POO2

Padrões Arquiteturais

Arquitetura de Software

- Já discutimos o que é um **padrão** quando falamos de padrões de projeto

E padrão de arquitetura?

- Organização macroscópica
- Escopo mais amplo que design patterns
- Estrutura fundamental do código
- Princípios de organização mais “alto-nível”
- Nível de componente, não de classe

Arquitetura de Software

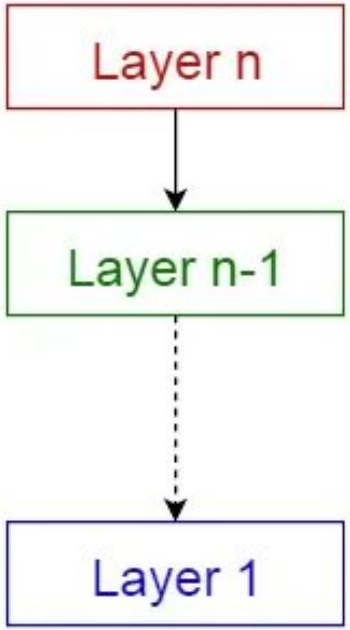
- **Por que usar?**
  - Separação de responsabilidades
  - Divisão de trabalho
  - Legibilidade
  - Reusabilidade

Padrão de camadas

- Layered Architecture
- Divisão em termos de nível de abstração
- Cada camada oferece serviços para a camada seguinte
- Decompõe projeto em sub-tarefas

Padrão de camadas

- Divisão em termos de nível de abstração
- Cada camada oferece serviços para a camada seguinte
- Decompõe projeto em sub-tarefas



Padrão de camadas

Divisão típica de camadas

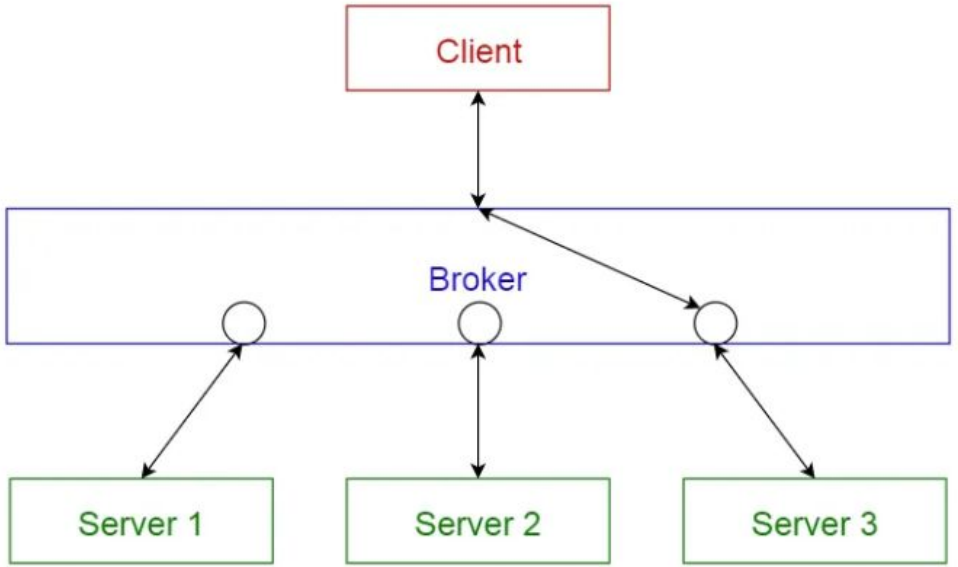
- **Presentation -** Apresentação, cuida da UI
- **Application -** Aplicação, também chamada de camada de serviço. Cuida da aplicação (aplicação X domínio, lembra?)
- **Business logic** - Lógica de negócio ou Domínio. Realiza operações e guarda detalhes do domínio da aplicação
- **Data access** - Camada de persistência, realiza comunicação com outros sistemas (geralmente base de dados)

Padrão Broker

- Broker = **Corretor**
- Padrão para sistemas distribuídos
- Componentes interagem por chamadas remotas
- Um dos componentes é o Broker que coordena a comunicação dos componentes remotos

