BACHARELADO EM SISTEMAS DE INFORMAÇÃO
CEFET-RJ NOVA FRIBURGO

Programação de Clientes Web

PROF. THIAGO DELGADO PINTO

M V x

versão: 2021.04.28

conceitos

Model-View-Controller é um modelo de separação de **responsabilidades**, não camadas!

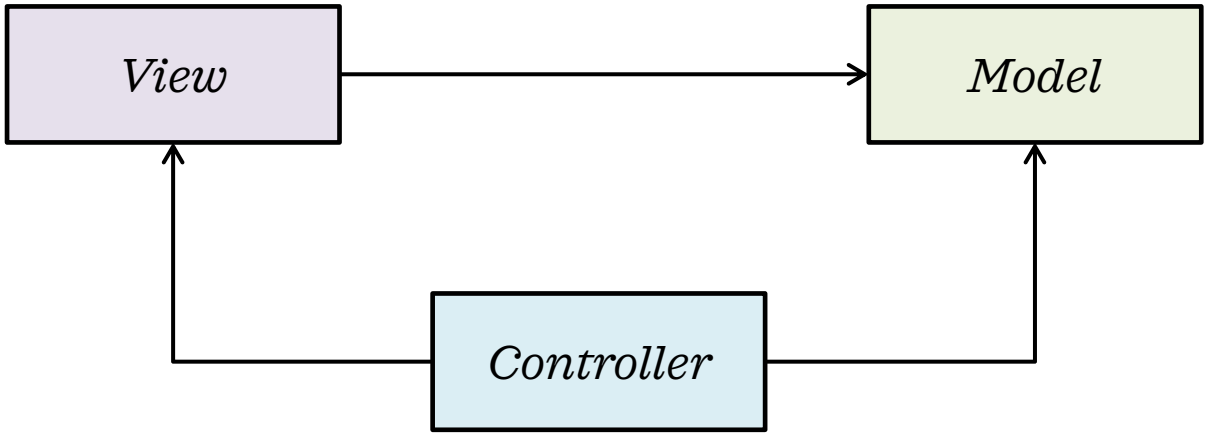
Model é responsável pelo **negócio** da aplicação
View é responsável pelas **entradas** e **saídas**
Controller pela **coordenação** entre *View* e *Model*

abordaremos o MVC visando **clientes web**

existem outras versões de MVC (ex.: *à la* GoF*, com *Observer* + *Composite* + *Strategy*)

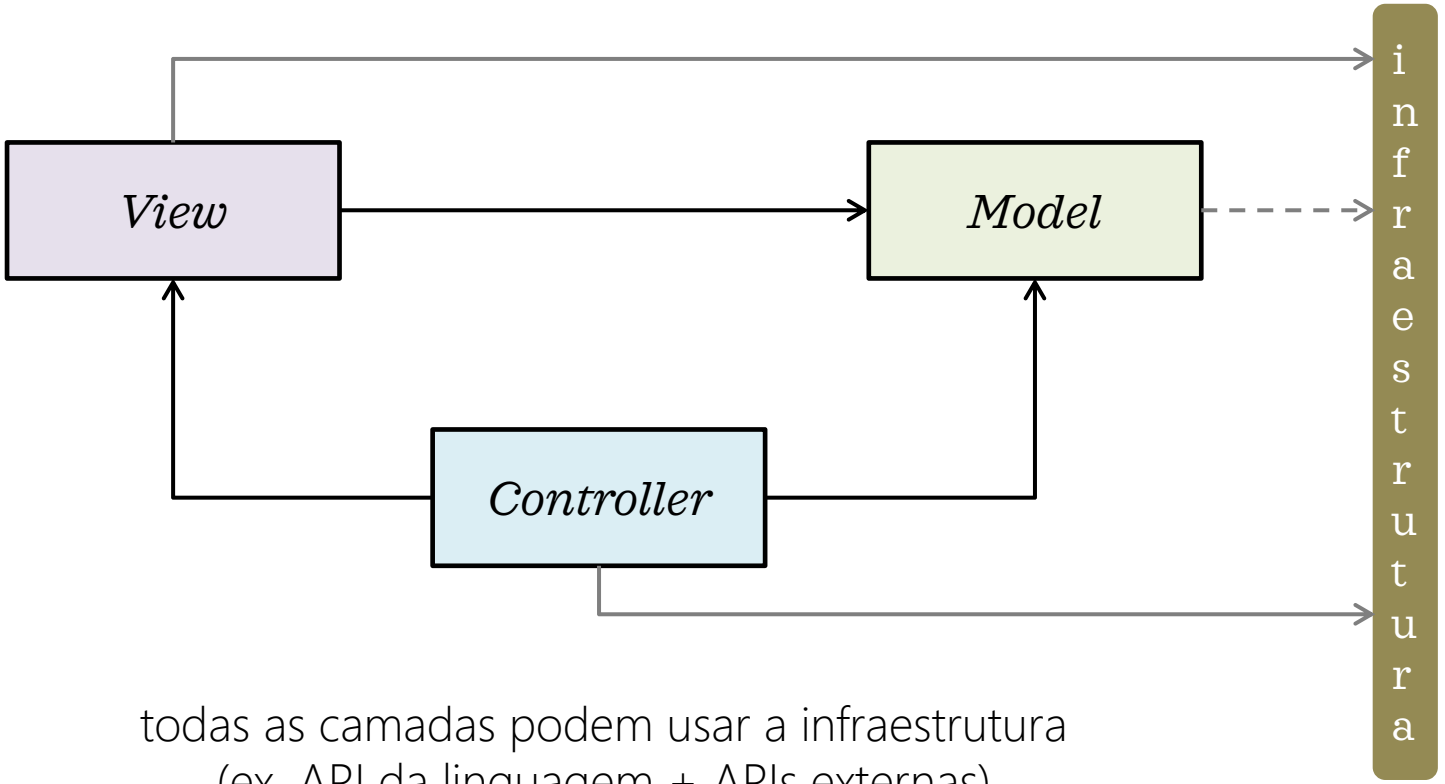
*GoF é uma referência à *Gang of Four*, os autores do livro "Padrões de Projeto", da editora Bookman

