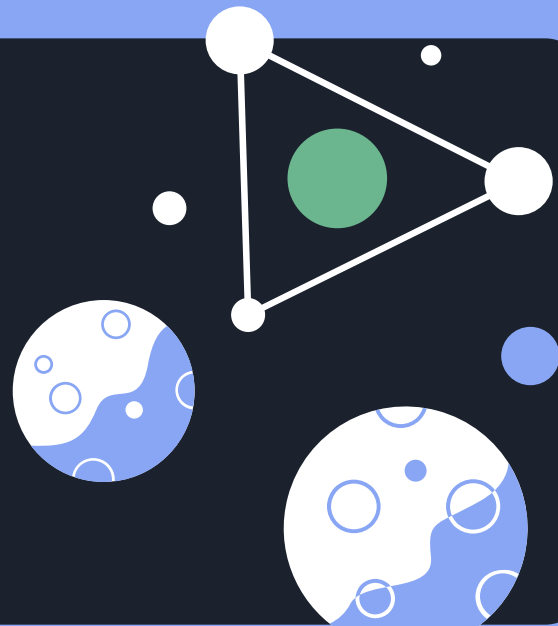


# Introdução ao Padrão MVC



Prof. Alexandre de Souza Jr.





# Sumário

**01**

**História do MVC**

**03**

**O padrão MVC**

**02**

**Padrão de Projeto**

**04**

**Implementação**





# História do MVC

- O Engenheiro Civil Christopher Alexander criou o que se considera o primeiro padrão de projeto em meados da década de 70.
- É considerado um padrão de projeto uma solução já testada e documentada que resolva um problema específico em projetos distintos.
- Através do trabalho de Alexander, profissionais da área de desenvolvimento de software utilizaram tais conceitos para iniciar as primeiras documentações de padrões de projetos, tornando-as acessíveis para toda a área de desenvolvimento.



# História do MVC

- O então funcionário da corporação Xerox PARC, Trygve Reenskaug, iniciou em 1979 o que seria o nascimento do padrão de projeto MVC.
- A ideia de Reenskaug gerou um padrão de arquitetura cujo objetivo é separar o projeto em três camadas independentes, que são o **modelo**, a **visão** e o **controlador**.
- Essa separação de camadas ajuda na **redução de acoplamento** e promove o **aumento de coesão** nas classes do projeto.
- Assim, quando o modelo MVC é utilizado, pode facilitar a **manutenção** do código e sua **reutilização** em outros projetos.



# Baixo Acoplamento vs. Coesão

## Baixo Acoplamento

É o grau em que uma classe conhece a outra. Se o conhecimento da classe A sobre a classe B for através de sua interface, temos um baixo acoplamento, e isso é bom.

## Coesão

Quando temos uma classe elaborada com um único e bem focado propósito, dizemos que ela tem alta coesão, e isso é bom.



02

Padrão de  
Projeto



# Padrão de Projeto

- O termo **padrões de projeto** ou **Design Patterns**, descreve soluções para problemas recorrentes no desenvolvimento de sistemas de software orientados a objetos.
- Padrões de projeto são estabelecidos por um **nome**, **problema**, **solução** e **consequências**.





# Padrão de Projeto

## Nome

O nome deve ser responsável por descrever o problema, a solução e as consequências.

## Problema

O problema descreve quando devemos aplicar o padrão, qual o problema a que se refere e seu contexto.

## Solução

A solução descreve os elementos que fazem parte da implementação, as relações entre eles, suas responsabilidades e colaborações.

## Consequências

As consequências são os resultados, vantagens e desvantagens ao utilizar o padrão.



# Padrão de Projeto

- Um exemplo existente e muito utilizado de padrão é o **Data Access Object**, ou simplesmente **DAO**.
- Foi uma solução encontrada para resolver problemas referentes à **separação das classes de acesso a banco de dados das classes responsáveis pelas regras de negócio**.



