Feb 25, 2016 · 3 min read

Desmistificando o MVC e MVP no Android

Ao longo dos anos tenho trabalhado em diversos projetos e muitas vezes entramos em debates com as equipes para decidir quais seriam as melhores práticas e padrões de projeto para um determinado Software. Uma questão que sempre retorna nessas discussões é: **quais as diferenças entre o Model View Controller (MVC) e o Model View Presenter (MVP)?**

Model View Controller (ou MVC)

MVC é um padrão de apresentação da interface do usuário que se concentra em separar a UI (View) de sua camada de negócios (Model). Para isso, o MVC define três componentes:

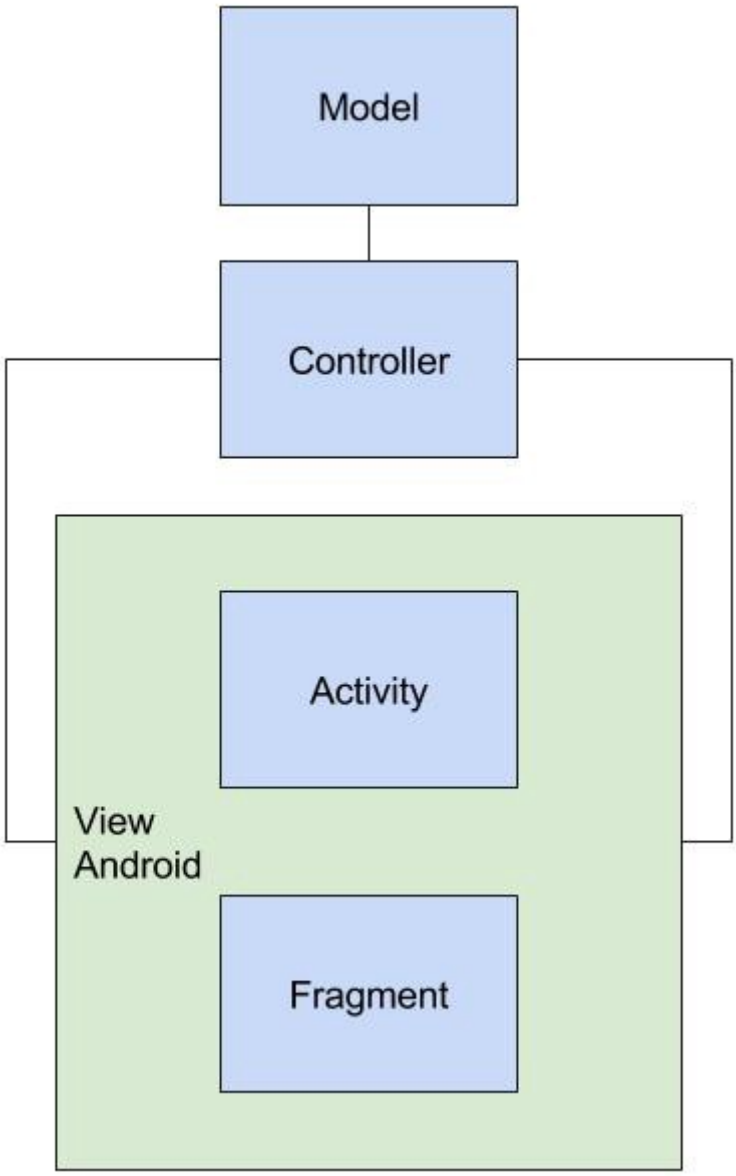

Diagrama do Model View Controller.

- **View:** apresenta os elementos da UI.
- **Controller:** responde as ações da UI.
- **Model:** comportamentos de negócio e gerenciamento de estado.

Na maioria das implementações todos os três componentes podem interagir diretamente uns com os outros e em algumas implementações o controlador é responsável por determinar qual visualização vai ser exibida.