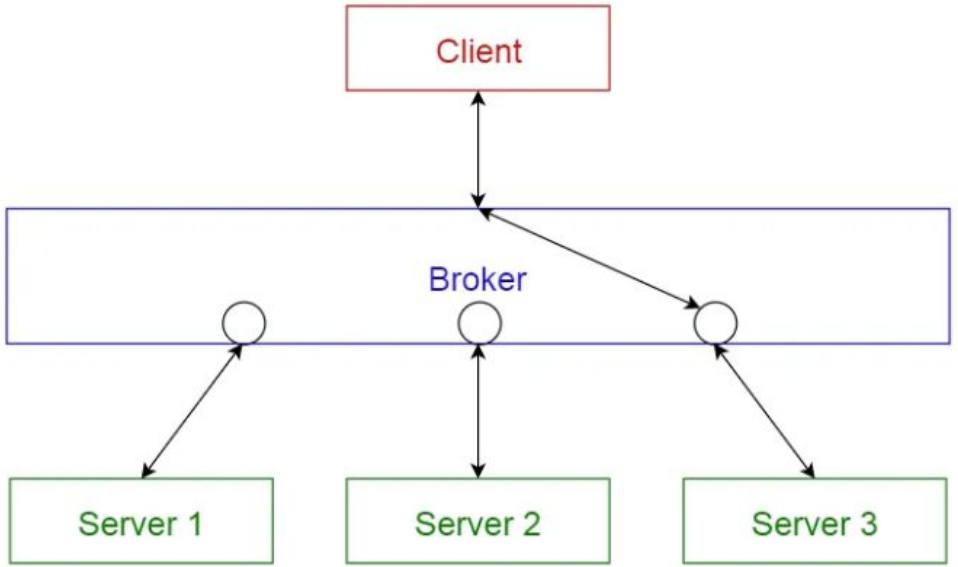
Padrão Broker

- Servidores publicam seus serviços para o Broker
- Clientes solicitam serviços do Broker, que redireciona para um serviço adequado
- Analogia com quais padrões de projeto?



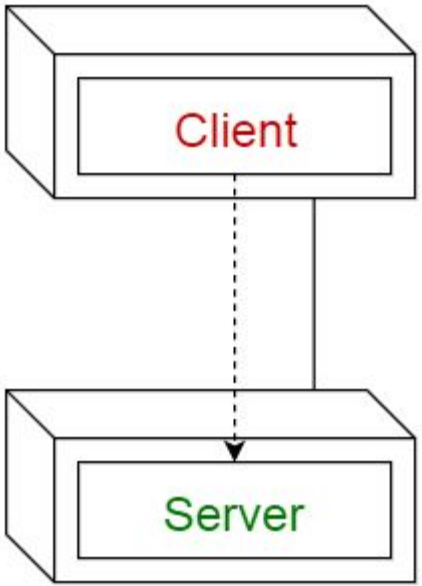
Padrão Broker

- Servidores publicam seus serviços para o Broker
- Clientes solicitam serviços do Broker, que redireciona para um serviço adequado
- Analogia com quais padrões de projeto? **Mediator** e **Façade**



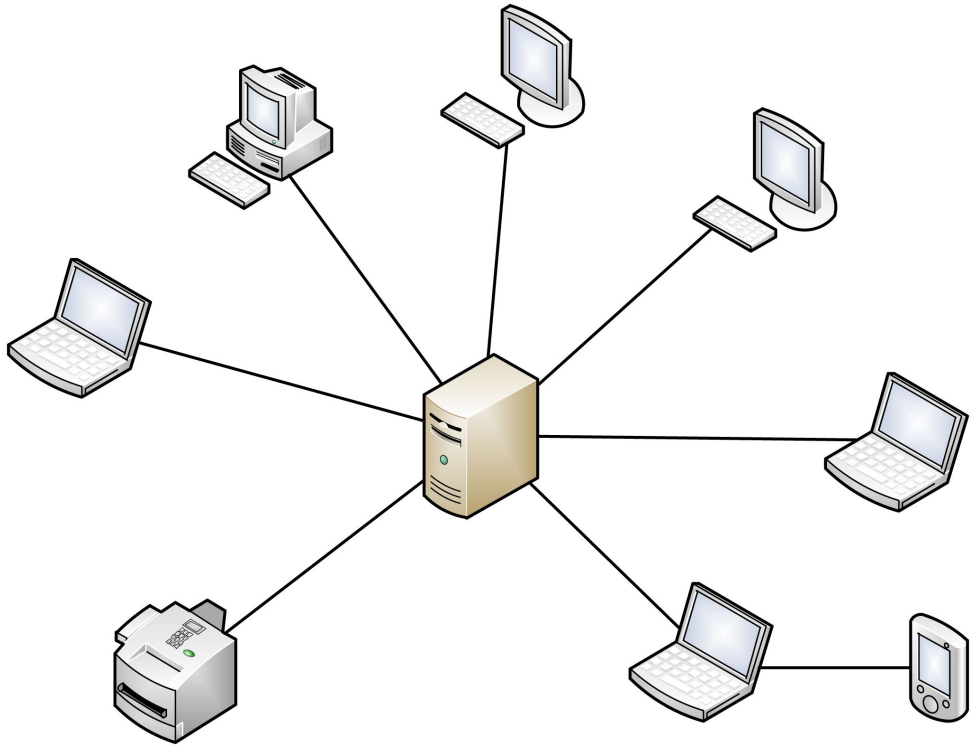
Padrão Cliente-Servidor

- Se sua aplicação tem duas partes cliente-servidor bem definidas
- Normalmente N para 1..N
- Comum em aplicações Web/aplicações na rede



