JUnit

Testes de Integração com JUnit

Prof. Evandro César Freiberger

Instituto Federal de Mato Grosso
Departamento da Área de Informática
evandro.freiberger@ifmt.edu.br

2023

Sumário

Teste de Integração

Nesse nível de teste, o objetivo é realizar testes que irão verificar a integração entre elementos do software.

Por exemplo:

- Integração entre duas classes;
- Integração entre uma classe e um componente;
- Integração entre dois componentes;
- Integração entre duas camadas;
- Integração entre uma aplicação e o banco de dados;
- Integração entre uma aplicação e uma API de serviços.

No teste de integração o que se espera é validar os serviços oferecidos pelas partes envolvidas, não como eles são executados.

Dessa forma, os testes de integração se caracterizam como testes de caixa preta, visto que invoca-se o serviço e verifica-se o resultado obtido e compara com o esperado.

Testes de Integração

O primeiro exemplo, testa a integração entre a classe DAO e o Banco de Dados.

O objetivo é fazer o teste de integração entre a Camada de Persistência e Banco de Dados.

Dessa forma, os casos de testes serão construídos a partir das classes DAO, que interagem com o Banco de Dados por meio da API JPA.

A primeira dúvida que pode surgir é: Devo Mockar o Banco?

Como o objetivo é testar a integração, não será usado o Mokito para "mokar"a comunicação com o Banco de Dados, e sim, executar as chamadas entre a camada de persistência e o banco, e verifar se as ações de persistência e recuperação de dados estão em conformidade com o esperado.

Testes de Integração - FabricaEntityManager (1)

Classe FabricaEntityManager responsavel por criar um contexto de persistência JPA, incluindo da conexão com o banco de dados.

```
1  package ifmt.cba.persistencia;
2
3  import jakarta.persistence.EntityManager;
4  import jakarta.persistence.EntityManagerFactory;
5  import jakarta.persistence.Persistence;
6
7  public class FabricaEntityManager {
8
9    private static EntityManagerFactory emf_producao;
10   private static EntityManagerFactory emf_teste;
11   public static final String UNIDADE_PRODUCAO = "restaurante_producao";
12   public static final String UNIDADE_TESTE = "restaurante_teste";
13
14   private FabricaEntityManager() {
15
16   }
17
18   public static EntityManager getEntityManagerProducao() {
19     if (emf_producao == null) {
20       emf_producao = Persistence.createEntityManagerFactory(UNIDADE_PRODUCAO);
21     }
22     return emf_producao.createEntityManager();
23   }
24
25   public static EntityManager getEntityManagerTeste() {
26     if (emf_teste == null) {
27       emf_teste = Persistence.createEntityManagerFactory(UNIDADE_TESTE);
```

Testes de Integração - FabricaEntityManager (2)

```
28      }
29      return emf_teste.createEntityManager();
30    }
31  }
```

Testes de Integração - FabricaEntityManagerTest (1)

Classe FabricaEntityManagerTest responsavel automatizar o teste de integração, que verifica se as conexões estão corretas.

```
1  package ifmt.cba.integracao;
2
3  import org.junit.jupiter.api.Assertions;
4  import org.junit.jupiter.api.Test;
5
6  import ifmt.cba.persistencia.FabricaEntityManager;
7  import jakarta.persistence.EntityManager;
8
9  public class FabricaEntityManagerIntegracaoTest {
10
11     @Test
12     public void testValidarConexaoBancoProducao(){
13         EntityManager em = FabricaEntityManager.getEntityManagerProducao();
14         Assertions.assertNotNull(em);
15         em.close();
16     }
17
18     @Test
19     public void testValidarConexaoBancoTeste(){
20         EntityManager em = FabricaEntityManager.getEntityManagerTeste();
21         Assertions.assertNotNull(em);
22         em.close();
23     }
24 }
```

Testes de Integração - GrupoAlimentarDAO e Banco (1)

Classe GrupoAlimentarDAO responsavel realizar as operações de persistência da entidade GrupoAlimentar.

```java
package ifmt.cba.persistencia;

import java.util.List;
import ifmt.cba.entity.GrupoAlimentar;
import jakarta.persistence.EntityManager;
import jakarta.persistence.Query;


public class GrupoAlimentarDAO extends DAO<GrupoAlimentar> {

  public GrupoAlimentarDAO(EntityManager entityManager) throws PersistenciaException {
    super(entityManager);
  }

  public GrupoAlimentar buscarPorCodigo(int codigo) throws PersistenciaException {

    GrupoAlimentar grupoAlimentar = null;

    try {
      grupoAlimentar = this.entityManager.find(GrupoAlimentar.class, codigo);
    } catch (Exception ex) {
      throw new PersistenciaException("Erro na selecao por codigo - " + ex.getMessage());
    }
    return grupoAlimentar;
  }

  @SuppressWarnings("unchecked")
```

