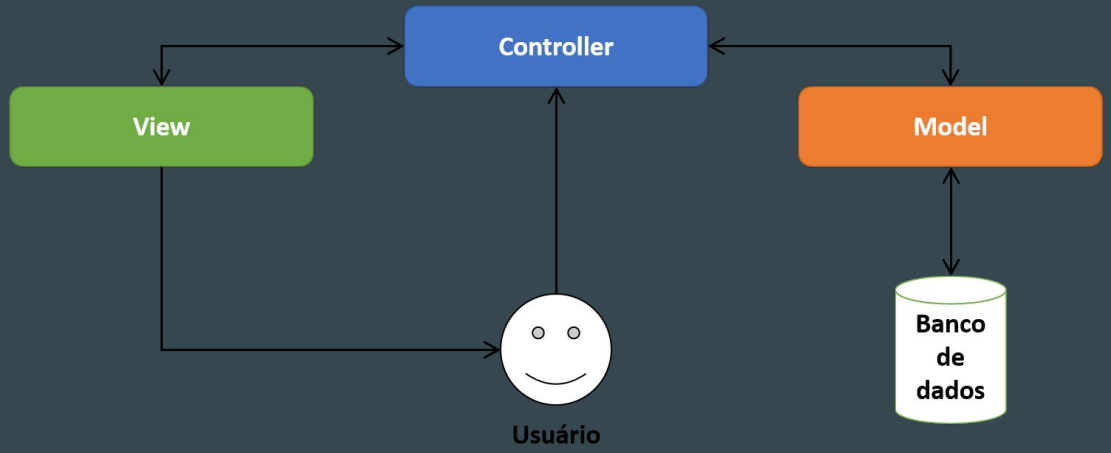
Forms and Controls

Modelo onde apenas temos um formulário e os controles para executar as ações, ou seja, todo o processamento é executado e exibido para o cliente através dessa interação.



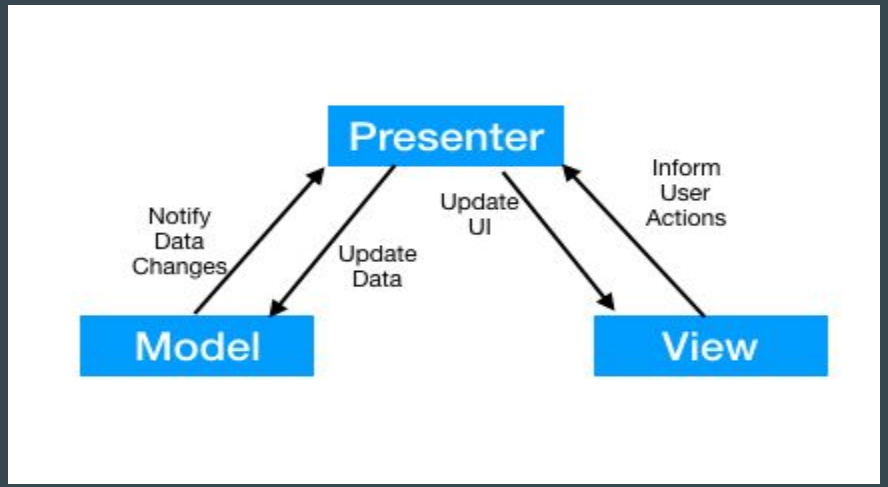
Model-View-Controller

O padrão MVC separa o projeto do software em três camadas independentes: o modelo (manipulação da lógica de dados), a visão (a interface do usuário) e o controlador (fluxo de aplicação).



