PRÁTICA ORIENTADA (P11)

LEANDRO GARCIA MAGALHÃES CERQUEIRA

1. Explique brevemente os conceitos fundamentais do padrão de arquitetura MVC (Model-View-Controller). Descreva o papel de cada componente (Model, View e Controller) e como eles interagem entre si.

O padrão MVC é como o trio dinâmico da arquitetura de software.
Model (Modelo): O modelo é onde os dados e as regras de negócios estão. É onde você armazena e manipula tudo, desde informações do usuário até lógica de negócios. Ele não se preocupa com a forma como os dados são exibidos ou como os usuários interagem com eles, apenas com a manipulação dos dados.
View (Visualização): Esta é a cara bonita da operação. A visualização é tudo o que o usuário vê e com o que interage. São as páginas da web, os elementos da interface do usuário, os gráficos, os botões e etc... As visualizações são como a fachada de uma loja, a parte que o mundo vê.
Controller (Controlador): Este é o intermediário. Ele recebe as entradas do usuário da visualização, interpreta essas entradas e decide o que fazer a seguir. Ele pode atualizar o modelo, selecionar a visualização correta para mostrar ao usuário ou fazer qualquer outra coisa necessária para manter a relação entre o modelo e a visualização

2. Quais são as principais vantagens de usar o padrão MVC em uma aplicação web? Dê exemplos de situações em que a separação de responsabilidades oferecida pelo MVC é benéfica

Separação de Responsabilidades: O MVC divide a aplicação em três componentes distintos, cada um com sua responsabilidade clara. Isso facilita a manutenção e o desenvolvimento, pois cada parte pode ser modificada, testada e atualizada independentemente das outras. Por exemplo, se você precisa alterar a aparência da sua aplicação, pode mexer na camada de visualização sem afetar o modelo ou o controlador. Isso torna o código mais organizado e mais fácil de entender e trabalhar.
Reutilização de Código: Com o MVC, é mais fácil reutilizar código em diferentes partes da aplicação. Por exemplo, suponha que você tenha uma lógica de negócios complexa no seu controlador que precise ser usada em várias partes da sua aplicação. Com o MVC, você pode simplesmente chamar essa lógica de diferentes visualizações, sem precisar reescrevê-la. Da mesma forma, se você tiver diferentes visualizações que precisam exibir os mesmos dados, pode reutilizar os modelos existentes. Isso economiza tempo e esforço de desenvolvimento e ajuda a manter a consistência em toda a aplicação.

3. Crie um cenário hipotético de uma aplicação web simples e mostre como esta aplicação funciona se implementada utilizando MVC.

Usando um exemplo parecido com o que o professor deu na aula, digamos que um usuário está preenchendo um formulário, logo após concluir ele clica em um botão. A **visualização** pega essa interação e a envia para o **controlador**, dizendo: "Ei, o usuário quer isso!". O controlador pega essa informação, vai na camada de **modelo** e busca dados para

manipular com métodos de validação, por exemplo. Estando tudo ok ele retorna para o usuário que os dados preenchidos estão corretos, caso contrário retorna um erro, por exemplo. Então, basicamente, o MVC mantém as coisas organizadas e separadas, o que facilita a manutenção e o desenvolvimento de aplicativos.

4.Como o MVC facilita a manutenção e a escalabilidade de um projeto? Dê exemplos práticos de como a estrutura do MVC contribui para esses objetivos.

O padrão MVC contribui para a manutenção e a escalabilidade de um projeto ao promover a separação de responsabilidades e facilitar a adição e modificação de funcionalidades sem impactar outras partes da aplicação. Isso torna o desenvolvimento mais ágil, eficiente e menos propenso a erros. Um exemplo de manutenção: Imagine que eu preciso fazer uma alteração na lógica de negócios da sua aplicação. Com o MVC, essa tarefa é mais fácil porque a lógica de negócios está isolada no controlador. Então eu posso fazer a alteração diretamente no controlador sem se preocupar com a interface do usuário ou com o armazenamento de dados, mantendo assim a separação de responsabilidades. Isso reduz o risco de introduzir bugs inadvertidamente em outras partes da aplicação. Já no caso da escalabilidade, um exemplo é quando você precisa lidar com um aumento no tráfego da sua aplicação. Com o MVC, é mais fácil escalar horizontalmente, adicionando mais servidores para distribuir a carga. Como o modelo é independente da interface do usuário e do controlador, você pode simplesmente adicionar mais instâncias do aplicativo sem alterar a lógica de negócios subjacente.

5.O que é o Spring Boot e quais são seus principais objetivos? Explique como o Spring Boot simplifica o desenvolvimento de aplicativos Java.