> *Para mais detalhes sobre MVC, veja (em inglês): aqui.*

Model View Presenter (ou MVP)

MVP também é um padrão de apresentação da interface do usuário e é considerado por muitos uma evolução dos conceitos do MVC. Ele separa as responsabilidades em quatro componentes:

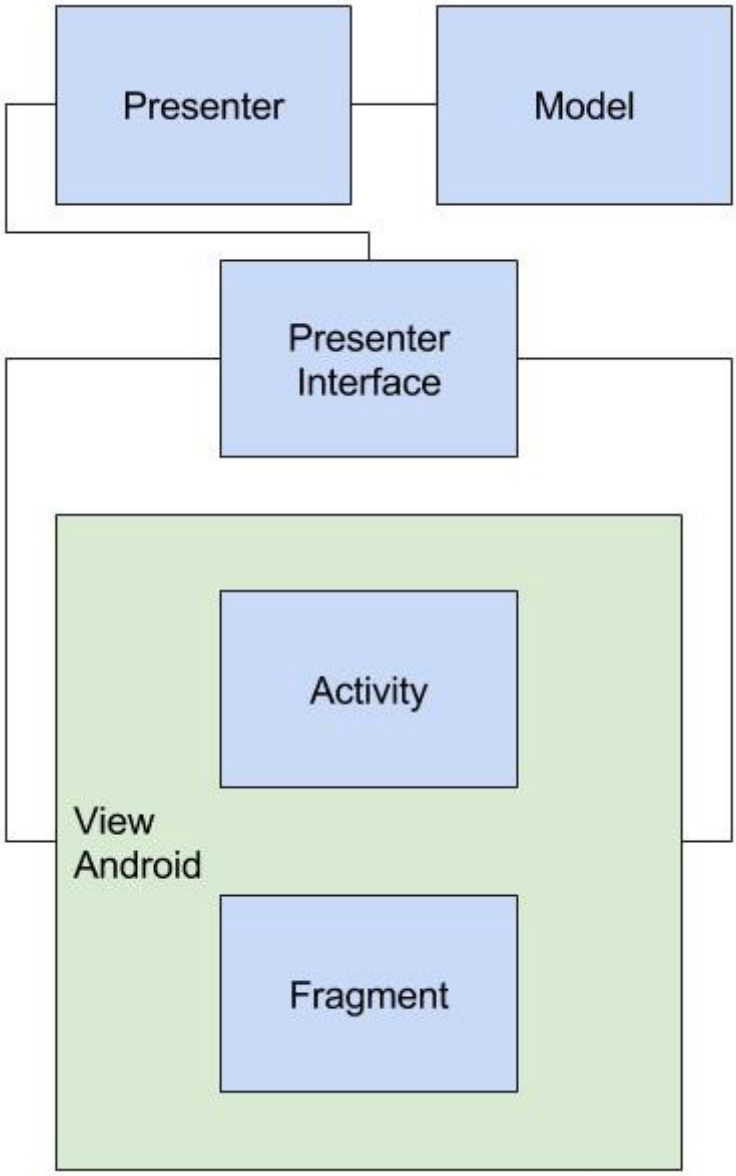

Diagrama do Model View Presenter.

- **View:** apresenta os elementos da UI.
- **Presenter (Interface):** interface que define os eventos que o Presenter poderá responder. Serve unicamente para desacoplar a View do Presenter.
- **Presenter:** responde as ações da UI controlando a interação entre a View e o Model.
- **Model:** comportamentos de negócio e gerenciamento de estado.

A interface do *Presenter* garante que seja fácil realizar testes na View. Em algumas implementações é criado um *Supervising Controller* que tem como responsabilidade cuidar da recuperação e persistência do Model.

