

Desenvolvimento Web II

Aula 02 - Padrões de Arquitetura

Prof. Fabricio Bizotto

Instituto Federal Catarinense
fabricio.bizotto@ifc.edu.br

Ciência da Computação
1 de fevereiro de 2024

1 Padrões de Arquitetura para Web

- Conceitos
- MVC
- MVP
- MVVM

2 Material Complementar

- Quiz

3 Tarefa

4 Prática

5 Experimentos

Os **Padrões de Arquitetura para Web** são soluções reutilizáveis para problemas comuns de design de software que surgem no desenvolvimento de aplicativos web. Eles fornecem **diretrizes e estruturas** para organizar o código, melhorar a escalabilidade, a manutenibilidade e a eficiência do desenvolvimento. Os mais comuns são:

- **Modelo-Visão-Controlador (MVC)**
- **Modelo-Visão-Presenter (MVP)**
- **Modelo-Visão-ViewModel (MVVM)**

Padrões de Arquitetura

MVC

Model-View-Controller

Definição

É um dos padrões de arquitetura mais conhecidos e adotados pela indústria de software. Foi introduzido pela primeira vez no final da década de 1970 por Trygve Reenskaug, um cientista da computação norueguês, e desde então se tornou um elemento básico na arquitetura de aplicativos. O padrão facilita a separação de interesses dividindo o aplicativo em três componentes principais.



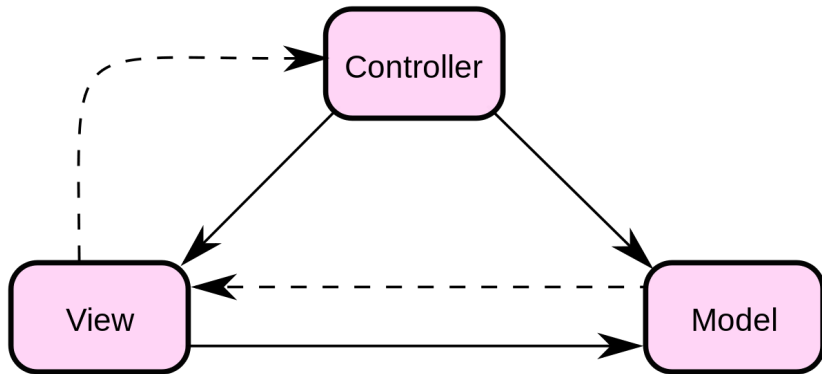


Figura: Estrutura do MVC.

Explicando

- **Model:** define quais dados o aplicativo deve conter. Se o estado desses dados mudar, o modelo geralmente notificará a visualização (View) (para que a exibição possa mudar conforme necessário) e, às vezes, o controlador (se for necessária uma lógica diferente para controlar a visualização atualizada).
- **View:** define como os dados do aplicativo devem ser exibidos (Interface com o Usuário). A visualização é responsável por receber a entrada do usuário e encaminhá-la para o controlador.
- **Controller:** atua como intermediário entre o modelo e a visualização. O controlador é responsável por receber a entrada do usuário da visualização e atualizar o modelo conforme necessário.

Padrões de Arquitetura

MVP

Model-View-Presenter

Aborda algumas das desvantagens da abordagem MVC tradicional. Originou-se no início da década de 1990 na Taligent, uma joint venture entre Apple, IBM e Hewlett-Packard. Foi ainda mais popularizado pelo Dolphin Smalltalk em 1998 e, em 2006, a Microsoft adotou o MVP para programação de interface de usuário no framework .NET.



No MVP quem manda é o View. Cada View chama seu Presenter ou possui alguns eventos que o Presenter escuta.

Exemplo

Quando o usuário clica no botão “Salvar”, o manipulador de eventos na View delega ao método “OnSave” do Presenter. O Presenter fará a lógica necessária e qualquer comunicação necessária com o Modelo e, em seguida, chamará de volta a Visualização por meio de sua interface para que a Visualização possa exibir que o salvamento foi concluído.