O Spring Boot é uma ferramenta para desenvolver aplicativos Java de forma rápida e fácil. Seu principal objetivo é simplificar o processo de desenvolvimento, tornando-o mais ágil e produtivo. Um exemplo é imaginar que você está construindo uma casa. Em vez de começar do zero, cortando árvores para fazer suas próprias tábuas de madeira e misturando cimento para construir suas próprias paredes, você usa um kit de construção pré-fabricado. O Spring Boot é como esse kit de construção pré-fabricado para o desenvolvimento de aplicativos Java. O que agiliza o processo de desenvolvimento com Spring é por exemplo a configuração automática, a convenção em configurações por exemplo, se você tiver um arquivo de propriedades chamado application.properties o Spring Boot automaticamente lê e aplica essas configurações para você. Além disso, ele traz facilidade na construção e implantação, integração com o ecossistema do Spring e suporte para o desenvolvimento de microsserviços.

6. Pesquise sobre o ciclo de vida de uma aplicação Spring Boot e o descreva aqui, incluindo as fases de inicialização, configuração e execução. Destaque a importância de anotações.

Basicamente, uma aplicação Spring Boot passa por um ciclo de vida de 3 fases: Inicialização, Configuração e Execução.
Inicialização: Esta é a fase em que a aplicação Spring Boot é iniciada. Durante a inicialização, o Spring Boot detecta automaticamente todas as classes anotadas com

@SpringBootApplication ou @EnableAutoConfiguration. Essas anotações indicam ao Spring Boot que a classe é uma configuração principal da aplicação e deve ser considerada durante a inicialização. Tipo a main, né Degas?

Configuração: Nesta fase, o Spring Boot configura todas as dependências e componentes necessários para a aplicação funcionar corretamente. Isso inclui a configuração de bancos de dados, servidores da web, segurança, entre outros. As anotações desempenham um papel importante aqui, pois são usadas para marcar classes e métodos que o Spring Boot deve considerar durante a configuração.

Execução: Após a fase de configuração, a aplicação está pronta para ser executada. Durante a execução, o Spring Boot inicializa todos os "*beans"* e componentes necessários e disponibiliza o aplicativo para lidar com solicitações HTTP, se for uma aplicação da web, por exemplo. Nesta fase, as anotações continuam sendo importantes para indicar como os componentes devem se comportar. Por exemplo, a anotação @RequestMapping que utilizamos em exercícios na sala é usada para mapear métodos de controlador a endpoints HTTP, enquanto @Service é usada para marcar classes de serviço que serão gerenciadas pelo Spring.

7.Você conhece outros Frameworks para desenvolvimento de APIs Rest como o Spring Boot? Pesquise sobre alguns (inclusive de outras linguagens) e fale um pouco sobre eles.

Express.js (Node.js) é um framework minimalista e flexível para Node.js, adequado para a construção de APIs RESTful. Ele fornece um conjunto robusto de recursos para lidar com solicitações HTTP, roteamento de URLs, middlewares e muito mais. Express.js é amplamente utilizado na comunidade Node.js devido à sua simplicidade e eficiência.

Ruby on Rails (Ruby) Ruby on Rails é um framework web popular para a linguagem Ruby, e também oferece suporte para o desenvolvimento de APIs RESTful. Ele vem com uma série de recursos integrados que facilitam a criação de APIs, como rotas RESTful, serialização de dados, autenticação e muito mais. Ruby on Rails é conhecido por sua produtividade e facilidade de uso.

Laravel (PHP) Laravel é um framework web moderno e popular para PHP, que também inclui suporte para construção de APIs RESTful. Ele oferece uma sintaxe elegante e expressiva, juntamente com uma variedade de recursos para facilitar o desenvolvimento de APIs, como roteamento, middleware, serialização de dados e muito mais. Laravel é conhecido por sua facilidade de uso e velocidade de desenvolvimento.

8. Uma aplicação desenvolvida com Spring Boot pode ser back end de aplicações front end desenvolvidas com outras plataformas que não sejam Java? Que relação há entre isto e o protocolo https?

Sim, pode. Uma das vantagens das APIs REST é que elas são independentes da linguagem de programação utilizada para desenvolver o backend. Enquanto o backend fornecer uma API RESTful que possa ser acessada através de requisições HTTP, o frontend

pode ser desenvolvido em qualquer tecnologia capaz de fazer requisições HTTP. Quanto ao protocolo HTTPS, ele desempenha um papel crucial na segurança da comunicação entre o frontend e o backend. O HTTPS é uma extensão segura do HTTP, que utiliza criptografía SSL/TLS para proteger os dados durante a transferência. Isso é especialmente importante ao lidar com informações sensíveis, como credenciais de usuários ou dados financeiros. Quando uma aplicação frontend faz uma requisição para o backend, seja para obter dados ou enviar informações, essa comunicação é realizada via protocolo HTTPS para garantir a segurança dos dados transmitidos. Portanto, tanto o backend desenvolvido com Spring Boot quanto o frontend desenvolvido com outras tecnologias devem suportar e utilizar HTTPS para garantir a integridade e a confidencialidade dos dados.