Padrão Cliente-Servidor

- Se sua aplicação tem duas partes cliente-servidor bem definidas
- Normalmente N para 1..N
- Comum em aplicações Web/aplicações na rede

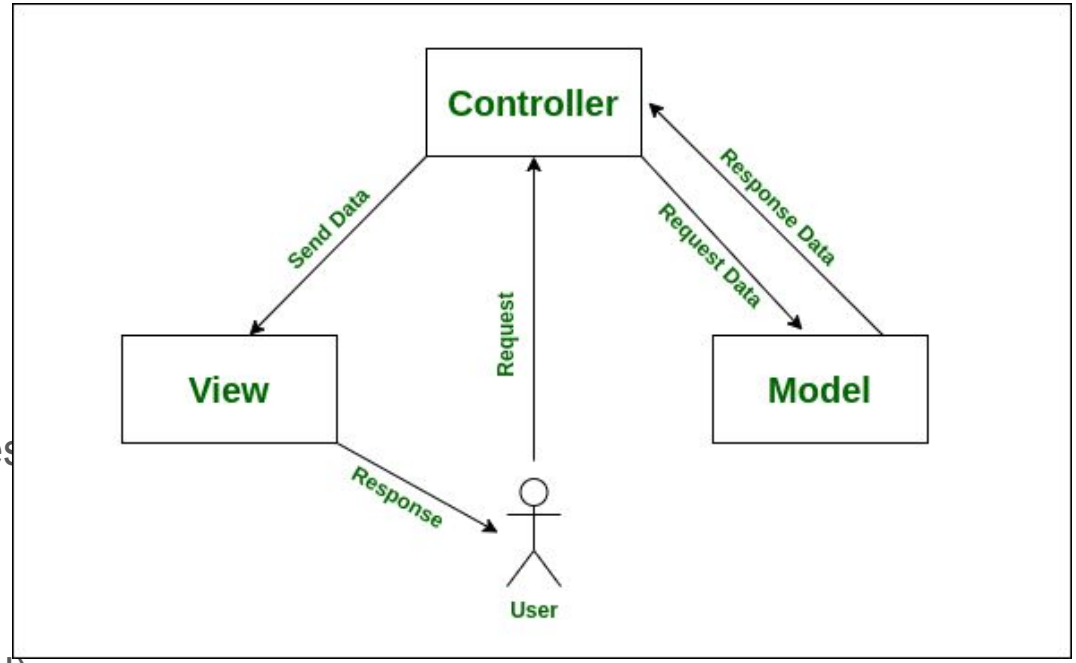


Arquitetura MVC

Model View Controller

Uma das mais populares

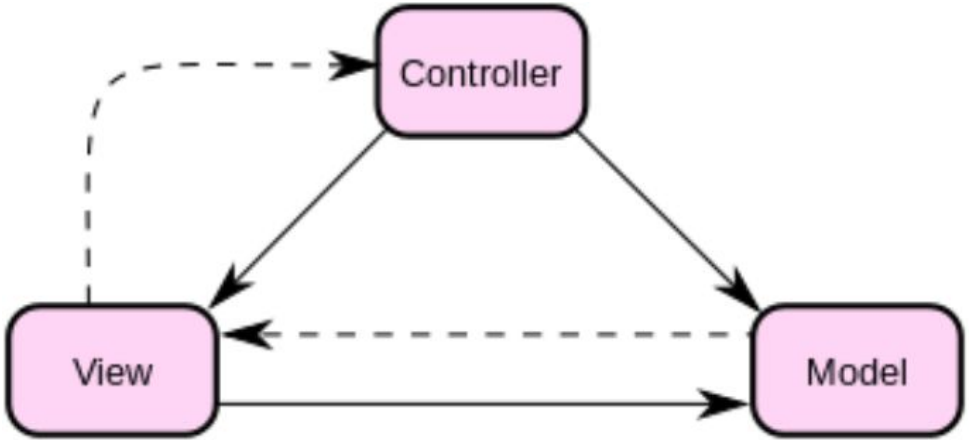
- Organização com baixo acoplamento
- Fácil de modificar
- Permite divisão de especialidades (como todos os padrões)
- Simplifica TDD, teste de unidade/debug
- Permite múltiplas Views (frontend)
- Model não pre preocupa com View



