Teste de Software

Testes automatizados utilizando a ferramenta Cucumber.

Membros do Grupo:

Bruno Barbosa
Lorena Salazar
Lucas Giacomin
Matheus Barbosa
Matheus Marmo
Renan Santana

Introdução:

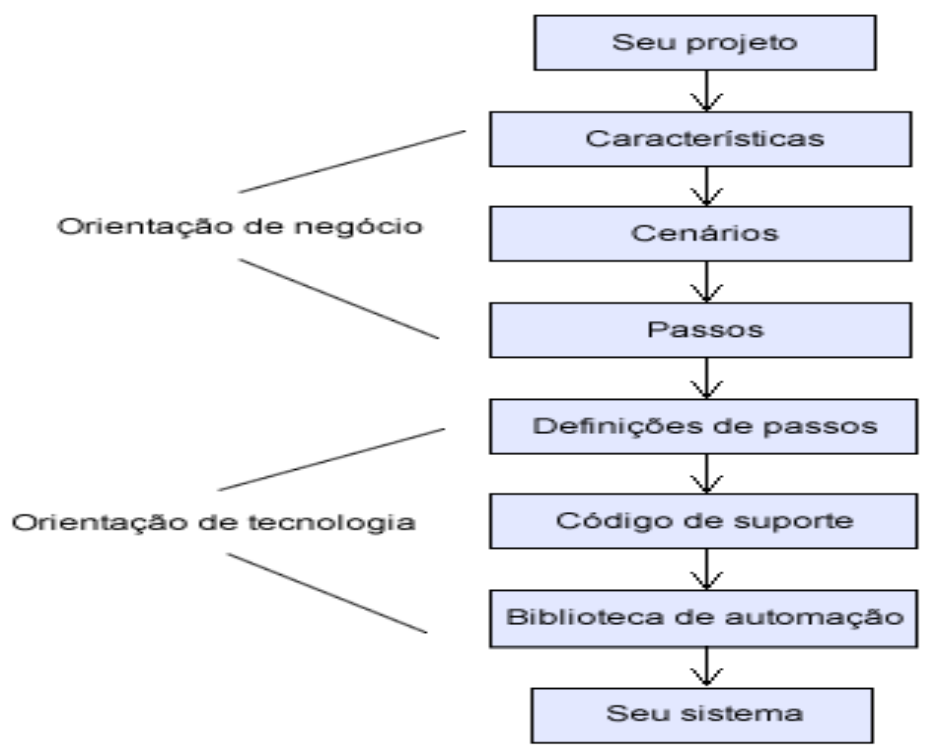
- Nessa apresentação utilizaremos a ferramenta Cucumber para demonstrar o processo de testes automatizados, baseado em nossa pesquisa e também conhecimentos estudados na disciplina (Teste de Software).

- O Cucumber é uma ferramenta que auxilia no desenvolvimento orientado ao comportamento (BDD).

Definição:

- O Cucumber foi originalmente criado por membros da comunidade Ruby para apoiar o desenvolvimento de testes de aceitação automatizado utilizando a técnica BDD, com isso, é possível descrever a real necessidade dos usuário de um determinado sistema, com a intenção de que técnicos e não técnicos entendam os fluxos de testes e trabalhos.

Visão geral:



1) Descrever um comportamento em um texto simples;

2) Escrever uma definição dos passos em Java ou em outras linguagens;

3) Executar e visualizar os passos falharem;

4) Escrever o código para fazer os passos passar;

5) Se necessário, refatorar o código ou o comportamento descrito.

Projeto:

- Faremos o desenvolvimento de testes de aceitação de duas funcionalidades utilizando o Cucumber em Java.
- A primeira funcionalidade vai possibilitar que o usuário realize as operações de fazer saque e deposito utilizando sua conta.
- A segunda funcionalidade irá possibilitar o usuário realizar operações básicas no banco, como, obter o dinheiro total disponível no banco e obter o total de contas criadas.

Primeira Parte:

- O sistema só libera o saque só o valor deste for menor ou igual ao valor do saldo disponível na conta, e o sistema só libera o deposito se o valor deste for menor ou igual ao valor do limite disponível na conta.