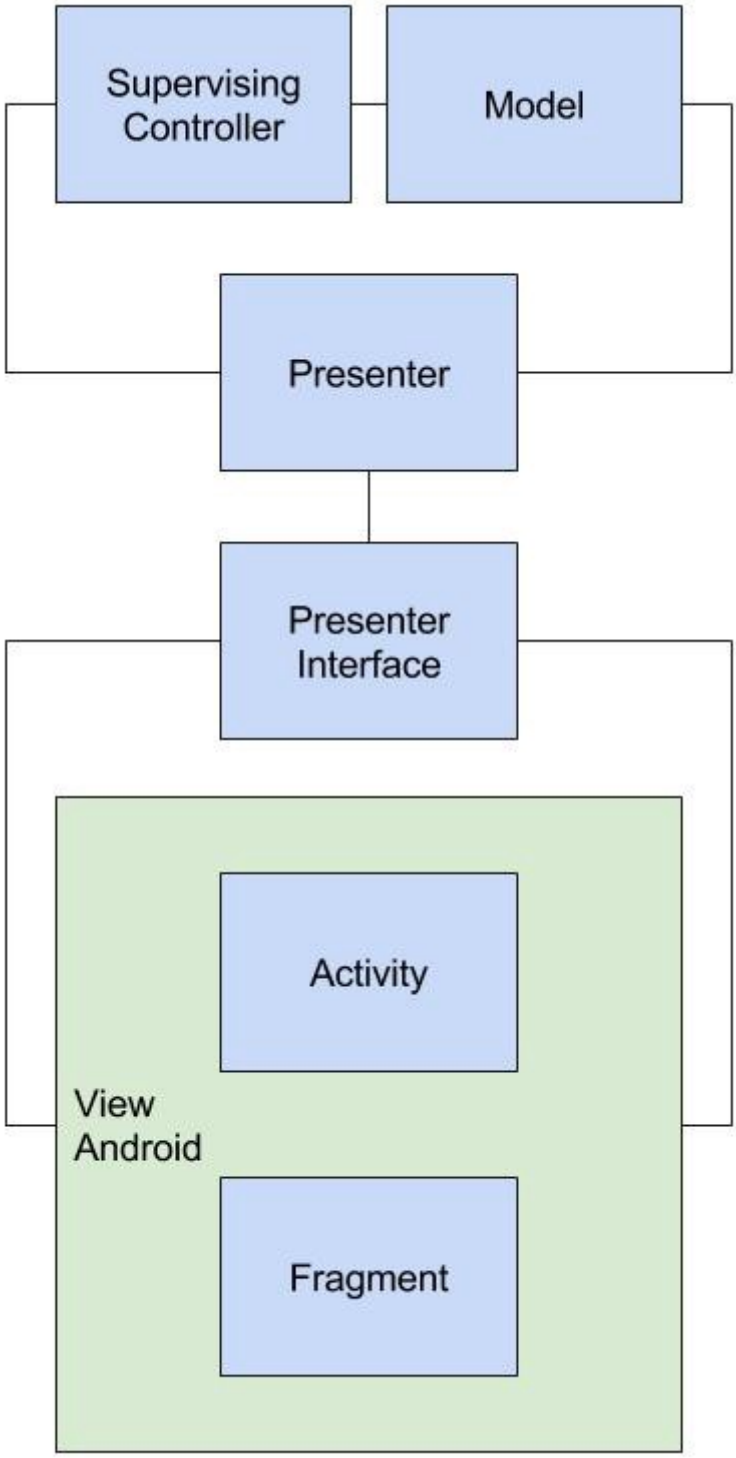

Diagrama do Model View Presenter com Supervising Controller.

> *Para mais detalhes sobre MVP, veja (em inglês): aqui.*

Principais Diferenças

Na prática, não há muitas diferenças entre as duas propostas de apresentação. Ambos se concentram em separar as responsabilidades entre as camadas e incentivar o desacoplamento da UI (View) com a camada de negócios (Model). As principais diferenças entre eles são:

- **Padrão MVC:** o Controller é baseado em comportamentos e podem ser compartilhados entre múltiplas Views, tendo menor burocracia e rápido reaproveitamento.
- **Padrão MVP:** pela grande separação entre a View e o Presenter (graças a interface), garante testes mais fáceis. As interfaces são criadas com relacionamento de 1 para 1, ou seja: para cada View existirá um Presenter.

Conclusão: e no Android, qual usar?

No Android o MVP tem se tornado popular por sua facilidade em desacoplar a View do *Presenter*. Isso facilita nos **Instrumentation Test** com a View e nos **Unit Test** com o *Presenter*. Apesar disso, o MVC pode ser muito útil no Android quando se tem o desejo de diminuir a burocracia entre as camadas (retirando uma camada de Interface).

> *Para mais detalhes sobre Instrumentation Test, veja (em inglês): aqui.*

> *Para mais detalhes sobre Unit Test, veja (em inglês): aqui.*

Conforme entendido as diferenças e realizado meus experimentos, o MVC se mostrou melhor para projetos menores onde é possível trabalhar com uma burocracia de comunicação menor, reaproveitar rápidamente os comportamentos dos Controllers entre múltiplas Views e ainda ter uma boa separação para a manutenção. Para projetos maiores o MVP se mostrou a solução ideal: seu desacoplamento entre a View e o Presenter, pela utilização de uma interface ajuda na criação de testes e permite uma evolução estrutural e de apresentação do aplicativo sem traumas maiores.

Espero que este artigo tenha sido útil e que tenha ajudado a esclarecer as diferenças entre o MVC e MVP. É importante ter em mente que eles não são “balas de pratas” e você sempre deve usa-los como um guia, mas modificar sua implementação de acordo com as suas regras de negócio.

Até a próxima!