MVC – acoplamento



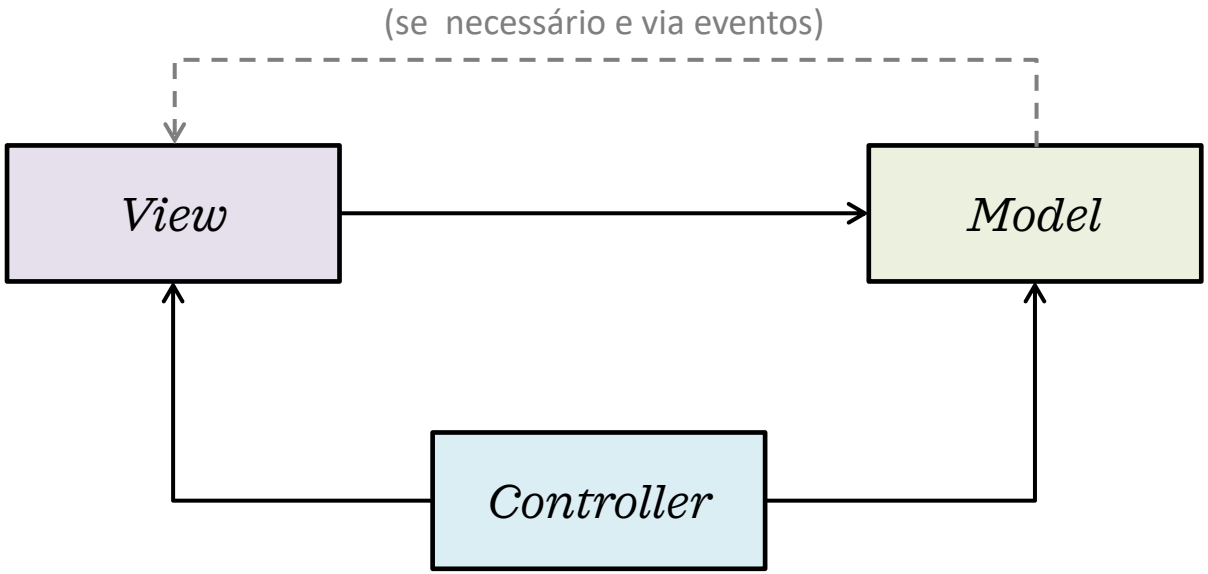
limitação do acoplamento visa reduzir impacto de manutenção

MVC – observação sobre infraestrutura



todas as camadas podem usar a infraestrutura
(ex. API da linguagem + APIs externas)
quando útil, *Model* deve procurar **abstrair APIs**

MVC – comunicação de Model para View



se for *realmente* necessária, comunicação da ***Model*** com a ***View*** deve ser feita de forma **indireta**, via abstração (ex. interface ou eventos)