Testes de Integração - GrupoAlimentarDAO e Banco (2)

```
28    public List<GrupoAlimentar> buscarPorParteNome(String nome) throws PersistenciaException {
29      List<GrupoAlimentar> listaGrupoAlimentar;
30      try {
31        Query query = this.entityManager
32            .createQuery("SELECT ga FROM GrupoAlimentar ga WHERE UPPER(ga.nome) LIKE :pNome ORDER BY ga.nome");
33        query.setParameter("pNome", "%" + nome.toUpperCase().trim() + "%");
34        listaGrupoAlimentar = query.getResultList();
35      } catch (Exception ex) {
36        throw new PersistenciaException("Erro na selecao por parte do nome - " + ex.getMessage());
37      }
38      return listaGrupoAlimentar;
39    }
40  }
```

Em testes que envolvem dados em Banco de Dados, uma boa prática e não considerar dados de execuções anteriores ao teste no momento.

Para isso podemos:

- Usar o controle de transações (rollback) para não confirmar os dados de um teste.
- Usar a opção para recriar a base em cada contexto de persistência criado.

Testes de Integração - GrupoAlimentarDAOTest1 (1)

Classe GrupoAlimentarDAOTest responsável automatizar o teste de integração, que verifica se as operações básicas de persitência estão ocorrendo corretamente.

```
1  package ifmt.cba.integracao;
2
3  import java.util.List;
4
5  import org.junit.jupiter.api.Assertions;
6  import org.junit.jupiter.api.Test;
7
8  import ifmt.cba.entity.GrupoAlimentar;
9  import ifmt.cba.persistencia.FabricaEntityManager;
10 import ifmt.cba.persistencia.GrupoAlimentarDAO;
11
12 public class GrupoAlimentarDAOIntegracaoTest1 {
13
14     @Test
15     public void testOperacaoInclusaoIntegracaoDAOBanco1(){
16
17         GrupoAlimentarDAO grupoDAO = Assertions.assertDoesNotThrow(
18             ()-> new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
19
20         GrupoAlimentar grupoAlimentar = new GrupoAlimentar();
21         grupoAlimentar.setNome("Teste Integracao Banco");
22         grupoDAO.beginTransaction();
23         Assertions.assertDoesNotThrow(()->grupoDAO.incluir(grupoAlimentar));
24         grupoDAO.commitTransaction();
25     }
26
27     @Test
```

Testes de Integração - GrupoAlimentarDAOTest1 (2)

```
28      public void testOperacaoInclusaoIntegracaoDAOBanco2(){
29
30          GrupoAlimentarDAO grupoDAO = Assertions.assertDoesNotThrow(
31              ()-> new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
32
33          GrupoAlimentar grupoAlimentar = new GrupoAlimentar();
34          grupoAlimentar.setNome("Teste Integracao Banco");
35          grupoDAO.beginTransaction();
36          Assertions.assertDoesNotThrow(()->grupoDAO.incluir(grupoAlimentar));
37          List<GrupoAlimentar> lista = Assertions.assertDoesNotThrow(
38              ()-> grupoDAO.buscarPorParteNome("Teste Integracao Banco"));
39          Assertions.assertFalse(lista.isEmpty());
40          grupoDAO.commitTransaction();
41      }
42  }
```

Testes de Integração - GrupoAlimentarDAOTest2 (1)

Usando RollBack para os dados de um caso de teste não interferir no outro.

```
1  package ifmt.cba.integracao;
2
3  import java.util.List;
4
5  import org.junit.jupiter.api.Assertions;
6  import org.junit.jupiter.api.Test;
7
8  import ifmt.cba.entity.GrupoAlimentar;
9  import ifmt.cba.persistencia.FabricaEntityManager;
10 import ifmt.cba.persistencia.GrupoAlimentarDAO;
11
12 public class GrupoAlimentarDAOIntegracaoTest2 {
13
14     @Test
15     public void testOperacaoInclusaoIntegracaoDAOBanco1(){
16
17         GrupoAlimentarDAO grupoDAO = Assertions.assertDoesNotThrow(
18             ()-> new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
19
20         GrupoAlimentar grupoAlimentar = new GrupoAlimentar();
21         grupoAlimentar.setNome("Teste Integracao Banco");
22         grupoDAO.beginTransaction();
23         Assertions.assertDoesNotThrow(()->grupoDAO.incluir(grupoAlimentar));
24         grupoDAO.rollbackTransaction();
25     }
26
27     @Test
28     public void testOperacaoInclusaoIntegracaoDAOBanco2(){
29
```

