Prova para Programador Java com ênfase em Spring Boot

Conceitos na plataforma Java com Spring Boot, testes unitários, SQL e serviços REST

Contexto - Cadastro de Produtores Rurais

Você irá atuar em um simples sistema de cadastro de produtores rurais. Este serviço não possui interface gráfica, seja desktop, ou web.

> Embora a prova esteja utilizando Spring Boot, este framework não é o único utilizado. Mas, para esta prova, você precisa se atentar apenas às expectativas relativas ao Spring Boot e às boas práticas na construção de serviços REST e testes unitários.

Como será a avaliação

Esta prova possui 5 questões práticas e 5 questões conceituais. As questões possuem nível de dificuldade variável. Cada questão prática vale 12 pontos, e cada questão conceitual vale 8 pontos; dessa forma, a pontuação máxima da prova será de 100 pontos.

As questões foram elaboradas com o pensamento de serem resolvidas em 5 horas.

Durante a prova, a utilização da internet é totalmente permitida, inclusive para resolução das questões conceituais.

Durante a correção da prova prática, serão considerados critérios de realização parcial ou total do problema em questão. Estes critérios serão informados, isto é, você irá saber porque não conseguiu pontuação máxima em um determinado tópico. Você receberá um feedback sobre o desempenho na prova nos dias seguintes. No entanto, lembre-se que a pontuação obtida não é o único critério de avaliação.

Estrutura do projeto

Este é um projeto Java, backend, utilizando Spring Boot como principal framework. Este projeto utiliza um banco de dados embutido, o H2; portanto, ele não conecta em um banco de dados físico. O H2 é configurado para ser compatível com sintaxe de SQL do Oracle.

Arquivos importantes:

- **`src/main/java/resources/schema.sql`** - contem a estrutura de tabelas do projeto.
- **`src/main/java/resources/data.sql`** - contem um conjunto de dados iniciais principalmente utilizado nos testes unitários. **Você pode modificar este arquivo**

conforme a necessidade, inserindo ou removendo as informações, para atender à resolução das questões propostas.

Prova

A prova está dividida em duas partes: conceitos e prática. Ambas serão feitas no código fonte. Na prova de conceitos você será questionado acerca de alguns tópicos comuns ao desenvolvimento de aplicações. Na prova prática, você será apresentado a um sistema simples de cadastro de produtores rurais. Este sistema possui algumas coisas prontas, alguns problemas, bugs, e outras atividades a serem implementadas.

Parte 1 - Conceitos

Como fazer

Dentro do projeto da prova, existe uma classe chamada `ProvaConceitosController`, ela fica dentro do pacote `com.welyab.teste.programador.springboottestebasico.extra.provaconceitos.web`.

As perguntas são de múltipla escolha, cada, uma com cinco alternativas de resposta e somente uma delas está correta. Para responder, você vai apenas retornar a letra da opção que você entende como sendo a resposta. Exemplo:

```
@GetMapping(path = "pergunta-exemplo", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Resposta> perguntaExemplo() {
   // o texto da pergunta é encapsulado dentro de uma uma classe
   // você não precisa mexer na pergunta
   Pergunta pergunta = new Pergunta(
           "Quanto é 1 + 2?"
   );

   // cada pergunta possui cinco alternativas de reposta. Apenas uma delas é correta.
   // você não precisa mexer nas alternativas
   Alternativa a = new AlternativaA("1");
   Alternativa b = new AlternativaB("2");
   Alternativa c = new AlternativaC("3");
   Alternativa d = new AlternativaD("4");
   Alternativa e = new AlternativaE("5");

   // Este é o único lugar em cada questão que você precisará mexer. Você não precisa mexer nas alternativas,
   // nem na pergunta. Basta que você ofereça a reposta correta
   // sua resposta deve ser encapsulada dentro da classe Resposta, que rece seu seu construtor
   // a pergunta, e alternativa escolhida como resposta
   Alternativa alternativaEscolhida = c; // vou dar C como reposta
   Resposta resposta = new Resposta(pergunta, alternativaEscolhida);
   return ResponseEntity.ok(resposta);
}
```

A única parte de cada pergunta que você realmente vai precisar mexer é a linha onde você escolhe sua resposta:

```
//Alternativa alternativaEscolhida = null;
Alternativa alternativaEscolhida = c; // ou a, b, d, e...
```

Após selecionar suas respostas, basta enviá-las para o git junto com todo o resto do código fonte que você irá trabalhar.