Model

responsável pelo **comportamento de negócio** da aplicação

não deve se comunicar diretamente com uma *View* ou *Controller*

exemplos:

entidades do negócio, serviços, persistência de entidades, ...

utiliza a infraestrutura disponível, **evitando acoplamento com tecnologia**

- API da linguagem de programação
- APIs externas **via camada de abstração** (importante)

View

responsável por **receber as entradas** e **produzir as saídas** da aplicação

não deve se comunicar diretamente com uma *Controller*

exemplos:

recebimento de dados de entrada, produção de saídas em HTML, JSON, PDF, etc.

utiliza a infraestrutura e tecnologias disponíveis

- API da linguagem de programação + APIs externas
- criar abstrações em caso de desejo ou facilitação de migração futura

Controller

responsável por **coordenar** a comunicação entre *View* e *Model*

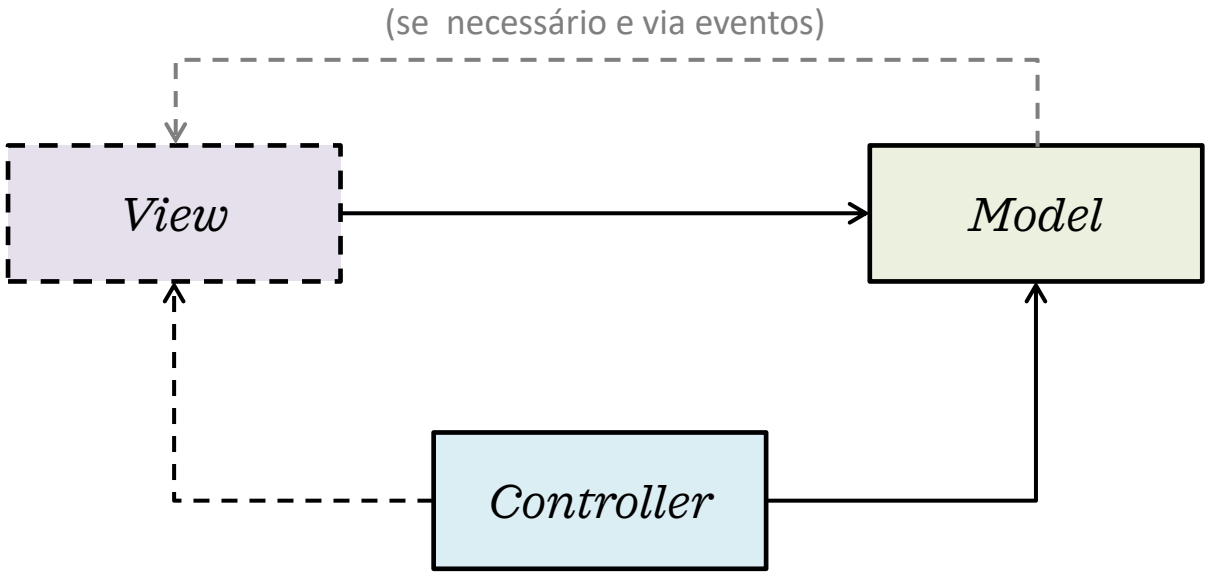
cria objetos de *View* e *Model*

