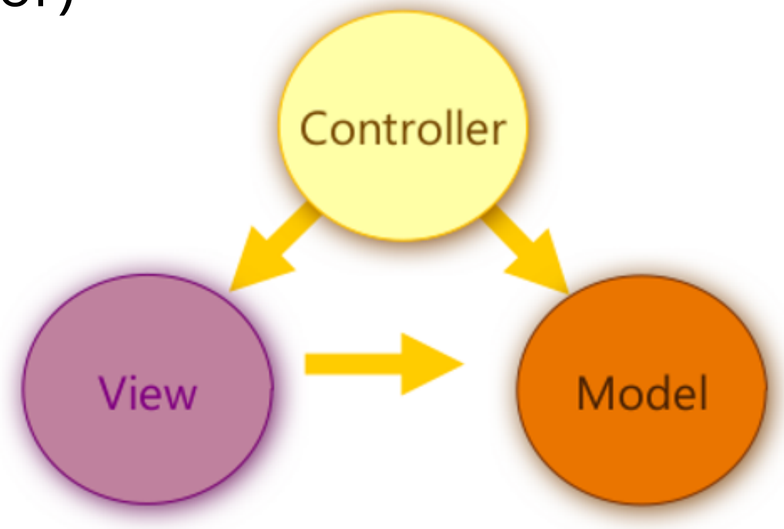
Apresentação do Projeto Escola nos padrões MVC (Model View Controller)

Grupo: Igor Xisto e Mylena Antonelli

Do que se trata o MVC?

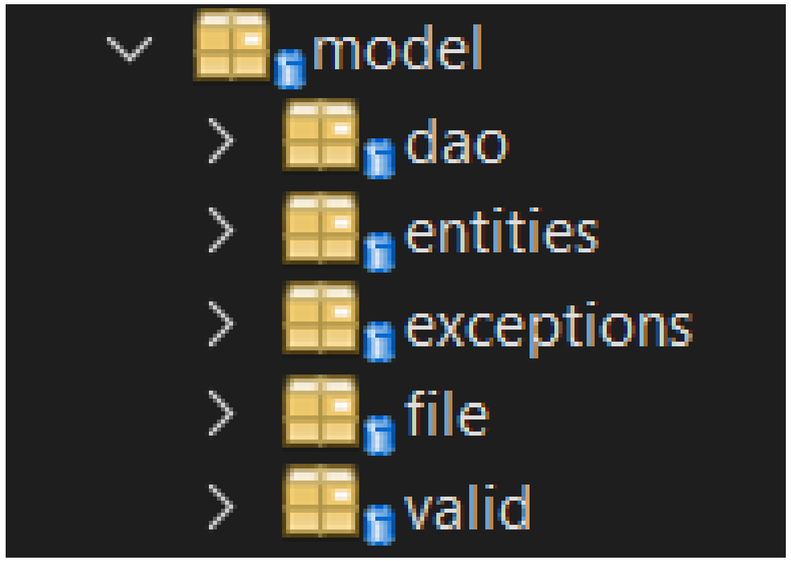
MVC (Model-View-Controller) é um padrão de arquitetura de software que separa uma aplicação em três componentes principais, cada um com responsabilidades distintas.

- Model (Modelo)
- View (Visão)
- Controller (Controlador)



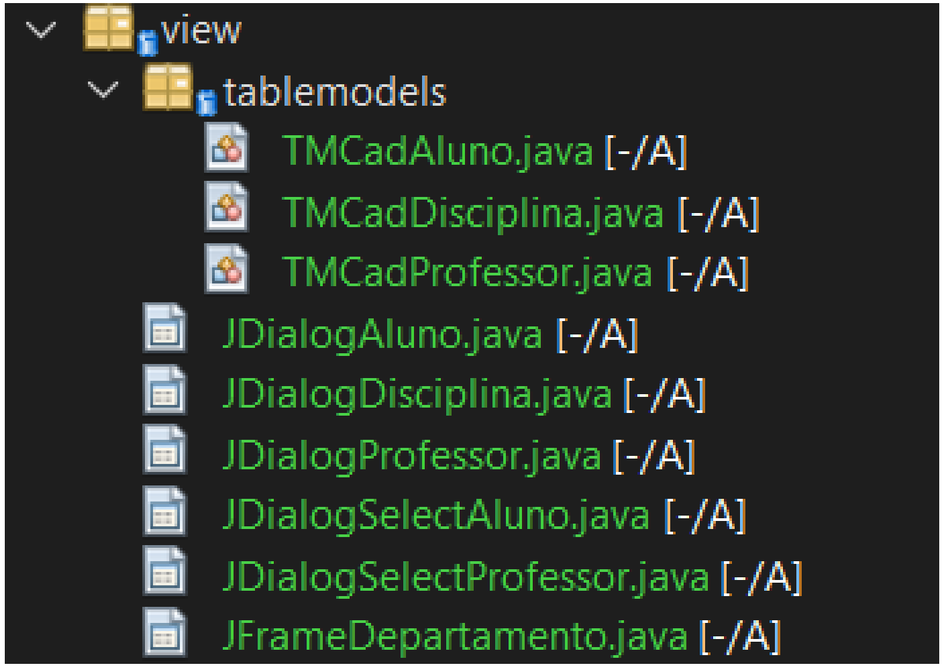
Model(Modelo)