# Benefícios

- Aumento de produtividade;
- Uniformidade na estrutura do software;
- Redução de complexidade no código;
- As aplicações ficam mais fáceis de manter;
- Facilita a documentação;
- Estabelece um vocabulário comum de projeto entre desenvolvedores;



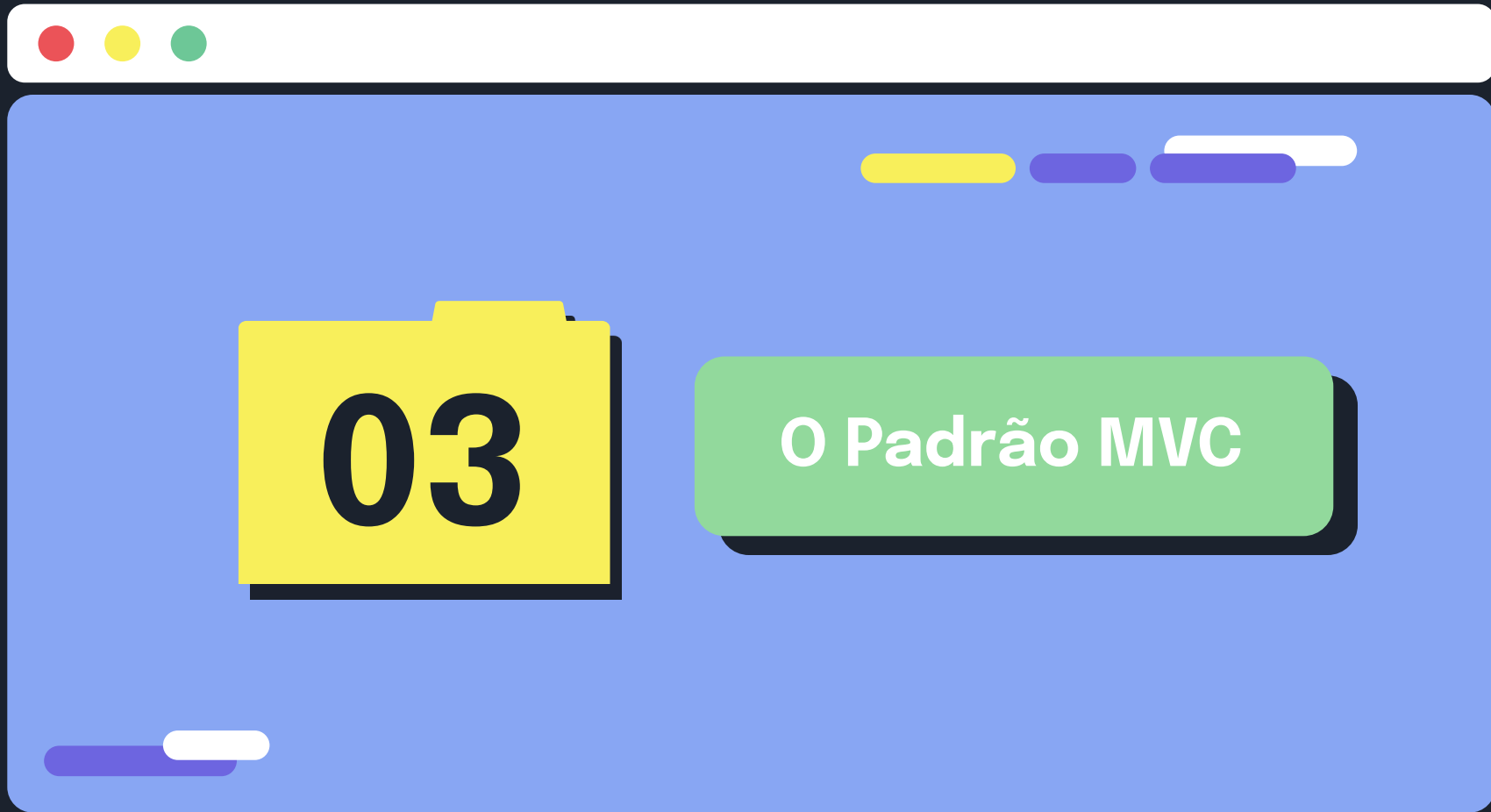
# Benefícios

- Permite a reutilização de módulos do sistema em outros sistemas;
- É considerada uma boa prática utilizar um conjunto de padrões para resolver problemas maiores que, sozinhos, não conseguiriam;
- Ajuda a construir softwares confiáveis com arquiteturas testadas;
- Reduz o tempo de desenvolvimento de um projeto.