- e todas as dependências necessárias

utiliza diretamente a infraestrutura e tecnologias disponíveis

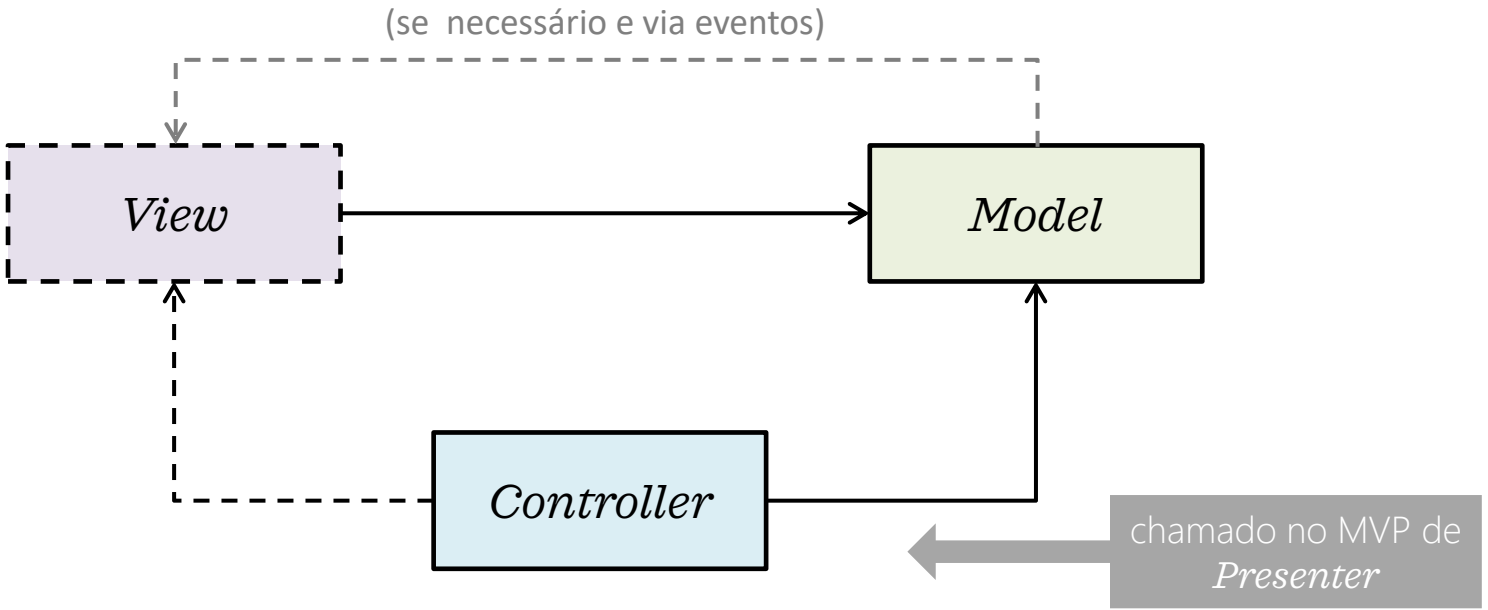
- API da linguagem de programação + APIs externas
- **não cria abstrações**

MVC – abstração da View



Representar a ***View*** como uma **interface** pode trazer benefícios:
1. facilidade de **testar** a ***Controller*** (ex. ***View*** falsa)
2. facilidade de trocar implementação da ***View*** (ex. *framework*)

MVC com abstração da View == MVP



modelo similar ao da *Taglient*[1]
também catalogada por Fowler como "***Supervising Controller***"[2]

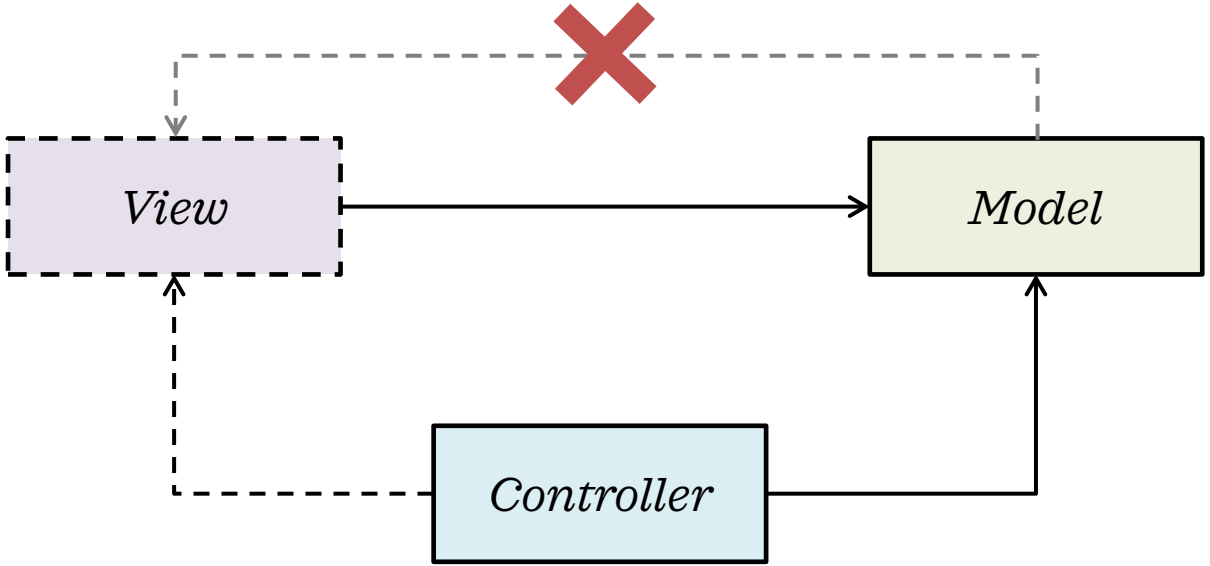
1= Citado por: BOWER, Andy. MCGLASHAN, Blair. **Twisting the MVC Triad**. 2000. Disponível em: http://www.object-arts.com/downloads/papers/TwistingTheTriad.PDF
2- FOWLER, Martin. **Supervising Controller**. 2006. Disponível em: https://martinfowler.com/eaaDev/SupervisingPresenter.html

