CCM310
Arquitetura de Software e Programação Orientada a Objetos

Profa. Dra. Gabriela Biondi
Prof. Dr. Isaac Jesus
Prof. Dr. Luciano Rossi

MVC e MVVM

Padrão de Arquitetura de Software

- Escolher o padrão de arquitetura de software adequado é como definir o projeto estrutural de uma grande construção: cada decisão influencia diretamente a base, a funcionalidade e a organização do sistema

- Todo arquiteto de soluções possui o conhecimento necessário para tomar decisões arquitetônicas fundamentadas que estabeleçam a base para soluções de software confiáveis e escaláveis, explorando as vantagens e desvantagens

- Dentre os principais padrões de arquitetura de software se destacam MVC (já estudado) e MVVM

MVC (Model-View-Controller)

- Originalmente criado para aplicações desktop, mas expandido para uso na web e mobile, o MVC é um padrão arquitetônico consolidado

- Sua principal característica é a separação da aplicação em três componentes distintos, cada um com uma responsabilidade bem definida

MVC (Model-View-Controller)

- **Modelo:** incorpora os dados e a lógica subjacente da aplicação. Ele encapsula as informações e funcionalidades essenciais que definem o comportamento do sistema

- **Visualização:** assume a responsabilidade de apresentar informações aos usuários. Ela gerencia a interface do usuário, garantindo que os dados estejam acessíveis e compreensíveis

- **Controlador:** atua como maestro, gerenciando as entradas do usuário e orquestrando as interações entre o Modelo e a Visualização. Ele garante que as ações do usuário se traduzam em mudanças significativas nos dados e na apresentação.

MVVM (Model-View-ViewModel)

- Surgiu como um refinamento do MVC
- MVVM foi desenvolvido especificamente para aplicações contemporâneas, especialmente aquelas que apresentam interfaces de usuário complexas e dinâmicas
- É dividido em três componentes:

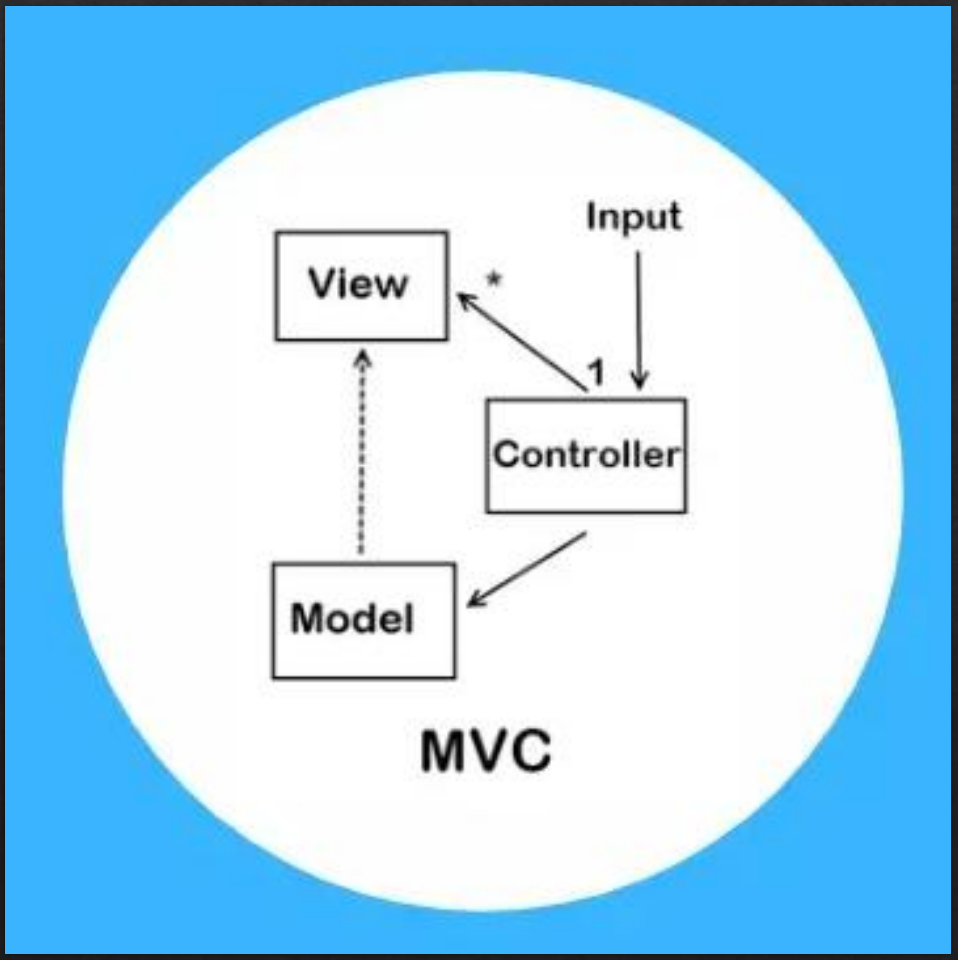
MVVM (Model-View-ViewModel)