Arquitetura MVC

Model View Controller

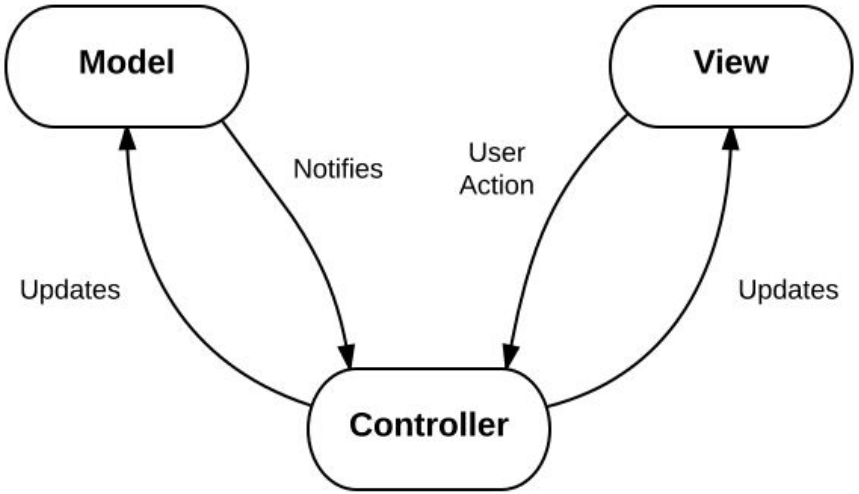
- Vamos dedicar um tempo a mais aqui para discutir cada camada



Arquitetura MVC

View

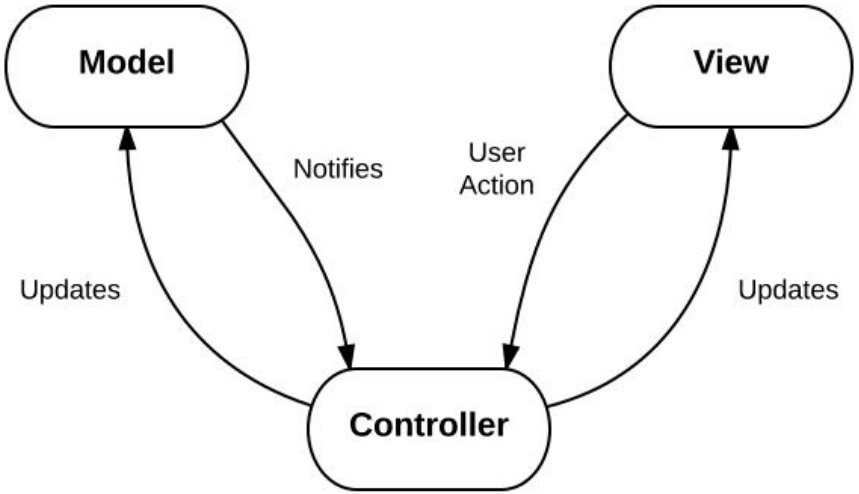
- Define a apresentação da aplicação
- A View recebe dados do Modelo e os apresenta de uma forma amigável ao usuário



Arquitetura MVC

Controller

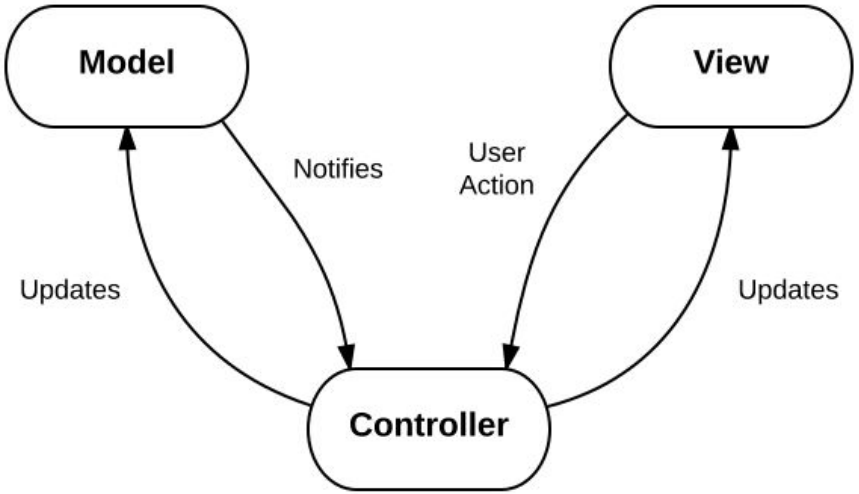
- Intermedia **View** e **Model**
- Envia requisições do usuário ao Modelo, atualizando-o conforme necessário
- Define que View deve ser exibida
- Coordena fluxo de dados para cumprir cada caso de uso



Arquitetura MVC

Model

- Classes de domínio/entidade
- Contém os dados e regras de negócio
- Gerencia os dados e aplica regras de negócio para processar os dados
- Não enxerga a View