Passive View

é o MVP **sem notificação** do *Model* para a *View*

View não é mais responsável por receber atualizações do *Model*

lógica de apresentação da *View* vai para *Controller*



observações

por padrão, usaremos *Passive View*

quando a notificação de *Model* para *View* for útil/interessante, podemos usar MVP/MVC

prática

exemplo 1

no computador: soma de dois números

exercício 1

Crie uma aplicação que calcule o Índice de Massa Corporal (IMC) de um adulto e indique sua classificação. O IMC é calculado como $peso/altura^2$, sendo o peso em $\mathbf{Kg}$ e a altura em $\mathbf{metros}$. A classificação segue a tabela a seguir:

Faixa	**Classificação**
Abaixo de 17	Muito abaixo do peso
Entre 17 e 18,49	Abaixo do peso
Entre 18,5 e 24,99	Peso normal
Entre 25 e 29,99	Acima do peso
Entre 30 e 34,99	Obesidade I
Entre 35 e 39,99	Obesidade II (severa)
Acima de 40	Obesidade III (mórbida)

exemplo 2

no computador: lista de tarefas a fazer

Uma tarefa possui id (inteiro), $descrição$ (2-60 caracteres) e um flag $fechada$ (0/1).

Deve permitir listar as tarefas, de forma agrupada: as tarefas abertas devem aparecer em cima e as concluídas abaixo.

Deve permitir alterar o status tarefa aberta/concluída.

Deve utilizar um serviço `ServicoTarefa` para representar uma abstração de serviço de persistência, acessado assincronamente, e a exceção `ServicoError`, para representar suas exceções. O serviço deve se comunicar assincronamente via HTTP (usando *fetch*) com uma API RESTful, em um servidor simulado (ex. json-server).

exercício 2

Crie uma aplicação que realize o cadastro, listagem, alteração e exclusão de **cidades**. Uma cidade contém *nome* (2-60 caracteres) e *uf* (unidade federada, com 2 letras maiúsculas). Crie um serviço `ServicoCidade` para representar uma abstração do meio de persistência, acessado assincronamente, e a exceção `ServicoError`, para representar suas exceções. Utilize os métodos `adicionar`, `atualizar`, `removerPeloId` e `comId` e `todos`, para representar suas operações. Crie uma implementação que utilize um servidor simulado com API RESTful (ex. json-server).

exercício 3

Crie uma aplicação que realize o cadastro, listagem, alteração e exclusão de **fornecedores**. Um fornecedor contém *nome* (2-100 caracteres), *cnpj* (14 caracteres numéricos), *telefone* (com DDD, de 10 a 11 caracteres numéricos) e *cidade* (referência para o *id* de uma cidade). Crie um serviço `ServicoFornecedor` para representar uma abstração do meio de persistência, acessado assincronamente, e a exceção `ServicoError`, para representar suas exceções. Utilize os métodos `adicionar`, `atualizar`, `removerPeloId` e `comId` e `todos`, para representar suas operações. Crie uma implementação que utilize um servidor simulado com API RESTful (ex. json-server). **Utilize o serviço criado para cidades no exercício anterior, para consulta e checagem de cidades.**

para saber mais

GAMMA, Erich. HELM, Richard. JOHNSON, Ralph. VLISSIDES, John. **Padrões de Projeto**: soluções reutilizáveis de software orientado a objetos. p. 20-22. 2000. Bookman.

BOWER, Andy. MCGLASHAN, Blair. **Twisting the MVC Triad**. 2000. Disponível em: http://www.object-arts.com/downloads/papers/TwistingTheTriad.PDF

FOWLER, Martin. **Supervising Controller**. 2006. Disponível em: https://martinfowler.com/eaaDev/SupervisingPresenter.html

FOWLER, Martin. **Passive View**. 2006. Disponível em: https://martinfowler.com/eaaDev/PassiveScreen.html

FOWLER, Martin. **Presentation Model**. 2004. Disponível em: https://martinfowler.com/eaaDev/PresentationModel.html

fim

versão 2021.04.28 – atualização visual (rodapés)
versão 2021.04.27 – versão inicial