Testes de Integração - GrupoAlimentarDAOTest2 (2)

```
30          GrupoAlimentarDAO grupoDAO = Assertions.assertDoesNotThrow(
31              ()-> new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
32
33          GrupoAlimentar grupoAlimentar = new GrupoAlimentar();
34          grupoAlimentar.setNome("Teste Integracao Banco");
35          grupoDAO.beginTransaction();
36          Assertions.assertDoesNotThrow(()->grupoDAO.incluir(grupoAlimentar));
37          List<GrupoAlimentar> lista = Assertions.assertDoesNotThrow(
38              ()-> grupoDAO.buscarPorParteNome("Teste Integracao Banco"));
39          Assertions.assertFalse(lista.isEmpty());
40          grupoDAO.rollbackTransaction();
41      }
42  }
```

Testes de Integração - GrupoAlimentarNegocio/DAO/Banco (1)

Classe GrupoAlimentarNegocio responsavel realizar as operações de verificaçao de consistência e regras de negócio, e por fim invocar a persistência.

```
1  package ifmt.cba.negocio;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  import org.modelmapper.ModelMapper;
7
8  import ifmt.cba.dto.GrupoAlimentarDTO;
9  import ifmt.cba.entity.GrupoAlimentar;
10 import ifmt.cba.persistencia.GrupoAlimentarDAO;
11 import ifmt.cba.persistencia.PersistenciaException;
12 import ifmt.cba.persistencia.ProdutoDAO;
13
14 public class GrupoAlimentarNegocio {
15
16   private ModelMapper modelMapper;
17   private GrupoAlimentarDAO grupoAlimentarDAO;
18   private ProdutoDAO produtoDAO;
19
20   public GrupoAlimentarNegocio(GrupoAlimentarDAO grupoAlimentarDAO, ProdutoDAO produtoDAO) throws NegocioException {
21
22     this.grupoAlimentarDAO = grupoAlimentarDAO;
23     this.produtoDAO = produtoDAO;
24
25     this.modelMapper = new ModelMapper();
26   }
27
```

Testes de Integração - GrupoAlimentarNegocio/DAO/Banco (2)

```
28   public void inserir(GrupoAlimentarDTO grupoAlimentarDTO) throws NegocioException {
29
30     GrupoAlimentar grupoAlimentar = this.toEntity(grupoAlimentarDTO);
31     String mensagemErros = grupoAlimentar.validar();
32
33     if (!mensagemErros.isEmpty()) {
34       throw new NegocioException(mensagemErros);
35     }
36
37     try {
38       // nao pode existir outro com o mesmo nome
39       if (!grupoAlimentarDAO.buscarPorParteNome(grupoAlimentar.getNome()).isEmpty()) {
40         throw new NegocioException("Ja existe esse grupo alimentar");
41       }
42       grupoAlimentarDAO.beginTransaction();
43       grupoAlimentarDAO.incluir(grupoAlimentar);
44       grupoAlimentarDAO.commitTransaction();
45     } catch (PersistenciaException ex) {
46       grupoAlimentarDAO.rollbackTransaction();
47       throw new NegocioException("Erro ao incluir o grupo alimentar - " + ex.getMessage());
48     }
49   }
50
51   public void alterar(GrupoAlimentarDTO grupoAlimentarDTO) throws NegocioException {
52
53     GrupoAlimentar grupoAlimentar = this.toEntity(grupoAlimentarDTO);
54     String mensagemErros = grupoAlimentar.validar();
55     if (!mensagemErros.isEmpty()) {
56       throw new NegocioException(mensagemErros);
57     }
58     try {
59       // deve existir para ser alterado
```