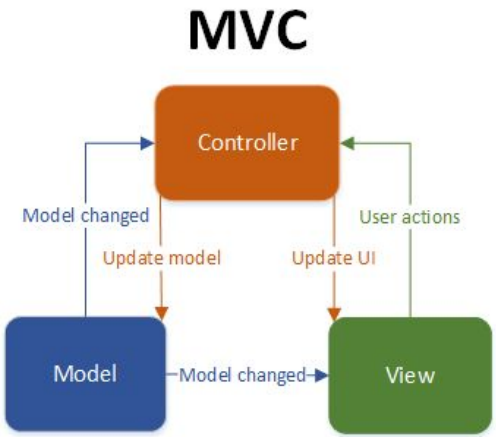


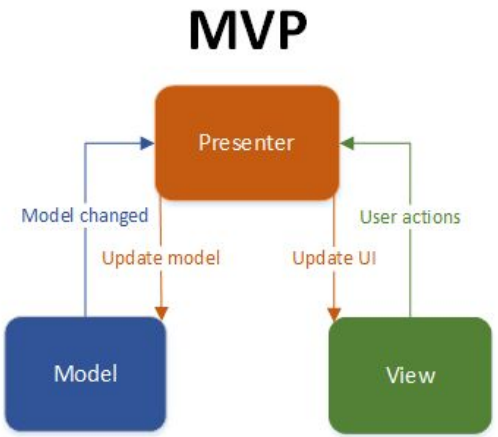
Arquitetura MVC

Padrões similares

Model-View-Presenter

- Reduz a responsabilidade da View de ler o Model
- Mas View trata eventos de UI
- O Presenter formata e oferece os dados para a View