- **Modelo:** sao seu equivalente MVC, o Modelo mantém seu papel de guardião dos dados e da lógica de negócios

- **Visualização:** o foco da Visualização está na representação visual e na interação do usuário

- **ViewModel:** atua como um intermediário entre o Modelo e a Visualização. Ele encapsula a lógica de apresentação, expondo dados e comandos aos quais a Visualização pode se vincular diretamente. Essa sinergia potencializa interações e atualizações dinâmicas do usuário, aprimorando a experiência do usuário e desvinculando a Visualização da manipulação direta de dados

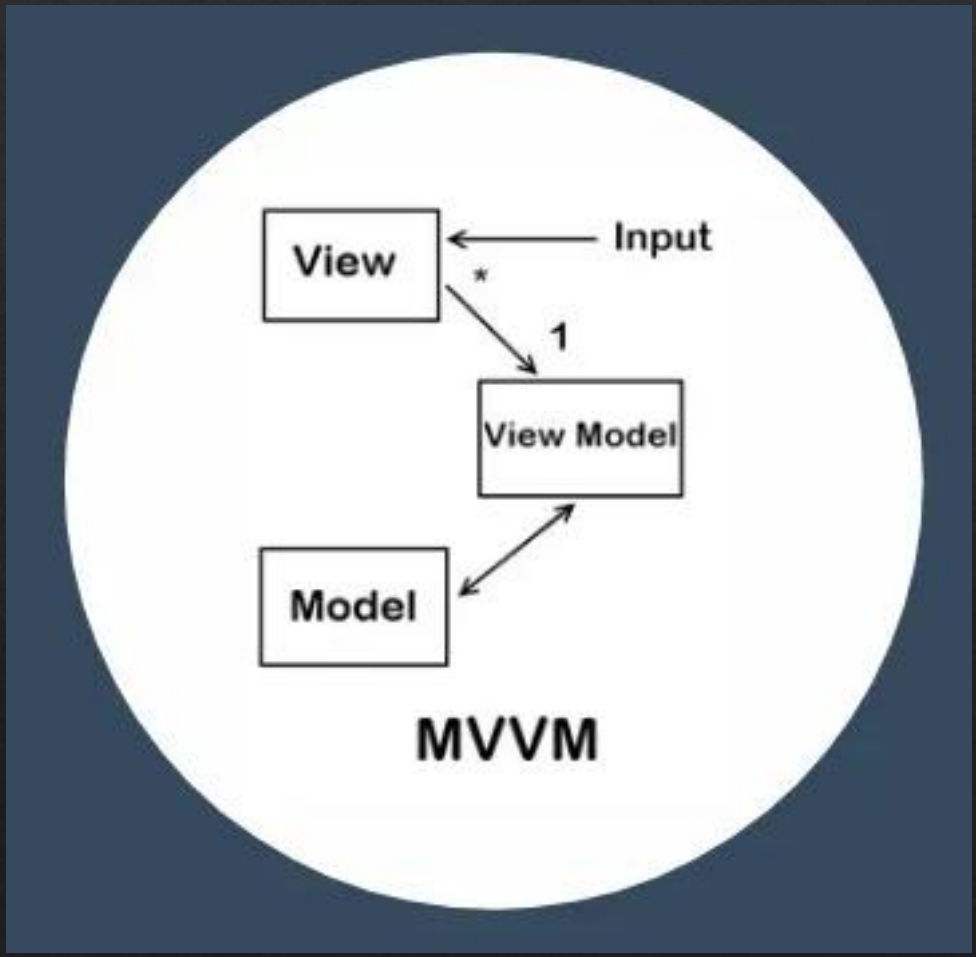
Comparação: MVC vs. MVVM



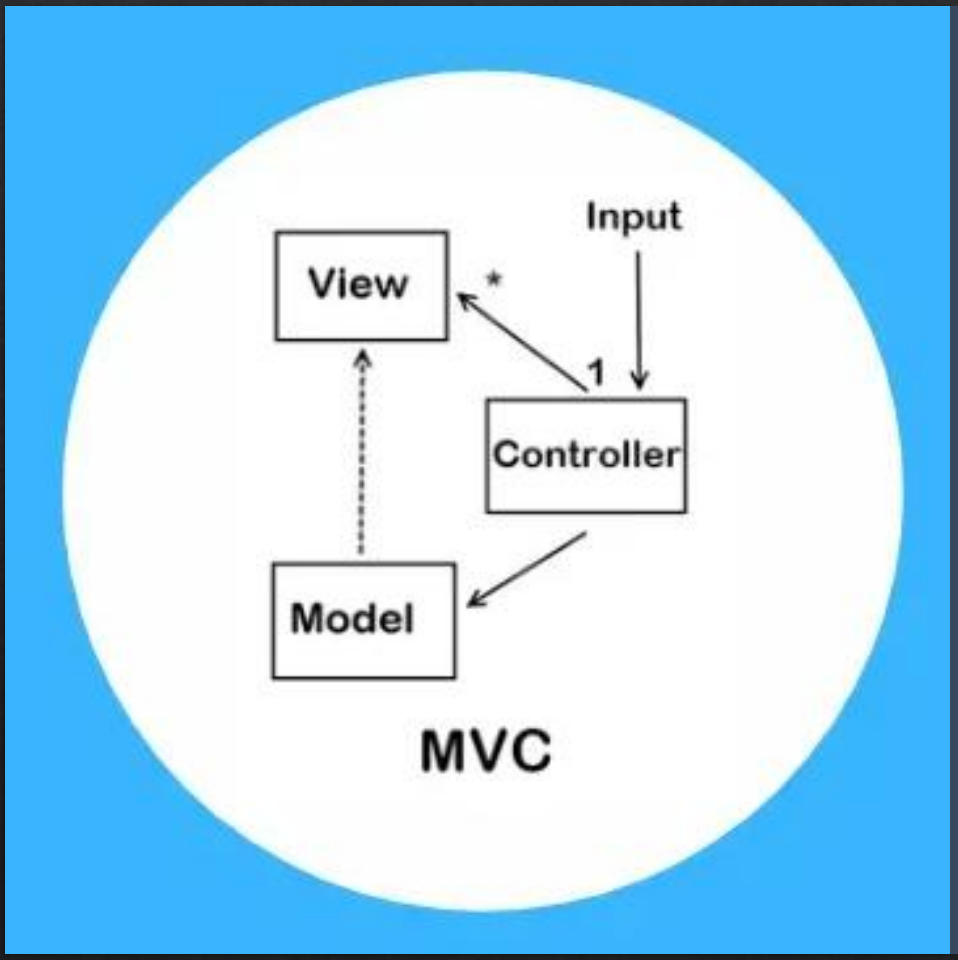
- O usuário interage com a View
- A View envia os inputs para o Controller
- O Controller processa os dados, atualiza o Model
- O Model notifica (direta ou indiretamente) a View com os dados atualizados
- Obs: a View pode ter referência ao Controller, e o Model pode ou não ser observado pela View (depende da implementação)

