INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO
CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
DISCIPLINA: PADRÕES DE PROJETO
PROF: IGOR JUSTINO
ALUNA: Daianny Evillin Costa De Oliveira

1. O que é MVC e qual seu objetivo?

O MVC (Model-View-Controller) é um padrão de arquitetura de software que organiza a aplicação em três componentes principais:

Model: Representa os dados e a lógica de negócios.

View: Responsável pela interface com o usuário.

Controller: Intermedia as interações entre a View e o Model.

Objetivo: O MVC busca separar as responsabilidades em diferentes camadas, melhorando a organização do código, facilitando a manutenção, a escalabilidade e o trabalho em equipe.

Referências:

GAMMA, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

2. Explique as funções de Model, View e Controller

Model: Gerencia os dados e a lógica de negócios. Exemplo: operações no banco de dados.

View: Apresenta os dados ao usuário e capta suas interações. Exemplo: páginas HTML.

Controller: Processa as entradas do usuário e interage com o Model para atualizar a View. Exemplo: validação de formulários.

Referência:

FOWLER, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

3. Cite exemplos de frameworks que utilizam MVC

Back-end: Laravel (PHP), Spring MVC (Java), ASP.NET MVC (C#), Django (Python).

Front-end: Angular, React (seguem variantes do MVC).

Full-stack: Ruby on Rails.

Referência:

LAU, T. Pro ASP.NET MVC Framework. Apress, 2009.

4. Como o MVC facilita a manutenção do sistema?

O MVC facilita a manutenção ao:

Promover baixa acoplamento entre camadas (alterações no Model não afetam a View).

Melhorar a legibilidade do código.

Permitir testes independentes em cada camada.

Referência:

BUSCHMANN, F. et al. Pattern-Oriented Software Architecture: A System of Patterns. Wiley, 1996.

5. Dê exemplos de projetos onde o MVC pode ser usado

E-commerce: Catálogos de produtos (Model), páginas de visualização de produtos (View) e controle de carrinho (Controller).

Sistemas de Gestão: Gerenciamento de alunos, funcionários, etc.

Aplicações API: Sistemas de consulta e resposta organizados.

Referência:

LARMAN, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004.

6. Quais os desafios de implementar MVC?

Curva de aprendizado: Especialmente para iniciantes.

Overhead inicial: Para pequenos projetos, o padrão pode ser complexo.

Manutenção de sincronização: Manter os dados atualizados entre Model e View.

Escalabilidade desordenada: Em projetos mal planejados, o Controller pode crescer demasiadamente.

Referência:

TILKOV, S.; VENZKE, S. REST und HTTP: Einsatz moderner Webarchitekturen. dpunkt.verlag, 2015.

7. O que ocorre se não houver separação de camadas?

Código confuso e difícil de manter (alta complexidade).

Baixa reutilização de componentes.

Dificuldade em implementar mudanças sem quebrar o sistema.

Problemas de desempenho e integração.

Referência:

PRESSMAN, R. S. Engenharia de Software: Uma Abordagem Profissional. McGraw-Hill, 2005.

8. MVC pode ser usado para APIs? Justifique

Sim, o MVC pode ser usado para APIs. O Model gerencia a lógica de negócios e os dados, o Controller expõe endpoints e gerencia requisições, e a View é representada pelas respostas JSON ou XML para os clientes.

Referência:

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine, 2000.

9. Explique a comunicação entre Model, View e Controller

O Controller recebe a entrada do usuário, processa-a, interage com o Model para manipular os dados e atualiza a View.

O Model notifica a View sobre mudanças no estado.

A View solicita ao Controller novos dados quando necessário.

Referência:

SHARP, J. Microsoft ASP.NET MVC Framework Unleashed. Sams Publishing, 2011.

10. Como o MVC se aplica em aplicações web modernas?

No contexto atual, o MVC é frequentemente adaptado:

Front-end frameworks: Utilizam variantes como MVVM (Model-View-ViewModel) ou MVP (Model-View-Presenter), com React, Angular ou Vue.

Back-end frameworks: APIs REST ou GraphQL são gerenciadas com padrões MVC.

Separação de responsabilidades: Melhor para microservices e SPAs (Single Page Applications).

Referência:

LEWIS, J.; FOWLER, M. Microservices: A Definition of this New Architectural Term. ThoughtWorks, 2014.