**Prova escrita**

**Orientação a objetos**

· O que é injeção de dependência?

R: Processo de criação e providência de instâncias de outras classes dentro de uma determinada classe para que esta seja utilizada. Esse processo não se realiza pela classe principal e sim por outros meios, como um framework por exemplo.

· O que você entende por SOLID?

R: São princípios muito importantes para o desenvolvimento de software moderno, principalmente quando o assunto é orientação a objetos. Com eles, o dev garantirá que o software seja de fácil leitura, manutenção e possibilitará a refatoração e reaproveitamento de código, sendo muito mais limpo.

· Você conhece o design pattern Strategy? De um exemplo de quando utilizá-lo.

R: O Strategy defende o encapsulamento de diferentes algoritmos, em classes diferentes, que funcionam para um mesmo contexto, sendo nele referenciado somente a interface padrão e posteriormente implementando o algoritmo que a aplica, dependendo da situação. Por exemplo, uma classe Pedido (contexto) referencia uma interface para cálculo de frete, esse cálculo pode ser feito de 2 maneiras diferentes; logo, teremos 2 classes diferentes para esses cálculos, implementando a interface criada. Assim, ao ser instanciada a classe Pedido, o programa só irá passar o algoritmo de cálculo desejado para a referência, dispensando qualquer mudança nesse contexto.

**Spring Framework**

· Quais subprojetos da Spring você conhece? Descreva-os brevemente.

R: Spring MVC, framework para criação de aplicações web no padrão MVC, ajudando no processamento de requisições e das respostas ao usuário; Spring Data, framework para auxilio na criação da camada de repositórios (DAO’s), dispensando criar interfaces manualmente e disponibilizando operações pré-implementadas; Spring Boot, facilita o processo de publishing e configuração do projeto por meio de starters declarados no pom.xml; Spring Security, fornece ferramentas para a parte de autenticação e autorização das requisições; Spring Cloud, fornece soluções para a parte de redes, implementações para conceitos como API Gateway, load balance, etc.

· Como podemos criar e injetar beans no Spring?

R: Podemos fazer uso da anotação @AutoWired ao declarar uma referência de uma classe. Para que seja possível, a classe deve estar anotada com @Component, @Repository, @Service ou @Controller. Caso não seja possível anotar a classe, deve haver um método anotado com @Bean dentro de uma classe de configuração que instancia essa classe.

· Como usar transações no Spring?

R: Por meio da anotação @Transactional em cima dos métodos que farão essas operações, tornando-as seguras e consistentes.

**Rest**

· O que você entende sobre o que são serviços web RESTful?

R: Aplicações RESTful, por padrão, é um conjunto de princípios de um Web Service que determinam que: este use arquitetura de cliente-server, suas operações sejam stateless, cacheable, tenha interface uniforme, sistema em camadas e, opcionalmente, aplique o princípio de code-on-demand.

· Cite algumas boas práticas envolvendo rest?

R: Algumas boas práticas são: Utilizar URI’s legíveis para fácil entendimento do usuário; identificar os recursos, abstraindo os “contextos” de cada operação, e padroniza-los, evitando uma hora escrever no singular e outra hora no plural; evitar URI’s que exijam o formato da representação do recurso, etc.

· Cite os métodos HTTP mais usados na arquitetura baseada REST e o que eles fazem?

R: Os principais são: PUT, para atualização de dados de um recurso; GET, para buscar dados de um recurso; POST, para criar um novo recurso; DELETE, para excluir;

· Qual a diferença entre os códigos de resposta (status codes) HTTP 200, 400 e 500?

R: 200 significa que a operação foi bem sucedida, 400 significa que há algo de errado na requisição feita pelo cliente (ex: Sintaxe) e 500 significa que houve algum erro dentro do servidor ao processar a requisição.