Código:

testes-cucumber ▸ src/test/java ▸ cucumber.teste ▸ BancoTeste

```
1  package cucumber.teste;
2
3  import org.junit.runner.RunWith;
4  import cucumber.api.CucumberOptions;
5  import cucumber.api.junit.Cucumber;
6
7  @RunWith(Cucumber.class)
8  @CucumberOptions(features = "classpath:caracteristicas", tags = "@BancoTeste",
9                   glue = "cucumber.teste.passos", monochrome = true, dryRun = false)
10
11 public class BancoTeste {
12
13 }
14
```

testes-cucumber ▸ src/test/java ▸ cucumber.teste ▸ ContaTeste

```
1  package cucumber.teste;
2
3  import org.junit.runner.RunWith;
6
7  @RunWith(Cucumber.class)
8  @CucumberOptions(features = "classpath:caracteristicas", tags = "@ContaTeste",
9                   glue = "cucumber.teste.passos", monochrome = true, dryRun = false)
10
11 public class ContaTeste {
12
13 }
14
```

Segunda Parte:

- Primeiro foram criados os arquivos *features*, onde é uma usado uma descrição de alto nível para relatar como nossos testes deve se comportar, usamos uma linguagem padrão para especificação de testes de aceitação, a famosa linguagem “Gherkin”, do Cucumber.

Código:


```
# language: pt
@ContaTeste
Funcionalidade: Testar as operacoes basicas de conta
  O sistema deve prover o saque e deposito na conta de forma correta.
  Seguindo as seguintes restrições:
  1) Só libera o saque, se o valor do saque for menor ou igual ao valor
      do saldo disponível na conta
  2) Só libera o deposito, se o valor do deposito for menor ou igual ao
      valor do limite disponível na conta

  Esquema do Cenario: Testar saque e deposito
    Dado a conta criada para o dono "<dono>" de numero <numero> com o limite <limite> e saldo <saldo>
    Quando o dono realiza o deposito no valor de <deposito> na conta
    E o dono realiza o primeiro saque no valor de <primeiro_saque> na conta
    E o dono realiza o segundo saque no valor de <segundo_saque> na conta
    Entao o dono tem o saldo no valor de <saldo_esperado> na conta

Exemplos:
| dono |numero|limite     |saldo  |deposito     |primeiro_saque|segundo_saque |saldo_esperado  |
|Renan | 111  | 10000     | 1250  | 500         | 100          | 300          | 1350           |
|Julia | 222  | 10000     | 2000  | 200         | 400          | 200          | 1600           |
```