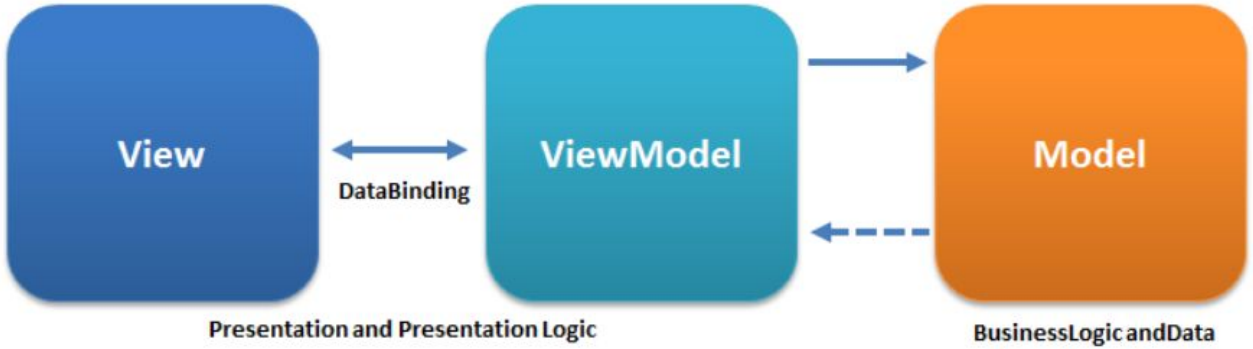


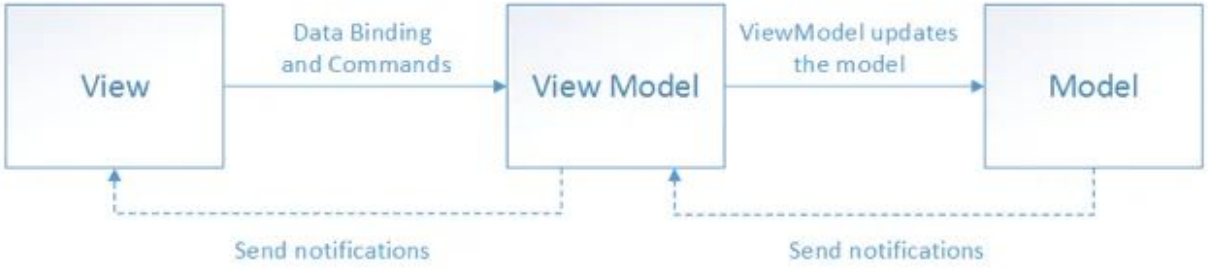
Arquitetura MVC

Padrões similares

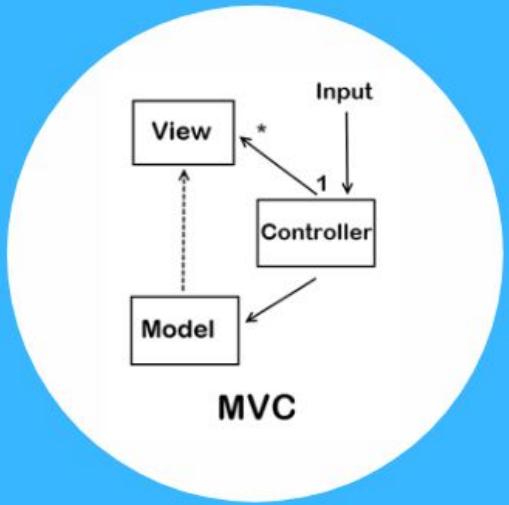
Model-View-ViewModel

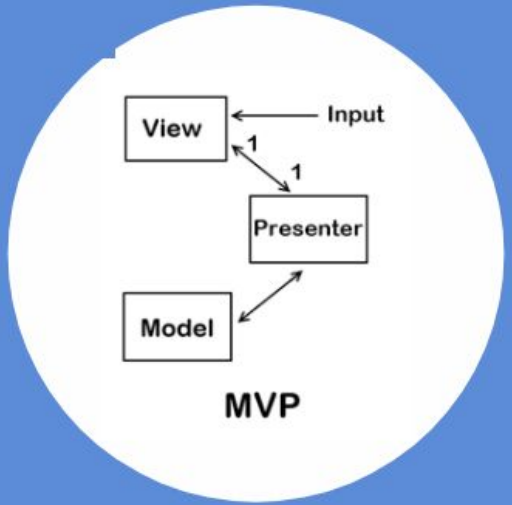
- Ao invés de um Controller, temos um ViewModel que faz o casamento entre View e Model
- VM oferece uma projeção dos dados pra View
- VM mais acoplado à View
- Muito usado em desenvolvimento Android

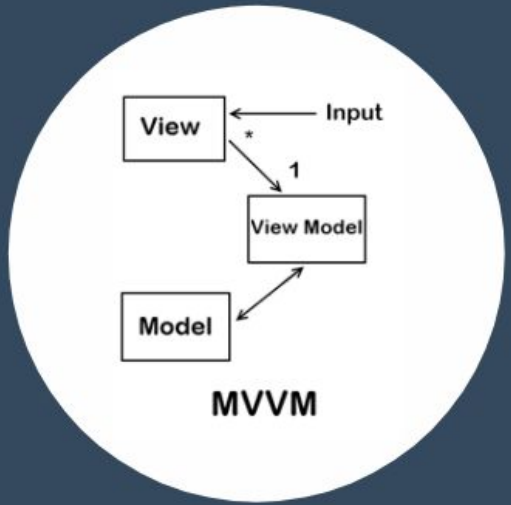




MVC VS MVP VS MVVM







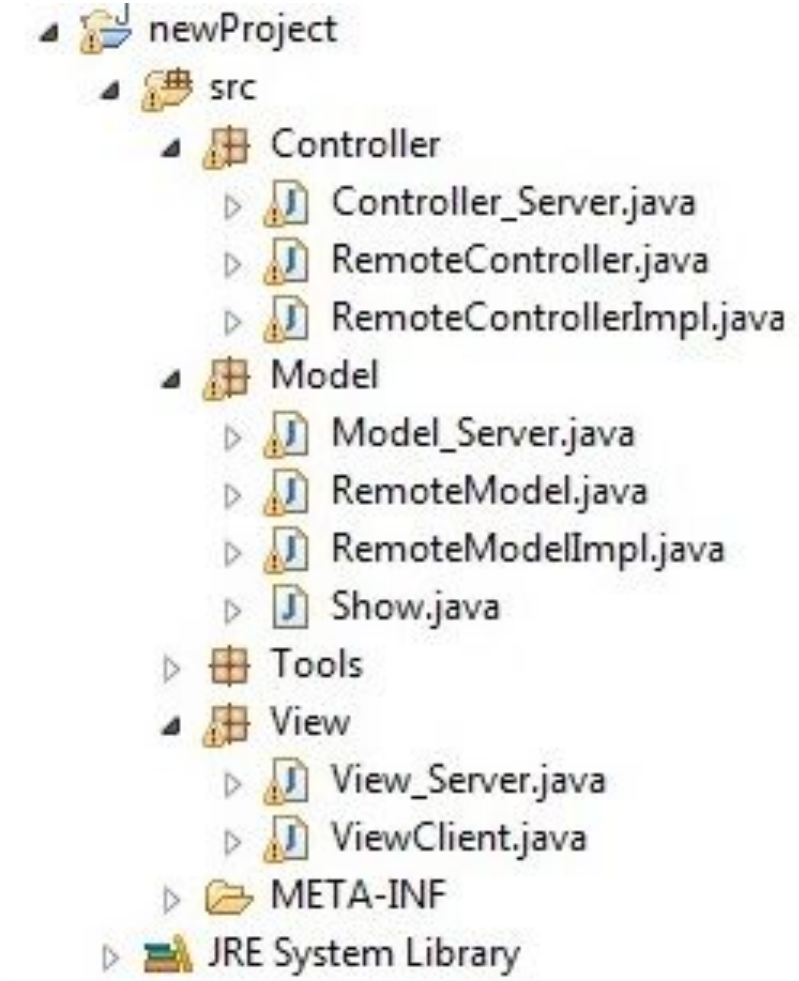
Arquitetura MVC

- Em geral **MVC** tem mais flexibilidade para trocar Views
  - View desacoplada do Controller
  - Input vai no Controller
  - View lê direto do Model
- No **MVP,** View é mapeada 1 pra 1 com o Presenter
  - Input na View
  - View tem referência pro Presenter
  - View não enxerga Model
- **MVVM** é parecido com MVP, mas
  - View enxerga VM, mas VM não enxerga View
  - VM funciona como Model para a View