# Padrão de Projeto

- Nos dias atuais existem diversos padrões e cada um possui uma função bem específica dentro da estrutura do projeto. Deste modo, podemos utilizar em um mesmo projeto de software mais de um padrão simultaneamente.





# O Padrão MVC (Model-View-Controller)

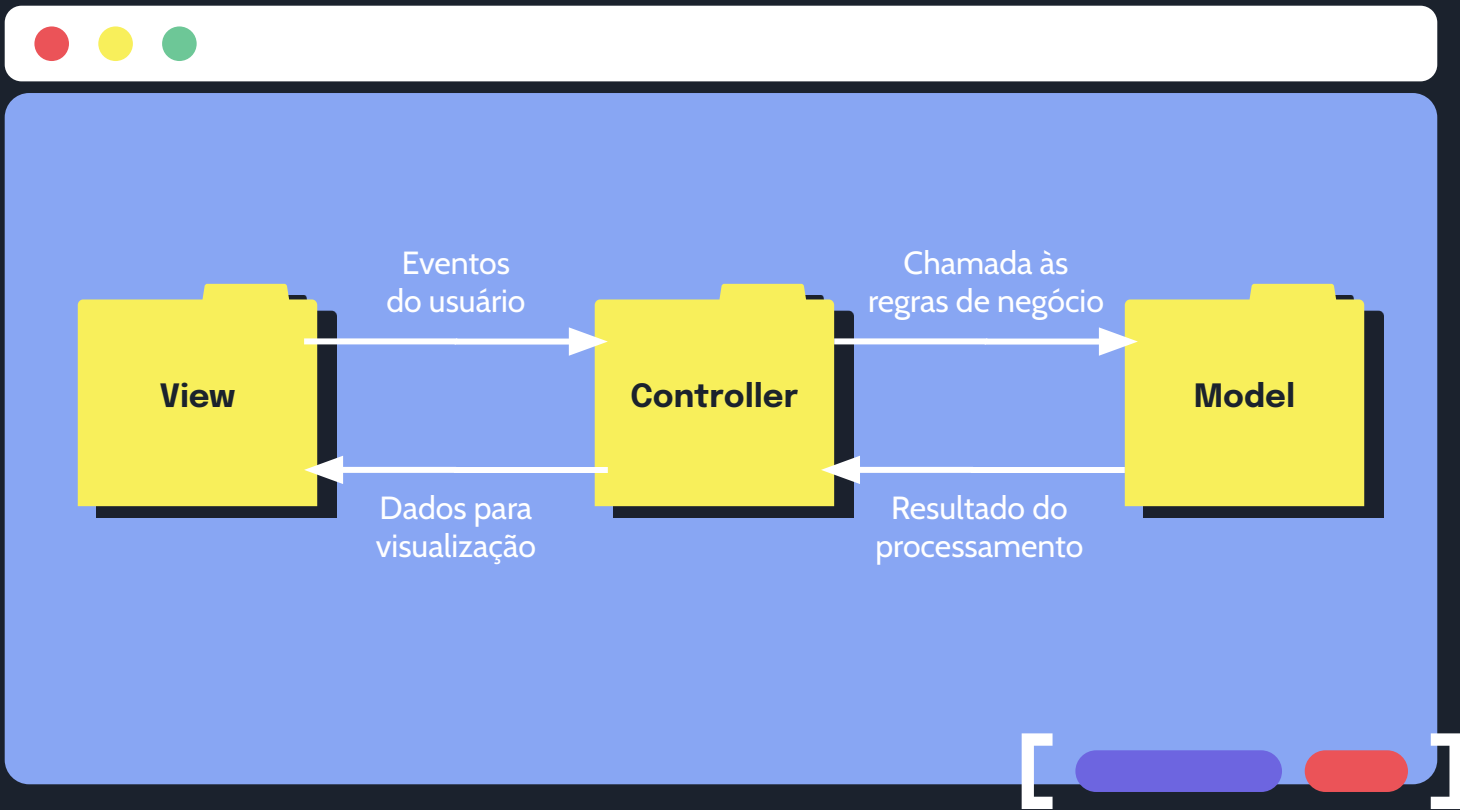
- A utilização do padrão MVC traz como benefício o **isolamento** das regras de negócios da lógica de apresentação, que é a interface com o usuário.
- Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem a necessidade de alterar as regras de negócios, proporcionando muito mais **flexibilidade** e oportunidades de **reuso** das classes.
- Pode ser utilizado em vários tipos de projetos como, por exemplo, **desktop**, **web** e **mobile**.



