Arquitetura MVC

No desenvolvimento de software, é comum utilizar o padrão de arquitetura de software MVC (Model-View-Controller) para separar as responsabilidades de cada componente do sistema. O padrão MVC separa as responsabilidades em três camadas:

- Model: representa os objetos do negócio e suas regras de negócio.
- View: representa a interface do usuário.
- Controller: processa as solicitações do usuário, interage com o Model e atualiza a View.
-

No contexto do Spring Framework, é comum utilizar a arquitetura baseada em camadas MVC com as seguintes responsabilidades:

- Model: representa os objetos do negócio e suas regras de negócio. No Spring, isso é implementado por meio de classes de entidades JPA, que representam tabelas em um banco de dados relacional.
- View: representa a interface do usuário. No Spring, isso é implementado por meio de páginas HTML, geralmente com o auxílio de um motor de renderização de templates como o Thymeleaf.
- Controller: processa as solicitações do usuário, interage com o Model e atualiza a View. No Spring, isso é implementado por meio de classes de controladores, que respondem às solicitações do usuário e executam a lógica de negócio necessária para atender à solicitação.
- O Controller é responsável por gerenciar o fluxo de solicitações do usuário e executar a lógica de negócio necessária para atender à solicitação. Ele interage com o Model por meio de classes de Repositories, que fornecem métodos para persistir e recuperar dados do banco de dados. O Controller também pode interagir com outras APIs externas ou serviços.

Os Repositories são responsáveis por interagir com o banco de dados e fornecer métodos para persistir e recuperar dados. No Spring, isso é geralmente implementado por meio de interfaces que estendem o JpaRepository ou outras interfaces similares do Spring Data JPA.

Os Services são responsáveis por fornecer a lógica de negócio necessária para atender às solicitações do usuário. Eles geralmente usam os Repositories para persistir e recuperar dados, mas também podem interagir com outras APIs externas ou serviços. Os Services são frequentemente injetados em classes de Controller como dependências.

Em resumo, o Controller é responsável por gerenciar o fluxo de solicitações do usuário, o Repository é responsável por interagir com o banco de dados e o Service é responsável por fornecer a lógica de negócio necessária para atender às solicitações do usuário.