Testes de Integração - GrupoAlimentarNegocio/DAO/Banco (3)

```
60        if (grupoAlimentarDAO.buscarPorCodigo(grupoAlimentar.getCodigo()) == null) {
61          throw new NegocioException("Nao existe esse grupo alimentar");
62        }
63        grupoAlimentarDAO.beginTransaction();
64        grupoAlimentarDAO.alterar(grupoAlimentar);
65        grupoAlimentarDAO.commitTransaction();
66      } catch (PersistenciaException ex) {
67        grupoAlimentarDAO.rollbackTransaction();
68        throw new NegocioException("Erro ao alterar o grupo alimentar - " + ex.getMessage());
69      }
70    }
71
72    public void excluir(int codigo) throws NegocioException {
73
74      try {
75        GrupoAlimentar grupoAlimentar = grupoAlimentarDAO.buscarPorCodigo(codigo);
76        if( grupoAlimentar == null){
77          throw new NegocioException("Esse GrupoAlimentar nao existe");
78        }
79
80        // nao pode excluir se estiver sendo referenciado por um ou mais produtos
81        if (produtoDAO.buscarPorGrupoAlimentar(grupoAlimentar.getCodigo()).size() > 0) {
82          throw new NegocioException("Grupo Alimentar esta relacionado a produtos");
83        }
84        grupoAlimentarDAO.beginTransaction();
85        grupoAlimentarDAO.excluir(grupoAlimentar);
86        grupoAlimentarDAO.commitTransaction();
87      } catch (PersistenciaException ex) {
88        grupoAlimentarDAO.rollbackTransaction();
89        throw new NegocioException("Erro ao excluir o grupo alimentar - " + ex.getMessage());
90      }
91    }
```

Testes de Integração - GrupoAlimentarNegocio/DAO/Banco (4)

```
92
93    public List<GrupoAlimentarDTO> pesquisaParteNome(String parteNome) throws NegocioException {
94      try {
95
96        return this.toDTOAll(grupoAlimentarDAO.buscarPorParteNome(parteNome));
97      } catch (PersistenciaException ex) {
98        throw new NegocioException("Erro ao pesquisar grupo alimentar pelo nome - " + ex.getMessage());
99      }
100   }
101
102   public GrupoAlimentarDTO pesquisaCodigo(int codigo) throws NegocioException {
103     try {
104       return this.toDTO(grupoAlimentarDAO.buscarPorCodigo(codigo));
105     } catch (PersistenciaException ex) {
106       throw new NegocioException("Erro ao pesquisar grupo alimentar pelo codigo - " + ex.getMessage());
107     }
108   }
109
110   public List<GrupoAlimentarDTO> toDTOAll(List<GrupoAlimentar> listaGrupoAlimentar) {
111     List<GrupoAlimentarDTO> listDTO = new ArrayList<GrupoAlimentarDTO>();
112
113     for (GrupoAlimentar grupoAlimentar : listaGrupoAlimentar) {
114       listDTO.add(this.toDTO(grupoAlimentar));
115     }
116     return listDTO;
117   }
118
119   public GrupoAlimentarDTO toDTO(GrupoAlimentar grupoAlimentar) {
120     return this.modelMapper.map(grupoAlimentar, GrupoAlimentarDTO.class);
121   }
122
123   public GrupoAlimentar toEntity(GrupoAlimentarDTO grupoAlimentarDTO) {
```

Testes de Integração - GrupoAlimentarNegocio/DAO/Banco (5)

```
124         return this.modelMapper.map(grupoAlimentarDTO, GrupoAlimentar.class);
125     }
126
127 }
```

Testes de Integração - GrupoAlimentarNegocioIntegracaoTest (1)

Classe GrupoAlimentarNegocioIntegracaoTest responsavel automatizar o teste de integração, que verifica se as operações básicas entre a camada de negócio e persistência, chegando ao banco.

```
1  package ifmt.cba.integracao;
2
3  import java.util.List;
4
5  import org.junit.jupiter.api.Assertions;
6  import org.junit.jupiter.api.Test;
7
8  import ifmt.cba.dto.GrupoAlimentarDTO;
9  import ifmt.cba.negocio.GrupoAlimentarNegocio;
10 import ifmt.cba.persistencia.FabricaEntityManager;
11 import ifmt.cba.persistencia.GrupoAlimentarDAO;
12 import ifmt.cba.persistencia.ProdutoDAO;
13
14 public class GrupoAlimentarNegocioIntegracaoTest {
15
16     @Test
17     public void testOperacaoInclusaoIntegracaoNegocioDAO1(){
18
19         GrupoAlimentarDAO grupoAlimentarDAO = Assertions.assertDoesNotThrow(
20             ()->new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
21         ProdutoDAO produtoDAO = Assertions.assertDoesNotThrow(
22             ()->new ProdutoDAO(FabricaEntityManager.getEntityManagerTeste()));
23         GrupoAlimentarNegocio grupoNegocio = Assertions.assertDoesNotThrow(
24             ()-> new GrupoAlimentarNegocio(grupoAlimentarDAO, produtoDAO));
25
26         GrupoAlimentarDTO grupoDTO = new GrupoAlimentarDTO();
```