Model-View-Presenter

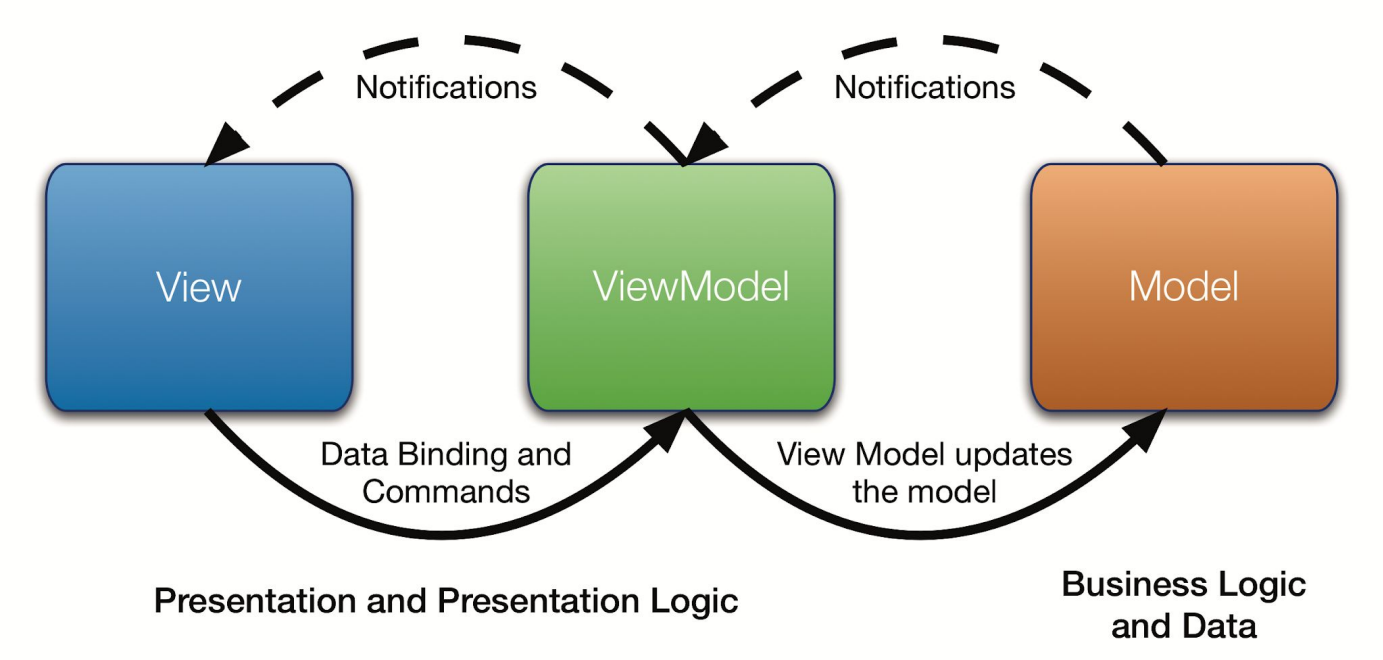
O Padrão MVP separa o projeto do software em três camadas independentes: o modelo (manipulação da lógica de dados), a visão (a interface do usuário) e o apresentador (atua como uma ponte que conecta o modelo a visão).



Model-View-ViewModel

O padrão MVVM separa o projeto do software em três camadas independentes: modelo (manipulação da lógica de dados e notificação da mudança de dados), a visão (a interface do usuário e código para mostrar os estados disponibilizados nos ViewModels) e os ViewModels que contém lógica de apresentação para recuperar dados de modelos e delegar a execução de lógica de negócios para seus modelos.

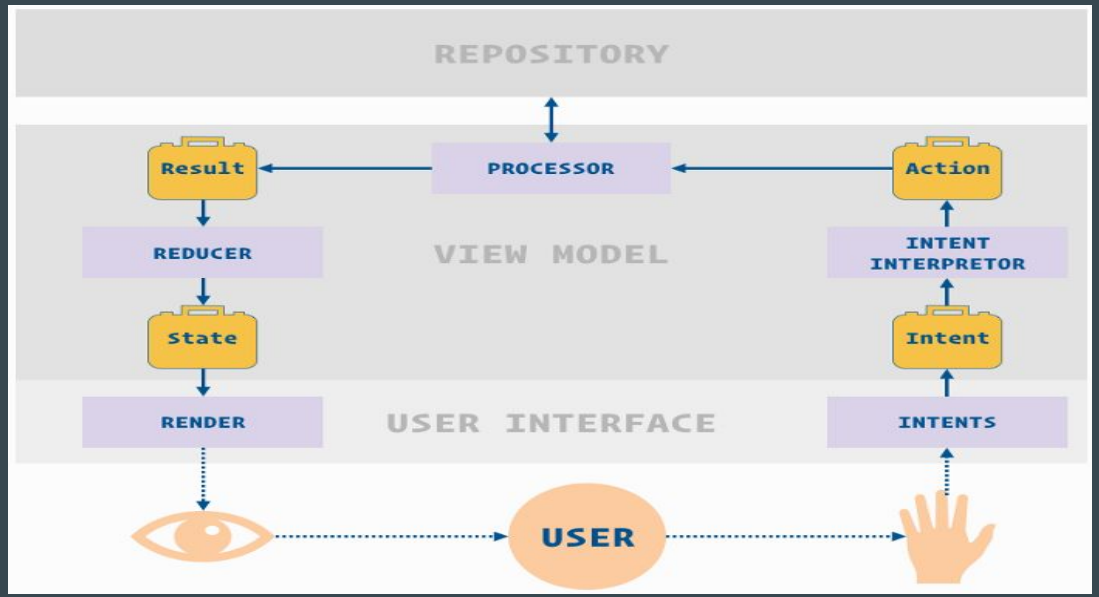
Model-View-ViewModel



Model-View-Intent

O padrão MVI separa o projeto do software em três camadas independentes: o modelo (que representa um estado imutáveis para garantir um fluxo de dados unidirecional entre eles e as outras camadas em sua arquitetura), a visão (que observa a intenção e os modelos e traduz para um novo estado) e as intenções (que representa uma intenção ou desejo de realizar uma ação, seja pelo usuário ou pelo próprio aplicativo).