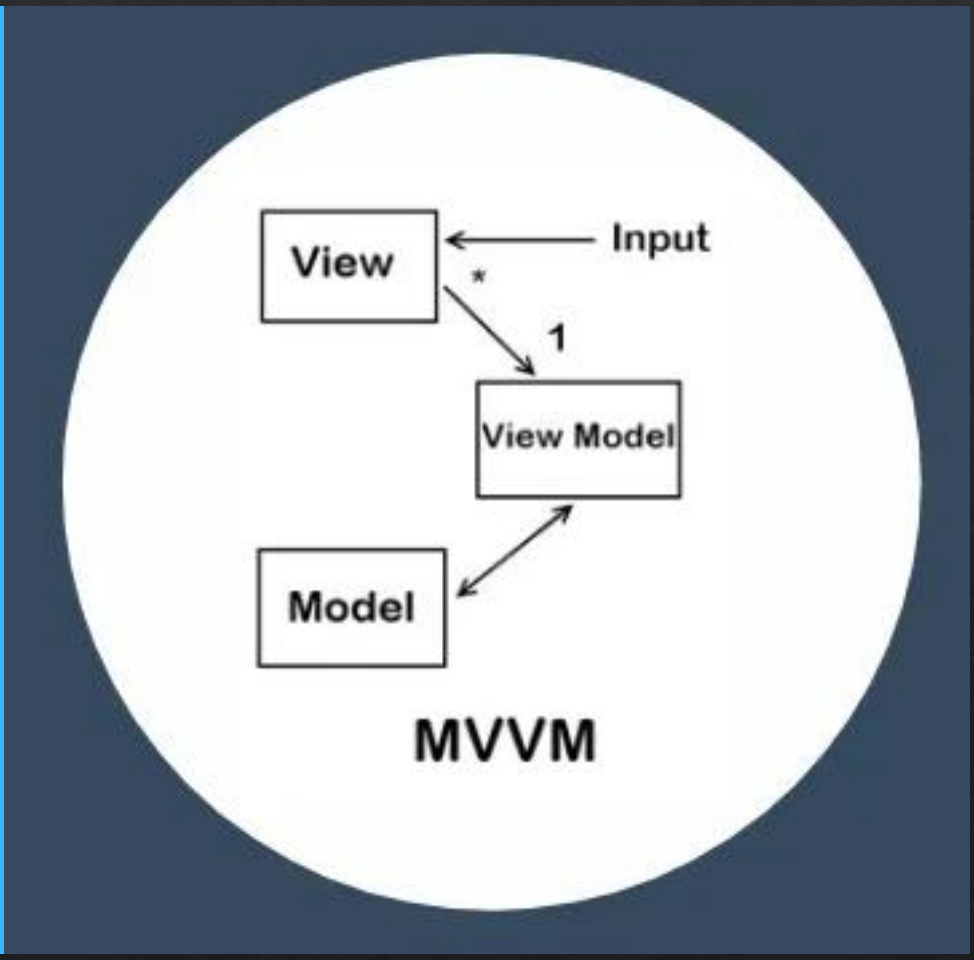
Comparação: MVC vs. MVVM

- O usuário interage com a View, que se comunica com o ViewModel
- O ViewModel processa e vincula os dados com o Model
- O data binding (vinculação) permite atualizações automáticas da View quando o estado no ViewModel muda, e vice-versa
- Obs: a View se comunica diretamente com o ViewModel por meio de data binding, reduzindo a necessidade de manipulação direta de eventos ou lógica na interface



Comparação: MVC vs. MVVM





Comparação: MVC vs. MVVM

Aspecto	**MVC**	**MVVM**
Interação da View	Com Controller	Com ViewModel
Atualização de dados	Manual (geralmente via código)	Automática via data binding
Complexidade	Mais simples	Pode ser mais complexo
Ideal para	Web, aplicações menores	Apps desktop/mobile com UI rica

Obrigada pela sua participação, nos vemos na próxima aula! :)