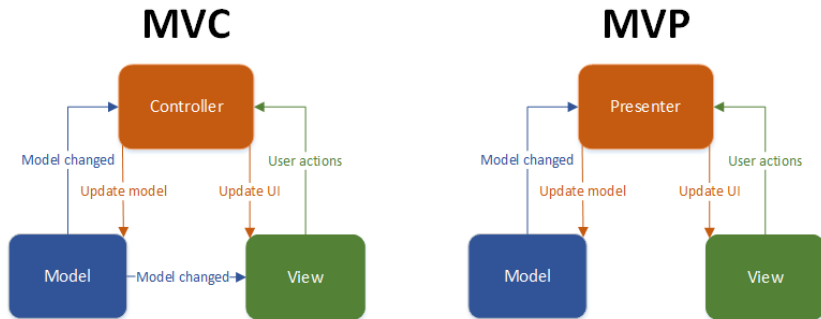
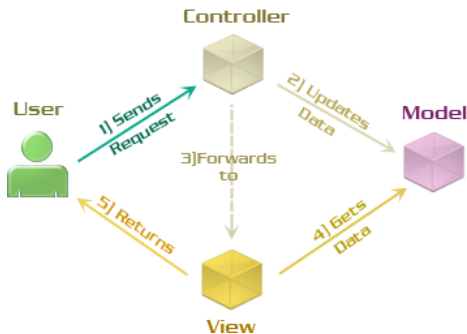


Figura: Estrutura do MVP.

MVC vs MVP

- O **MVC** não coloca o View no comando, os Views atuam como escravos que o Controlador pode gerenciar e direcionar.
- No **MVC**, as visualizações são sem estado, ao contrário das visualizações no MVP, onde são com estado e podem mudar com o tempo.
- No **MVP**, as Views não têm lógica e devemos mantê-las o mais burras possível. Por outro lado, Views em MVC podem ter algum tipo de lógica.
- No **MVP**, o Presenter é desacoplado da View e se comunica com ela através de uma interface. Isso permite que o Presenter seja testado sem a View.
- No **MVP**, as visualizações são completamente isoladas do modelo. No entanto, no MVC, as Views podem se comunicar com o Modelo para mantê-lo atualizado.

Model View Controller



Model View Presenter



Padrões de Arquitetura

MVVM

Model-View-ViewModel

Padrões de Arquitetura para Web

MVVM - Model-View-ViewModel

O MVVM foi criado pelo arquiteto de software do WPF (Windows Presentation Foundation) e Silverlight da Microsoft, John Grossman em 2005. Desde então, ele vem sendo usado principalmente no desenvolvimento mobile. Ele foi criado para ser usado em aplicativos WPF e usava XAML (uma linguagem declarativa para objetos e suas propriedades) a fim de separar a interface do usuário da lógica de negócios, aproveitando o data binding que é a vinculação de dados. Na prática, a camada Model não se comunica com a View nem a View se comunica com a Model.

Onde é usado?

O MVVM é particularmente adequado para aplicativos de **UI complexos**, onde é necessária uma extensa ligação de dados, e para projetos que usam estruturas como WPF, UWP, Angular e Xamarin. Com seu forte foco no desenvolvimento de UI, o MVVM se tornou popular no mundo do **desenvolvimento móvel**.

- **Model:** representa os dados e as regras de negócios. Ele é responsável por recuperar, armazenar e processar dados.
- **View:** é a interface do usuário. Ela é responsável por exibir os dados ao usuário e capturar a entrada do usuário.
- **ViewModel:** é um intermediário entre a View e o Model. Ele é responsável por expor métodos, comandos e outras propriedades que ajudam a manter o estado da View, manipular a entrada do usuário e se comunicar com o Modelo.
- **Data Binding:** é o mecanismo que sincroniza automaticamente a View e o ViewModel. Quando os dados no ViewModel mudam, a View é atualizada automaticamente e vice-versa.
- **Send Notifications:** o ViewModel envia notificações para a View quando os dados mudam. Isso permite que a View saiba quando atualizar a interface do usuário.

Model-View-ViewModel

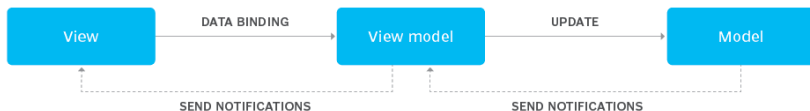






Figura: Estrutura do MVP.