Testes de Integração - GrupoAlimentarNegocioIntegracaoTest (2)

```
27          grupoDTO.setNome("Teste Integracao Negocio -> DAO");
28          Assertions.assertDoesNotThrow(()->grupoNegocio.inserir(grupoDTO));
29      }
30
31      @Test
32      public void testOperacaoInclusaoIntegracaoNegocioDAO2(){
33
34          GrupoAlimentarDAO grupoAlimentarDAO = Assertions.assertDoesNotThrow(
35              ()->new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerTeste()));
36          ProdutoDAO produtoDAO = Assertions.assertDoesNotThrow(
37              ()->new ProdutoDAO(FabricaEntityManager.getEntityManagerTeste()));
38          GrupoAlimentarNegocio grupoNegocio = Assertions.assertDoesNotThrow(
39              ()-> new GrupoAlimentarNegocio(grupoAlimentarDAO, produtoDAO));
40
41          GrupoAlimentarDTO grupoDTO = new GrupoAlimentarDTO();
42          grupoDTO.setNome("Teste Integracao Negocio -> DAO");
43          Assertions.assertDoesNotThrow(()->grupoNegocio.inserir(grupoDTO));
44
45          List<GrupoAlimentarDTO> lista = Assertions.assertDoesNotThrow(
46              ()->grupoNegocio.pesquisaParteNome("Teste Integracao Negocio -> DAO"));
47
48          Assertions.assertFalse(lista.isEmpty());
49      }
50  }
```

Testes de Integração - Classe de Negócio X DAO

Considerações:

- Fazer teste de integração entre DAO (persitência) e Banco
- Fazer teste unitário da classe de Negócio
- Não fazer o teste de integração entre Negócio e DAO
- Testes Unitário (classes que envolvam regras de negócio)
- Testes de Integração (classes que envolvam comunicação com algo externo)

Testes de Integração da API de Serviços

Para realizar os testes da API, usaremos o framework JUnit, API RestAssured e os Matchers da API Hamcrest.

A API RestAssured possibilita a geração de requisições e comparações com os retorno de uma forma bem prática e produtiva.

Outra característica da RestAssured é que ela foi projetada para trabalhar com o padrão GivenWhenThen, facilitando a estruturação dos casos de testes.

A API Hamcrest já vem como dependência automática do JUnit.

```
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.3.1</version>
    <scope>test</scope>
</dependency>
```

API de Serviços - GrupoAlimentarServico (1)

Classe que implementa o serviço Rest do GrupoAlimentar

```
1  package ifmt.cba.servico;
2
3  import java.util.List;
4
5  import ifmt.cba.dto.GrupoAlimentarDTO;
6  import ifmt.cba.negocio.GrupoAlimentarNegocio;
7  import ifmt.cba.negocio.NegocioException;
8  import ifmt.cba.persistencia.FabricaEntityManager;
9  import ifmt.cba.persistencia.GrupoAlimentarDAO;
10 import ifmt.cba.persistencia.PersistenciaException;
11 import ifmt.cba.persistencia.ProdutoDAO;
12 import jakarta.ws.rs.Consumes;
13 import jakarta.ws.rs.DELETE;
14 import jakarta.ws.rs.GET;
15 import jakarta.ws.rs.POST;
16 import jakarta.ws.rs.PUT;
17 import jakarta.ws.rs.Path;
18 import jakarta.ws.rs.PathParam;
19 import jakarta.ws.rs.Produces;
20 import jakarta.ws.rs.core.MediaType;
21 import jakarta.ws.rs.core.Response;
22 import jakarta.ws.rs.core.Response.ResponseBuilder;
23
24 @Path("/grupoalimentar")
25 public class GrupoAlimentarServico {
26
27     private static GrupoAlimentarNegocio grupoAlimentarNegocio;
28
29     private static GrupoAlimentarDAO grupoAlimentarDAO;
```