- Responsável pela regra de negócios;
- Responsável por modelar as entidades;
- Manipulação do banco de dados(DAO, por exemplo).



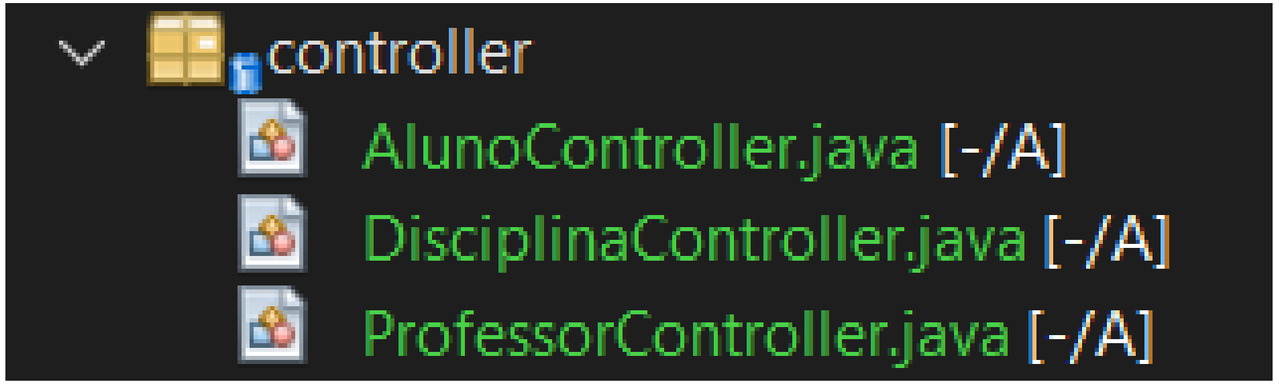
View (Visão)

- Camada que exibe a interface de comunicação com o usuário;
- A parte da aplicação que irá permitir a interação do usuário com o sistema.
- Ela exibe os dados contidos no Model e envia comandos do usuário ao Controller.



Controller (Controlador)

- Atua como intermediário entre o Model e a View.
- Ele processa as entradas do usuário (geralmente de eventos de interface), atualiza o Model e notifica a View sobre quaisquer mudanças relevantes no Model.



Fluxo de execução nas camadas do MVC

- O fluxo de execução em uma aplicação MVC começa quando o usuário interage com a View.
- O Controller recebe a entrada do usuário e a processa, atualizando o Model de acordo.
- O Model notifica a View sobre quaisquer mudanças relevantes e a View atualiza sua interface de acordo.

Vantagens de usar o padrão MVC

- Separação de responsabilidades: Camadas independentes, código organizado e fácil de manter.
- Reutilização de código: Model e View são independentes, portanto podem ser reutilizados em outras partes.
- Testabilidade: Camadas separadas facilitam testes individuais.
- Flexibilidade: Possibilidade de escolher a melhor tecnologia para cada camada.

Polimorfismo e Inversão de Dependência

- Polimorfismo: Permite que o Controller interaja com diferentes Models sem modificar seu código.
- O Controller usa uma interface, como IDAO, e se adapta a implementações concretas (DAOFile ou DAOBanco) automaticamente.
- Inversão de dependência: O Controller depende de abstrações (interfaces) e não de classes concretas, tornando o código mais modular.

Obrigado!