# O Padrão MVC (Model-View-Controller)


- A comunicação entre interfaces e regras de negócios é definida através de um **controlador**, que separa as camadas.
- Quando um evento é executado na **interface gráfica (View)**, como **um clique em um botão (Controller)**, a interface se comunicará com o controlador, que por sua vez se comunica com **as regras de negócios e/ou uma base de dados (Model)**.









# Camadas do Padrão MVC




Primeiramente o controlador (Controller), que interpreta as entradas do mouse ou do teclado enviadas pelo usuário e mapeia essas ações do usuário em comandos que são enviados para o modelo (Model) e/ou para a janela de visualização (View) para efetuar a alteração apropriada.



Por sua vez, o modelo (Model) gerencia um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado, além de ser a principal estrutura computacional da arquitetura, pois é ele quem modela o problema a ser resolvido.



Por fim, a visão (View) é responsável por apresentar as informações para o usuário através de uma combinação de gráficos e textos. Tudo que ela realmente faz é receber instruções do controle e informações do modelo e então exibi-las, além de se comunicar de volta com o modelo e com o controlador para reportar o seu estado.





# Vantagens e Desvantagens

- Separação muito clara entre as camadas de visualização e regras de negócios;
- Manutenção do sistema se torna mais fácil;
- Reaproveitamento de código, principalmente da camada de modelo, que pode ser reutilizada em outros projetos;
- As alterações na camada de visualização não afetam as regras de negócios já implementadas na camada de modelo;
- Permite o desenvolvimento, testes e manutenção de forma isolada entre as camadas;
- O projeto passa a ter uma melhor organização em sua arquitetura;



# Vantagens e Desvantagens

- Torna o entendimento do projeto mais fácil para novos programadores que não participaram de sua criação.
- As desvantagens são poucas, mas alguns desenvolvedores acabam não usando o padrão MVC por conta delas, veja algumas:
  - Em sistemas de baixa complexidade, o MVC pode criar uma complexidade desnecessária;
  - Exige muita disciplina dos desenvolvedores em relação à separação das camadas;
  - Requer um tempo maior para modelar o sistema.



04

Implementação



# Sistema Web

Vamos desenvolver uma aplicação web que tem como objetivo de **adicionar**, **ler**, **atualizar** e **remover** contatos (CRUD), utilizando o padrão de projeto MVC.

[ — — ]



# Obrigado!

Dúvidas?

[www.alexandrejunior.dev](http://www.alexandrejunior.dev)