API de Serviços - GrupoAlimentarServico (2)

```
30      private static ProdutoDAO produtoDAO;
31
32      static {
33          try {
34              grupoAlimentarDAO = new GrupoAlimentarDAO(FabricaEntityManager.getEntityManagerProducao());
35              produtoDAO = new ProdutoDAO(FabricaEntityManager.getEntityManagerProducao());
36              grupoAlimentarNegocio = new GrupoAlimentarNegocio(grupoAlimentarDAO, produtoDAO);
37          } catch (PersistenciaException | NegocioException e) {
38              e.printStackTrace();
39          }
40      }
41
42      @POST
43      @Consumes(MediaType.APPLICATION_JSON)
44      @Produces(MediaType.APPLICATION_JSON)
45      public Response adicionar(GrupoAlimentarDTO grupoAlimentarDTO) {
46          ResponseBuilder resposta;
47          try {
48              grupoAlimentarNegocio.inserir(grupoAlimentarDTO);
49              resposta = Response.ok();
50              resposta.entity(grupoAlimentarNegocio.pesquisaParteNome(grupoAlimentarDTO.getNome()).get(0));
51          } catch (Exception ex) {
52              resposta = Response.status(400);
53              resposta.entity("{\"erro\": \""+ex.getMessage()+"\"}");
54          }
55          return resposta.build();
56      }
57
58      @PUT
59      @Consumes(MediaType.APPLICATION_JSON)
60      @Produces(MediaType.APPLICATION_JSON)
61      public Response alterar(GrupoAlimentarDTO grupoAlimentarDTO) {
```

API de Serviços - GrupoAlimentarServico (3)

```
62          ResponseBuilder resposta;
63          try {
64              grupoAlimentarNegocio.alterar(grupoAlimentarDTO);
65              resposta = Response.ok();
66              resposta.entity(grupoAlimentarNegocio.pesquisaCodigo(grupoAlimentarDTO.getCodigo()));
67          } catch (Exception ex) {
68              resposta = Response.status(400);
69              resposta.entity("{\"erro\": \""+ex.getMessage()+"\"}");
70          }
71          return resposta.build();
72      }
73
74      @DELETE
75      @Path("/{codigo}")
76      @Produces(MediaType.APPLICATION_JSON)
77      public Response excluir(@PathParam("codigo") int codigo) {
78          ResponseBuilder resposta;
79          try {
80              grupoAlimentarNegocio.excluir(codigo);
81              resposta = Response.ok();
82          } catch (Exception ex) {
83              resposta = Response.status(400);
84              resposta.entity("{\"erro\": \""+ex.getMessage()+"\"}");
85          }
86          return resposta.build();
87      }
88
89      @GET
90      @Path("/codigo/{codigo}")
91      @Produces(MediaType.APPLICATION_JSON)
92      public Response buscarGrupoAlimentarPorCodigo(@PathParam("codigo") int codigo) {
93          ResponseBuilder resposta;
```

API de Serviços - GrupoAlimentarServico (4)

```
94          try {
95              GrupoAlimentarDTO grupoAlimentarDTO = grupoAlimentarNegocio.pesquisaCodigo(codigo);
96              resposta = Response.ok();
97              resposta.entity(grupoAlimentarDTO);
98          } catch (Exception ex) {
99              resposta = Response.status(400);
100             resposta.entity("{\"erro\": \""+ex.getMessage()+"\"}");
101         }
102         return resposta.build();
103     }
104
105     @GET
106     @Path("/nome/{nome}")
107     @Produces(MediaType.APPLICATION_JSON)
108     public Response buscarGrupoAlimentarPorNome(@PathParam("nome") String nome) {
109         ResponseBuilder resposta;
110         try {
111             List<GrupoAlimentarDTO> listaGrupoAlimentarDTO = grupoAlimentarNegocio.pesquisaParteNome(nome);
112             resposta = Response.ok();
113             resposta.entity(listaGrupoAlimentarDTO);
114         } catch (Exception ex) {
115             resposta = Response.status(400);
116             resposta.entity("{\"erro\": \""+ex.getMessage()+"\"}");
117         }
118         return resposta.build();
119     }
120 }
```

Testes de Integração da API de Serviços (1)

Testando o serviço GrupoAlimentarServico

```java
package ifmt.cba;

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

import io.restassured.RestAssured;
import io.restassured.http.Method;
import io.restassured.response.Response;
import io.restassured.response.ValidatableResponse;

public class GrupoAlimentarServicoTest1 {

    @Test
    public void testConsultarPorCodigo1(){

        Response response = RestAssured.request(Method.GET, "http://localhost:8080/grupoalimentar/codigo/1");
        Assertions.assertEquals(200, response.getStatusCode());

    }

    @Test
    public void testConsultarPorCodigo2(){

        ValidatableResponse validacao = RestAssured.get("http://localhost:8080/grupoalimentar/codigo/1").then();
        validacao.statusCode(200);
    }

    @Test
    public void testConsultarPorCodigo3(){
```

