Universidade de Brasília Faculdade UnB Gama

Orientação a Objetos | FGA0158 | Turma 02 (2023.1 – 24T23)

Professor(a): Andre Luiz Peron Martins Lanna

Entrega 3: Model-View-Controller

Cláudio Luiz Lima Corrêa Júnior	Matrícula: 15/0122021
Marcelo Adrian Ribeiro de Araújo	Matrícula: 20/2016909
Samuel Alves Silva	Matricula: 20/2063462
Victor Hugo Gomes Costa	Matrícula: 22/1022140
Vitória Aquere Matos	Matrícula: 19/0096616

1. Introdução

O desenvolvimento de software é um processo complexo que envolve a organização e a distribuição eficiente das responsabilidades entre as classes e componentes de um projeto. A falta de uma estrutura clara pode levar a um código confuso, difícil de manter e com baixa escalabilidade. É aqui que o padrão Model-View-Controller (MVC) se destaca como uma abordagem eficaz para a separação de interesses e a criação de uma arquitetura robusta.

O padrão MVC é amplamente utilizado na indústria de software devido à sua capacidade de dividir o sistema em três componentes principais: Model, View e Controller. Essa separação permite que cada componente se concentre em tarefas específicas, promovendo uma clara separação de responsabilidades e tornando o código mais legível, testável e extensível.

2. Visão geral do padrão MVC

O padrão MVC estabelece uma estrutura de comunicação e colaboração entre as classes Model, View e Controller. O Model representa os dados e a lógica de negócios subjacentes, a View é responsável pela apresentação dos dados ao usuário e o Controller manipula as interações do usuário e coordena a lógica do sistema.

A interação entre essas classes ocorre da seguinte maneira: o usuário interage com a View, que envia comandos para o Controller. O Controller, por sua vez, atualiza o Model de acordo com esses comandos e notifica a View sobre as mudanças ocorridas. A View então busca os dados atualizados do Model e os exibe para o usuário. Esse ciclo de interação contínua permite uma separação clara de responsabilidades e promove um baixo acoplamento entre os componentes.

3. Descrição das classes

Model: A classe representa os dados subjacentes e a lógica de negócios do sistema. Ela mantém o estado atual do sistema e fornece métodos para atualizar e acessar esses dados. O Model não tem conhecimento direto da View ou do Controller. Suas responsabilidades incluem armazenar os dados, validar e processar

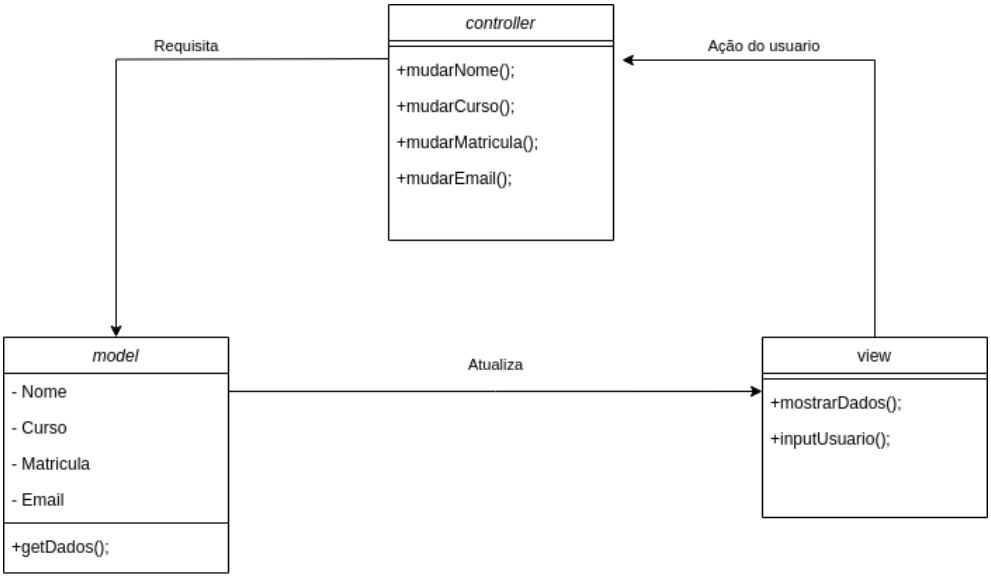
informações, e notificar a View sobre as mudanças relevantes. Exemplos de atributos podem incluir variáveis de dados e configurações, enquanto os métodos podem incluir funções para atualizar os dados, realizar cálculos e emitir eventos para notificar a View.

View: A classe é responsável pela apresentação dos dados ao usuário. Ela recebe informações do Model e renderiza a interface gráfica ou textual correspondente. A View não contém lógica de negócios, apenas exibe os dados fornecidos pelo Model. Ela também pode lidar com eventos de entrada do usuário e enviá-los para o Controller. Os métodos da View podem incluir funções para atualizar a interface gráfica, exibir dados e capturar eventos de entrada do usuário.

Controller: A classe atua como intermediário entre a View e o Model. Ela recebe comandos e eventos da View, processa-os e atualiza o Model de acordo. O Controller é responsável pela coordenação da lógica do sistema, tomando decisões com base nos comandos recebidos e atualizando o estado do Model. Ele também notifica a View sobre as mudanças relevantes para que a interface do usuário possa ser atualizada. Os métodos do Controller podem incluir funções para manipular eventos da View, atualizar o Model e coordenar a interação entre os componentes.

4. Diagrama UML de classes do padrão MVC

Os métodos *get* e *set* padrões foram ocultados para facilitação da leitura.



5. Conclusão

O padrão MVC oferece uma estrutura sólida e flexível para o desenvolvimento de software. Através da clara divisão de responsabilidades entre as classes Model, View e Controller, é possível criar sistemas bem organizados e de fácil manutenção.

O Model trata dos dados e da lógica de negócios, garantindo a consistência e a integridade dos dados subjacentes. A View cuida da apresentação dos dados ao usuário, fornecendo uma interface gráfica ou textual amigável. Por fim, o Controller atua como um intermediário entre o Model e a View, coordenando as interações do usuário, atualizando o estado do Model e garantindo que a View seja notificada sobre as mudanças relevantes.

Com o uso do padrão MVC, os desenvolvedores podem trabalhar de forma mais eficiente, colaborar de maneira mais efetiva e criar software de alta qualidade. A modularidade e a separação de interesses proporcionadas pelo MVC permitem a reutilização de código, a testabilidade e a escalabilidade do sistema, tornando-o mais adaptável a mudanças futuras.

Em resumo, o padrão MVC é uma ferramenta poderosa para a organização e a estruturação de projetos de software, fornecendo uma base sólida para o desenvolvimento de aplicações robustas e de fácil manutenção. Ao adotar esse

padrão, os desenvolvedores podem criar sistemas mais eficientes, flexíveis e duradouros.