Model-View-Intent

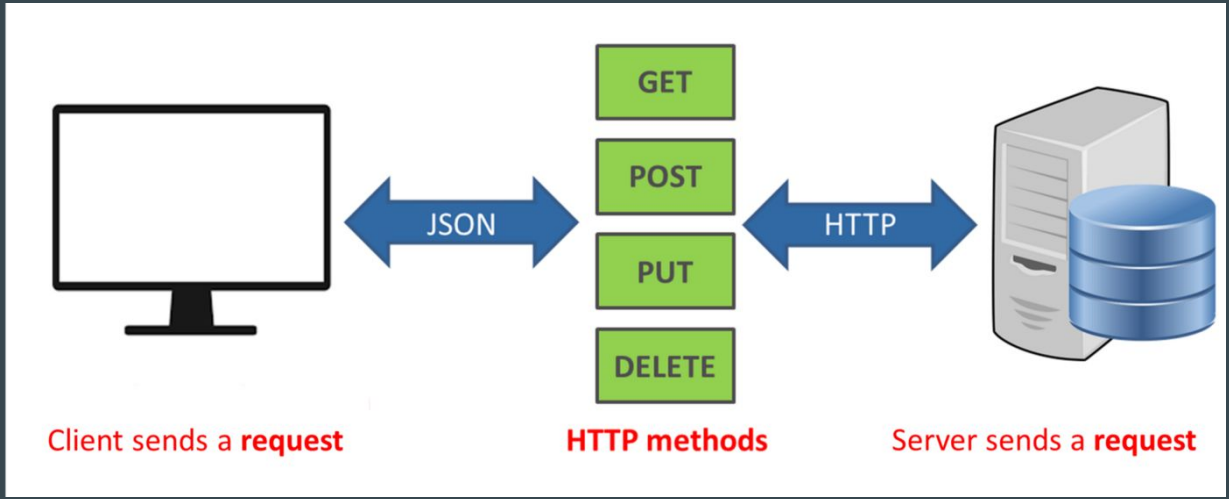


Arquiteturas de Comunicação

- REST
- SOAP
- Binder
- RPC e gRPC

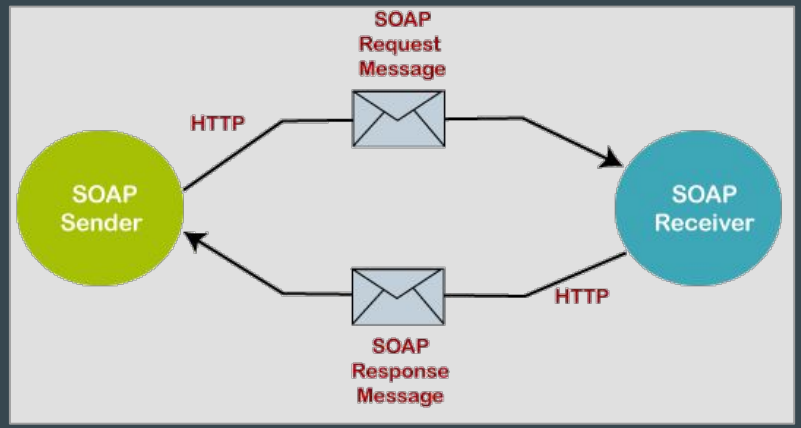
REST

Concebido como uma abstração da arquitetura da web, trata-se de um conjunto de princípios e definições necessários para a criação de um projeto com interfaces bem definidas.