Testes de Integração da API de Serviços (2)

```
30
31          RestAssured.get("http://localhost:8080/grupoalimentar/codigo/1")
32          .then()
33          .statusCode(200);
34      }
35
36      @Test
37      public void testConsultarPorCodigo4(){
38          RestAssured
39          .given()
40          .when()
41              .get("http://localhost:8080/grupoalimentar/codigo/1")
42          .then()
43              .statusCode(200);
44      }
45  }
```

Testes de Integração da API de Serviços (1)

Testando o Serviço GrupoAlimentarServico

```
1  package ifmt.cba;
2
3  import org.hamcrest.Matchers;
4  import org.junit.jupiter.api.Assertions;
5  import org.junit.jupiter.api.Test;
6
7  import io.restassured.RestAssured;
8  import io.restassured.http.Method;
9  import io.restassured.path.json.JsonPath;
10 import io.restassured.response.Response;
11
12 public class GrupoAlimentarServicoTest2 {
13
14     @Test
15     public void testConsultarPorCodigoVerificandoValores1(){
16         RestAssured
17         .given()
18         .when()
19             .get("http://localhost:8080/grupoalimentar/codigo/1")
20         .then()
21             .statusCode(200)
22             .body("codigo", Matchers.is(1))
23             .body("nome", Matchers.is("Carboidrato"))
24         ;
25     }
26
27     @Test
28     public void testConsultarPorCodigoVerificandoValores2(){
29         RestAssured
```

Testes de Integração da API de Serviços (2)

```
30          .given()
31          .when()
32              .get("http://localhost:8080/grupoalimentar/codigo/1")
33          .then()
34              .statusCode(200)
35              .body("codigo", Matchers.greaterThan(0))
36              .body("nome", Matchers.not(Matchers.emptyString()));
37      }
38
39      @Test
40      public void testConsultarPorCodigoVerificandoValores3(){
41          Response resposta = RestAssured.request(Method.GET, "http://localhost:8080/grupoalimentar/codigo/1");
42
43          Assertions.assertEquals(Integer.valueOf(1), resposta.path("codigo"));
44          Assertions.assertEquals("Carboidrato", resposta.path("nome"));
45      }
46
47      @Test
48      public void testConsultarPorCodigoVerificandoValores4(){
49          Response resposta = RestAssured.request(Method.GET, "http://localhost:8080/grupoalimentar/codigo/1");
50
51          JsonPath jsonPath = new JsonPath(resposta.asString());
52
53          Assertions.assertEquals(1, jsonPath.getInt("codigo"));
54          Assertions.assertEquals("Carboidrato", jsonPath.getString("nome"));
55      }
56  }
```

Testes de Integração da API de Serviços (1)

Testando o Serviço GrupoAlimentarServico

```
1  package ifmt.cba;
2
3  import org.hamcrest.Matchers;
4  import org.junit.jupiter.api.Test;
5
6  import io.restassured.RestAssured;
7
8  public class GrupoAlimentarServicoTest3 {
9
10     @Test
11     public void testInclusaoComDadosCorretos1(){
12
13         RestAssured
14         .given()
15             .log().all()
16             .contentType("application/json")
17             .body("{\"nome\":\"Inclusao pelo Teste\"}")
18         .when()
19             .post("http://localhost:8080/grupoalimentar/")
20         .then()
21             .log().all()
22             .statusCode(200)
23             .body("codigo",Matchers.is(Matchers.notNullValue()))
24             .body("nome",Matchers.is("Inclusao pelo Teste"));
25     }
26
27     @Test
28     public void testInclusaoComDadosIncorretos2(){
29
```

Testes de Integração da API de Serviços (2)

```
30          RestAssured
31          .given()
32              .log().all()
33              .contentType("application/json")
34              .body("{}")
35          .when()
36              .post("http://localhost:8080/grupoalimentar/")
37          .then()
38              .log().all()
39              .statusCode(400)
40              .body("erro",Matchers.is("Nome nao valido"));
41      }
42
43      @Test
44      public void testAlteracaoComDadosCorretos(){
45
46          RestAssured
47          .given()
48              .log().all()
49              .contentType("application/json")
50              .body("{\"codigo\": 1, \"nome\":\"Alteracao pelo Teste\"}")
51          .when()
52              .put("http://localhost:8080/grupoalimentar/")
53          .then()
54              .log().all()
55              .statusCode(200)
56              .body("codigo",Matchers.is(Matchers.notNullValue()))
57              .body("nome",Matchers.is("Alteracao pelo Teste"));
58      }
59
60      @Test
61      public void testAlteracaoComDadosInCorretos(){
```