-  **Padrão MVC (Model - View - Controller).**
Canal **Cod3r Cursos**.
-  **MVVM (A Arquitetura de Apps Mobile).**
Canal **Código Fonte TV**.
-  **MVVM in 100 Seconds.**
Canal **Philipp Lackner**.
-  **MVC, MVP ou MVVM?**
Canal **ArjanCodes**.

Vamos praticar um pouco o que vimos até agora?

QUIZ - Padrões de arquitetura MVC, MVP e MVVM

Padrões de Arquitetura

Tarefa

Lista de Exercícios

Observações

- As respostas devem ser entregues manuscritas (escritas à mão).
- **Entrega:** antes do início da próxima aula.

- **Questão 01:** Explique resumidamente o que é um padrão de arquitetura no contexto web.
- **Questão 02:** Explique o padrão de arquitetura MVC (Model-View-Controller). Destaque as responsabilidades de cada componente (Modelo, Visão e Controlador) e como eles interagem para criar uma aplicação estruturada.
- **Questão 03:** Explique o padrão MVP (Model-View-Presenter) e compare-o com o padrão MVC em termos de responsabilidades e interações entre os componentes. Destaque as vantagens do MVP em cenários específicos.
- **Questão 04:** Como o padrão MVC aborda a testabilidade em uma aplicação? Compare isso com as estratégias de teste no padrão MVP. Destaque os desafios e benefícios associados à testabilidade em ambos os padrões.
- **Questão 05:** Explique como a separação de preocupações é alcançada nos padrões MVC, MVP e MVVM. Destaque como essa separação facilita a manutenção do código e a colaboração entre equipes de desenvolvimento.
- **Questão 06:** Considere um cenário em que a interface do usuário precisa ser atualizada dinamicamente com base em mudanças frequentes nos dados do Modelo. Qual padrão de arquitetura (MVC, MVP ou MVVM) você escolheria para otimizar a atualização da interface do usuário e por quê?

Padrões de Arquitetura

Aula 03 - Exemplos e Experimentos Práticos

Lista de Exercícios

Padrões de Arquitetura

Exemplos Práticos

Padrões de Arquitetura para Web

Exemplo Prático - Sem padrão de projeto

```
1  from flask import Flask, render_template, request, redirect, url_for
2
3  app = Flask(__name__)
4
5  tasks = []
6  task_id_counter = 1
7
8  @app.route('/')
9  def index():
10     return render_template('tasks_no_pattern.html', tasks=tasks)
11
12 @app.route('/add_task', methods=['POST'])
13 def add_task():
14     global task_id_counter
15     description = request.form['task']
16     task = {'id': task_id_counter, 'description': description, 'completed': False}
17     tasks.append(task)
18     task_id_counter += 1
19     return redirect(url_for('index'))
20
21 @app.route('/mark_completed/<task_id>', methods=['POST'])
22 def mark_completed(task_id):
23     for task in tasks:
24         if task['id'] == int(task_id):
25             task['completed'] = not task['completed']
26     return redirect(url_for('index'))
27
28 if __name__ == '__main__':
29     app.run(debug=True)
```

Exemplos Práticos

- 🔗 Exemplo do padrão MVC com Python/Flask.
- 🔗 Exemplo do padrão MVP com Python/Flask.
- 🔗 Exemplo do padrão MVVM com VueJS.

- Modifique o exemplo do padrão MVC para adicionar a funcionalidade de **excluir** uma tarefa.
- Modifique o exemplo do padrão MVP para adicionar a funcionalidade de **excluir** uma tarefa.
- Modifique o exemplo do padrão MVVM para adicionar a funcionalidade de **excluir** uma tarefa.
- Crie outro exemplo simples de aplicação web utilizando o padrão MVC, MVP ou MVVM para demonstrar o funcionamento do padrão.

Observações

- Os experimentos devem ser publicados no **GitHub**. Explique resumidamente o que foi feito e como foi feito no arquivo **README.md**.
- O link para o repositório deve ser enviado para o e-mail do professor com o seguinte título: **[DesWebII] Aula 03 - Experimentos - NOME DO ALUNO**.
- **Entrega:** antes do início da próxima aula.

Perguntas? Comentários?

Aula 03 - Padrões de Arquitetura