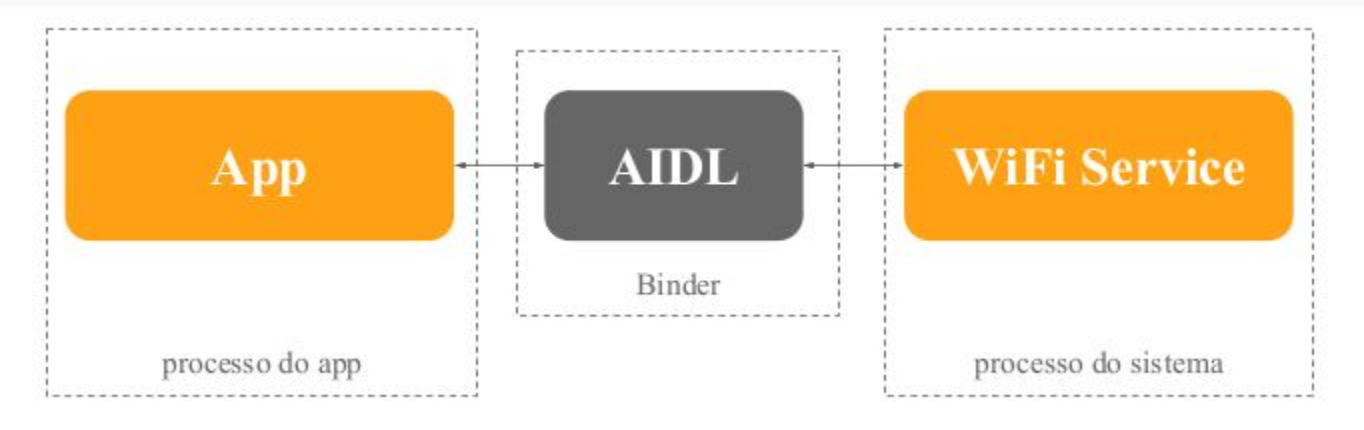
SOAP

Ele usa o formato XML para transferir mensagens pelo protocolo HTTP. Em serviços da Web, o SOAP permite que a solicitação do usuário interaja com outras linguagens de programação e fornece uma forma de comunicação entre aplicações rodando em diferentes plataformas.



Binder

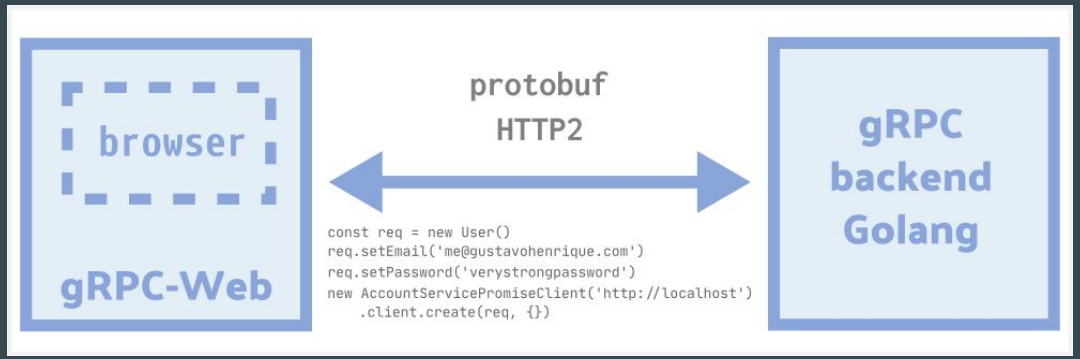
É o modelo de comunicação e invocação de métodos entre processos no Android, sendo chamadas internas da aplicação ou chamadas externas através de system services ou componentes "bounded".



RPC e gRPC

O RPC é um modelo client-server. O solicitante é um cliente e quem fornece a informação é um serviço. O RPC é uma operação síncrona e exige que o cliente seja suspenso até que o resultado do Server seja retornado.

O Google é o criador do gRPC. O objetivo era conectar os Microsserviços em seus datacenters.



Design Patterns

Design patterns são **modelos** que já foram **utilizados** e **testados** anteriormente, portanto podem representar um bom **ganho de produtividade** para os **desenvolvedores**.

Seu uso também contribui para a **organização** e **manutenção** de projetos, já que esses padrões se baseiam em **baixo acoplamento** entre as classes e **padronização** do código.

Design Patterns mais conhecidos

Os autores do livro “**Design Patterns: Elements of Reusable Object-Oriented Software**” (**Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides**) agruparam os Design Patterns em três tipos diferentes: **Creational (Criação)**, **Structural (Estrutura)**, **Behavioral (Comportamental)**.

Design Patterns Creational (Criação)

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Design Patterns Structural (Estrutura)

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

Design Patterns Behavioral (Comportamental)

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

Design Patterns (Outros)

- Rules Design Patterns
- Dependency Injection
- Intercepting filter
- Lazy loading
- Mock object
- Method chaining
- Inversion of control
- Unit of Work
- Repository
- DAO

Conclusão

A arquitetura do Software não é um **fotografia comum** para todos os interessados, você deve pensar no todo para traçar a estratégia de evolução, mas, também se preocupar nas **fotografias de grupos** menores. **Na arquitetura, cada parte tem sua planta ;)**

Referências

- https://posdigital.pucpr.br/blog/tipos-de-arquitetura-de-software
- https://www.devmedia.com.br/arquitetura-de-software-desenvolvimento-orientado-para-arquitetura/8033
- https://martinfowler.com/eaaDev/uiArchs.html
- https://medium.com/luizalabs/descomplicando-a-clean-architecture-cf4dfc4a1ac6
- https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel#:~:text=Model%E2%80%93view%E2%80%93viewmodel%20(MVVM,is%20not%20dependent%20on%20any
- https://www.infoq.com/br/news/2009/02/Architectural-Styles-Patterns/
- https://www.raywenderlich.com/817602-mvi-architecture-for-android-tutorial-getting-started

Obrigado!