Testes de Integração da API de Serviços (3)

```
62
63          RestAssured
64          .given()
65              .log().all()
66              .contentType("application/json")
67              .body("{\"codigo\": 1, \"nome\":\"AI\"}")
68          .when()
69              .put("http://localhost:8080/grupoalimentar/")
70          .then()
71              .log().all()
72              .statusCode(400)
73              .body("erro",Matchers.is("Nome nao valido"));
74      }
75
76      @Test
77      public void testExclusaoComDadosCorretos(){
78
79          RestAssured
80          .given()
81              .log().all()
82          .when()
83              .delete("http://localhost:8080/grupoalimentar/1")
84          .then()
85              .log().all()
86              .statusCode(200);
87      }
88
89      @Test
90      public void testExclusaoComDadosInCorretos(){
91
92          RestAssured
93          .given()
```

Testes de Integração da API de Serviços (4)

```
94              .log().all()
95          .when()
96              .delete("http://localhost:8080/grupoalimentar/12345")
97          .then()
98              .log().all()
99              .statusCode(400)
100             .body("erro",Matchers.is("Esse GrupoAlimentar nao existe"));
101     }
102 }
```

Testes de Integração da API de Serviços (1)

Testando o Serviço ProdutoServico

```java
package ifmt.cba;

import org.hamcrest.Matchers;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

import ifmt.cba.dto.GrupoAlimentarDTO;
import ifmt.cba.dto.ProdutoDTO;
import io.restassured.RestAssured;
import io.restassured.http.Method;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;

public class ProdutoServicoTest1 {

    @Test
    public void testConsultarPorCodigoVerificandoValores1() {
        Response resposta = RestAssured.request(Method.GET, "http://localhost:8080/produto/codigo/1");

        JsonPath jsonPath = new JsonPath(resposta.asString());
        Assertions.assertEquals(1, jsonPath.getInt("codigo"));
        Assertions.assertEquals("Alcatra bovina", jsonPath.getString("nome"));
        Assertions.assertEquals("Proteinas", jsonPath.getString("grupoAlimentar.nome"));
    }

    @Test
    public void testConsultarPorCodigoVerificandoValores2() {

        RestAssured
```

Testes de Integração da API de Serviços (2)

```
30                  .given()
31                  .when()
32                  .get("http://localhost:8080/produto/estoquebaixo")
33                  .then()
34                  .statusCode(200)
35                  .body("$", Matchers.hasSize(2))
36                  .body("nome", Matchers.hasItems("Batata Doce", "Costela suina"))
37                  .body("estoque[0]", Matchers.is(100));
38      }
39
40      @Test
41      public void testInclusaoComDadosCorretos() {
42
43          Response response = RestAssured.request(Method.GET, "http://localhost:8080/grupoalimentar/codigo/1");
44          Assertions.assertEquals(200, response.getStatusCode());
45          GrupoAlimentarDTO grupoAlimentarDTO = response.getBody().as(GrupoAlimentarDTO.class);
46
47          ProdutoDTO produtoDTO = new ProdutoDTO();
48          produtoDTO.setNome("Produto pela api");
49          produtoDTO.setCustoUnidade(3.0f);
50          produtoDTO.setEstoque(100);
51          produtoDTO.setEstoqueMinimo(10);
52          produtoDTO.setValorEnergetico(50);
53          produtoDTO.setGrupoAlimentar(grupoAlimentarDTO);
54
55          RestAssured
56              .given()
57                  .log().all()
58                  .contentType("application/json")
59                  .body(produtoDTO)
60              .when()
61                  .post("http://localhost:8080/produto/")
```

Testes de Integração da API de Serviços (3)

```
62              .then()
63                  .log().all()
64                  .statusCode(200)
65                  .body("codigo", Matchers.is(Matchers.notNullValue()));
66      }
67
68      @Test
69      public void testAlteracaoComDadosCorretos() {
70
71          Response response = RestAssured.request(Method.GET, "http://localhost:8080/produto/codigo/1");
72          Assertions.assertEquals(200, response.getStatusCode());
73          ProdutoDTO produtoDTO = response.getBody().as(ProdutoDTO.class);
74
75          int estoqueAtual = produtoDTO.getEstoque();
76          produtoDTO.setEstoque(estoqueAtual - 20);
77
78          RestAssured
79              .given()
80                  .log().all()
81                  .contentType("application/json")
82                  .body(produtoDTO)
83              .when()
84                  .put("http://localhost:8080/produto/")
85              .then()
86                  .log().all()
87                  .statusCode(200)
88                  .body("codigo", Matchers.is(Matchers.notNullValue()))
89                  .body("estoque", Matchers.is(estoqueAtual-20));
90      }
91  }
```