Arquitetura MVC

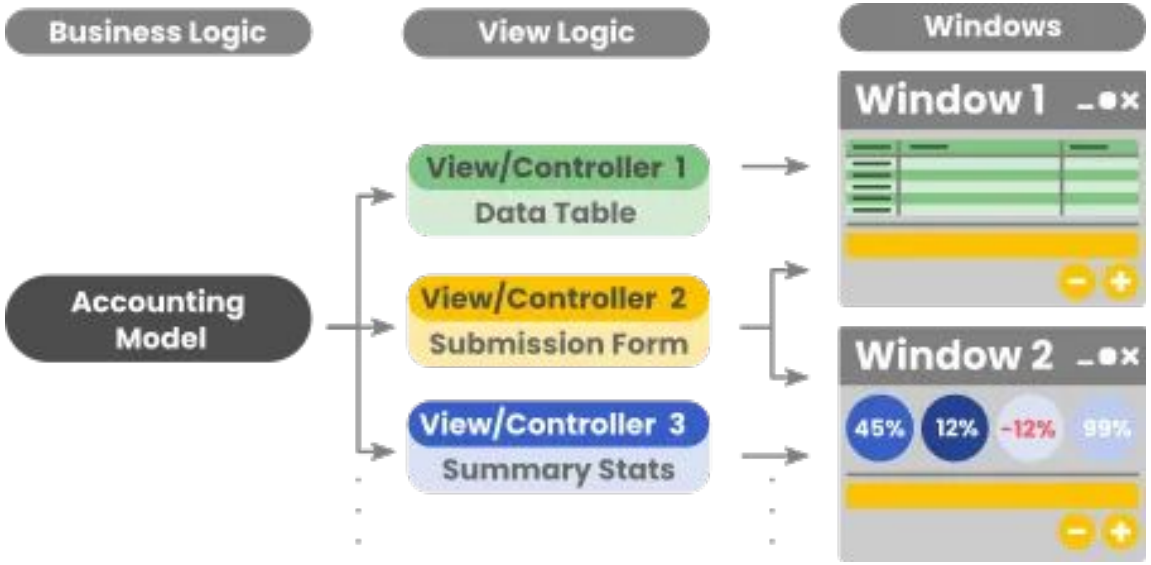
Como fica no Java?

