Sistemas de Informação - DCOMP/IFMA

Laboratório de Engenharia de Software II

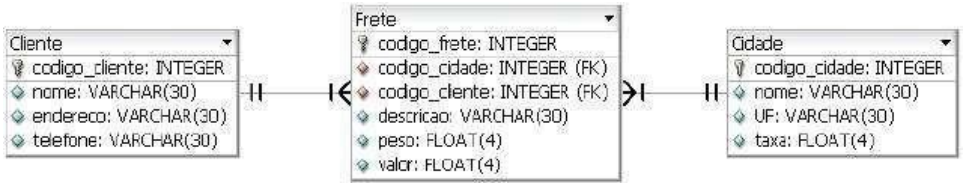
Implementação de Testes para uma API REST usando Spring Boot para um fragmento de um sistema de Transportadora

Data de Entrega: 01-12-2022

Use as técnicas que você aprendeu durante as aulas para criar um bom teste de integração. Pense com cuidado nos nomes dos métodos de teste e no nome das variáveis.

1. ***Camada de Modelo – Mapear as entidades com as anotações JPA***
2. ***Você deverá anotar as validações com o Bean Validation nas classes do modelo***

Modelo de banco de dados do sistema de Frete a ser considerado



3. **Camada de Repositório (deverá ser implementada com Spring Data JPA)**

- Cadastrar clientes com adição dos métodos buscaPor(String telefone). Todos os campos são obrigatórios
- Cadastrar **cidades,** que representam os lugares abrangidos pela empresa de transportes e contêm o nome da cidade, o estado a que pertence, e o valor para a taxa de entrega. Sendo todos os campos obrigatórios
    - incluir o método `buscaPor(String nome)` em sua interface de repositório
- Cadastro de Fretes, contendo um código, uma descrição, o peso total, um cliente e a cidade de destino, não podendo haver um frete sem os dados citados.

- Recuperar uma lista com todos os fretes de um determinado cliente ordenados por valor.

Para esta camada deverá ser implementado os seguintes testes:

- Testes de validação com *Bean Validation*
- Teste de todas as buscas no repositório

Dicas: Nessa camada é necessária apenas usar a anotação `@DataJpaTest`.

Para os testes com Bean Validation veja o código de exemplo no link:

https://github.com/jcpinheiro/eng-de-software2-2022/blob/master/06a_teste_integracao_spring/src/test/java/edu/es2/teste/spring/integracao/repositorio/ContatosRepositoryIntegrationTest.java

Para os testes das buscas veja o código no link

https://github.com/jcpinheiro/eng-de-software2-2022/blob/master/06a_teste_integracao_spring/src/test/java/edu/es2/teste/spring/integracao/repositorio/ContatosRepositoryIntegrationQueryTest.java

4. **Camada de Serviço – Sugestão FreteService**

- Implementar o cadastro de um novo frete. Deverá ser lançado exceções caso não exista a cidade ou cliente não esteja cadastrado.
- Recuperar o valor do frete, que deve ser calculado através do peso multiplicado por um valor fixo (por exemplo R$ 10,00), acrescido da taxa de entrega associada a cada cidade de destino
- Recuperar o frete de maior valor (custo)
- Recuperar a cidade que é destinatária da maior quantidade de fretes

OBS 1: veja onde deverá ficar cada parte da regra de negócio.

OBS 2: Provavelmente será preciso incluir novos métodos na camada de repositório

OBS 3: Você deverá utilizar a anotação @SpringBootTest

Dicas: Veja os códigos de exemplo disponíveis nos links:

https://github.com/jcpinheiro/eng-de-software2-2022/blob/master/06b_teste_servico_componente_spring/src/test/java/edu/es2/teste/spring/integracao/servico/ContatosServiceTest.java

https://github.com/jcpinheiro/eng-de-software2-2022/blob/master/06b_teste_servico_componente_spring/src/test/java/edu/es2/teste/spring/integracao/controle/AgendaControllerTest.java

O exemplo *AgendaControllerTest.java* faz o **uso de mocks**

5. Camada de Controle – API REST

Escreva um teste de integração para a classe FreteController. E crie testes para validar as buscas, a inclusão, as exceções e a exclusão de fretes, você também pode criar outros métodos de testes, caso necessário.

Dicas:

Use a anotação:

`@SpringBootTest(webEnvironment =`

`WebEnvironment.RANDOM_PORT)` **Link do código apresentado em aula:**

https://github.com/jcpinheiro/eng-de-software2-2020/blob/master/06c_teste_spring_restcontroller/src/test/java/edu/es2/teste/spring/restcontroller/controle/AgendaControllerTest.java