Parte 2 - Prática

Todo o código do “**Sistema de Cadastro de Produtores Rurais**” está dentro do pacote `com.welyab.teste.programador.springboottestebasico`. Lembre-se que o pacote com a prova conceitual está neste mesmo projeto, mas não faz parte do sistema de código de cadastro.

Embora os testes unitários seja um parte importante do desenvolvimento de um software, aqui nesta prova você só precisará fazer quando for explicitamente solicitado. Repetindo, você não precisa escrever testes unitários para uma rotina a menos que a o problema em questão peça para você fazer isso.

Você tem liberdade para criar, deletar ou modificar absolutamente qualquer arquivo do projeto, **exceto quando dito o contrário**.

Questão 1

Na classe `ProdutorRuralController`, escreva um método que retorna um JSON com a lista de produtos cadastrados para um produtor rural. O número de inscrição do produtor rural deverá ser passado como parâmetro desta consulta.

Lembre-se de utilizar as boas práticas no desenvolvimento de APIs REST.

Neste problema você não precisa escrever testes unitários.

O JSON de resposta deverá ter o seguinte formato:

```
{
  “produtor-rural”: “Chácara Milho-Verde”,
  “inscricao”: “321654987”,
  “produtos”: [
    {
      “code”: “eb5cf44d-8596-40b8-ac3e-3a44d9747264”,
      “nome”: “Soja”
    },
    {
      “code”: “ac29e2a4-fad1-41a4-8dda-68ca983ca16c”,
      “nome”: “Feijão”
    },
    ...
  ]
}
```

Questão 2

Na classe `ProdutorRuralController`, existe um método chamado `consultarProducaoPorHectareNaData`. Nesse problema, você deverá escrever os testes unitários que julgar necessários. Lembre-se que os testes da classe `ProdutorRuralController` devem ficar na classe `ProdutorRuralControllerTest` (esta classe de teste já existe).

Avalie a necessidade de criar testes de borda e testes para cenários de sucesso e falha. Faça quantos testes quiser.

Para criar os cenários de testes, talvez seja necessário adicionar informações no arquivo `data.sql`.

Questão 3

Na classe `ProdutorRuralController`, existe um método chamado `consultarProducaoPorIntervaloData`. Este método está sendo testado na classe `ProdutorRuralControllerTest` com um único teste chamado `consultarProducaoPorIntervaloDataDeveRetornarProducaoNoIntervaloEspecificado`. O método de teste, no entanto, está **falhando**.

Sua tarefa é fazer o teste passar.

Atenção: **você não pode alterar nenhuma parte do método de teste**. Você deve encontrar os problemas possivelmente nas classes de controller, serviços e repositórios. Também, você pode fazer qualquer modificação no arquivo `data.sql` que faça sentido para resolução do problema.

Questão 4

Na classe `ProdutorRuralController`, escreva um método que atualiza somente o nome do produtor rural. O produtor rural deve ser identificado pelo seu número de inscrição.

Lembre-se de utilizar as boas práticas no desenvolvimento de APIs REST.

Neste problema você não precisa escrever testes unitários.

Questão 5

Crie um novo *controller* web chamado `ProdutoController` para manipular as informações de produtos cadastrados. Neste controller, escreva um método remove um produto cadastrado.

Lembre-se de utilizar as boas práticas no desenvolvimento de APIs REST.

Neste problema você não precisa escrever testes unitários.

Boa sorte!