- Podemos discriminar os componentes da arquitetura com pacotes/packages



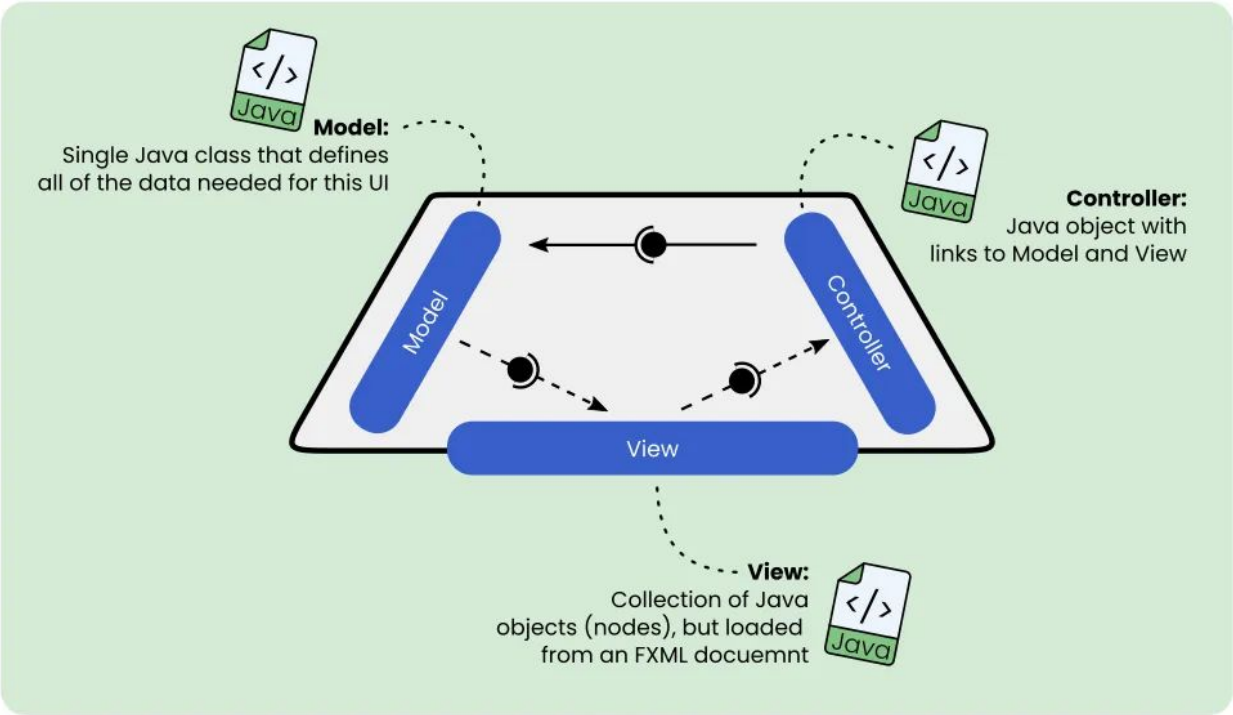
Arquitetura MVC

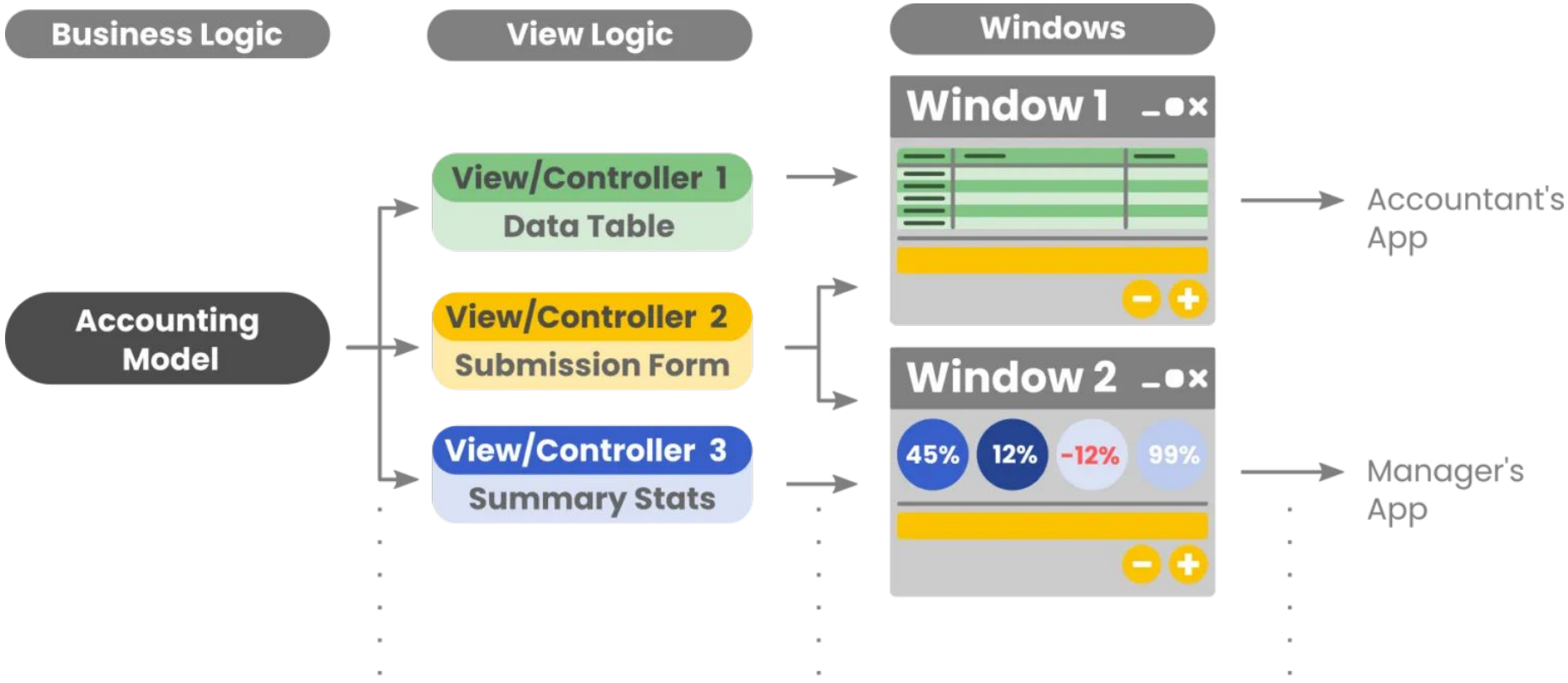
Como fica no JavaFX?



Arquitetura MVC

Como fica no JavaFX?





Single **Model** reused across multiple **Views**

View and Controller reusable across multiple **Windows**

Multiple user groups satisfied with no modification of code