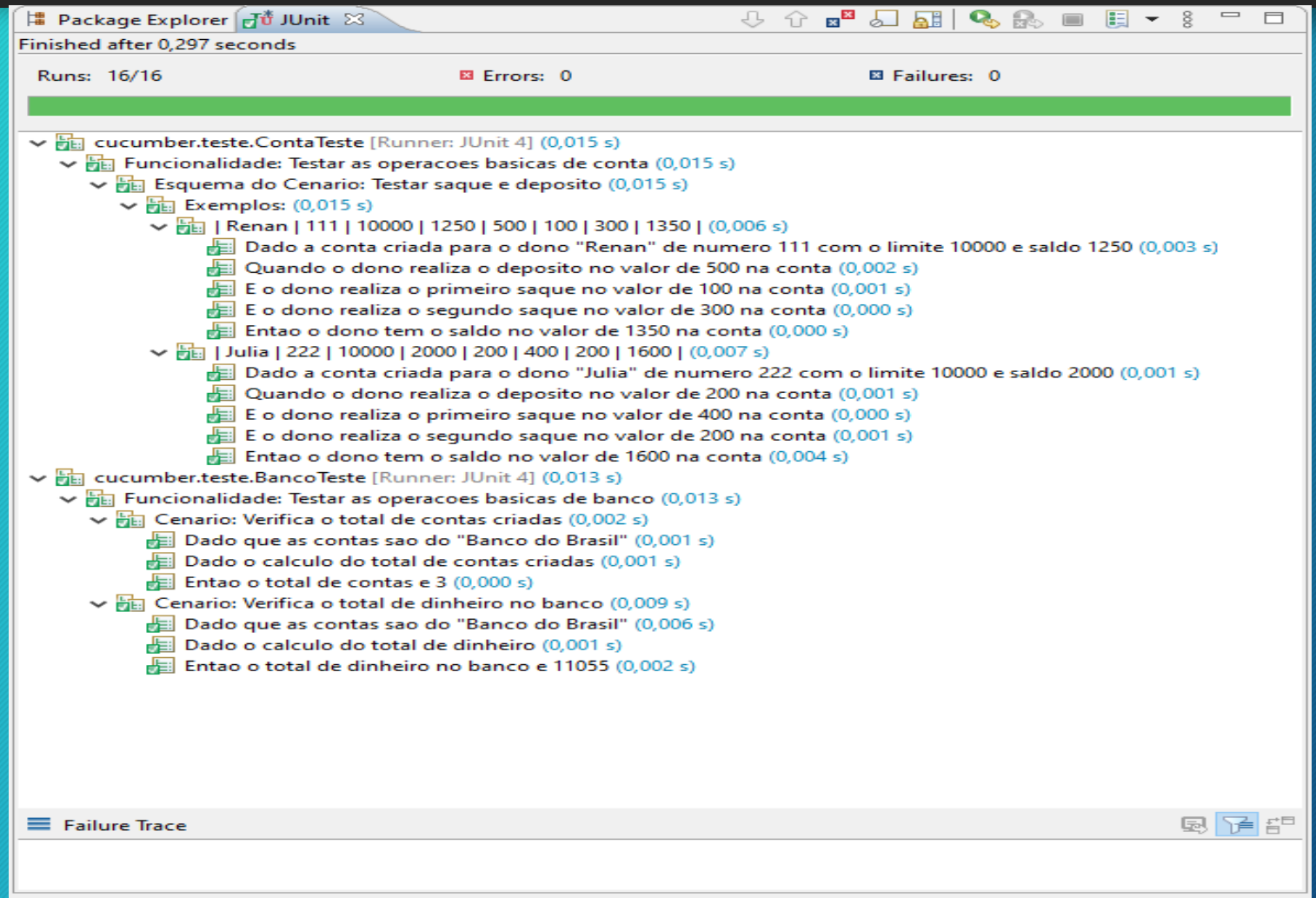
```
# language: pt
@BancoTeste
Funcionalidade: Testar as operacoes basicas de banco
  O sistema deve prover operações básicas de banco de forma correta.

  Contexto: Cria todas as contas e associa ao banco
    Dado que as contas sao do "Banco do Brasil"
      | dono               | numero | saldo  |
      | Renan Santana      | 111    | 2250   |
      | Matheus Barbosa    | 222    | 3360   |
      | Lucas Giacomin     | 333    | 5445   |


  Cenario: Verifica o total de contas criadas
    Dado o calculo do total de contas criadas
    Entao o total de contas e 3

  Cenario: Verifica o total de dinheiro no banco
    Dado o calculo do total de dinheiro
    Entao o total de dinheiro no banco e 11055
```

Resultado:


Package Explorer
JUnit
Finished after 0,297 seconds
Runs: 16/16
Errors: 0
Failures: 0
cucumber.teste.ContaTeste [Runner: JUnit 4] (0,015 s)
Funcionalidade: Testar as operacoes basicas de conta (0,015 s)
Esquema do Cenario: Testar saque e deposito (0,015 s)
Exemplos: (0,015 s)
| Renan | 111 | 10000 | 1250 | 500 | 100 | 300 | 1350 | (0,006 s)
Dado a conta criada para o dono "Renan" de numero 111 com o limite 10000 e saldo 1250 (0,003 s)
Quando o dono realiza o deposito no valor de 500 na conta (0,002 s)
E o dono realiza o primeiro saque no valor de 100 na conta (0,001 s)
E o dono realiza o segundo saque no valor de 300 na conta (0,000 s)
Entao o dono tem o saldo no valor de 1350 na conta (0,000 s)
| Julia | 222 | 10000 | 2000 | 200 | 400 | 200 | 1600 | (0,007 s)
Dado a conta criada para o dono "Julia" de numero 222 com o limite 10000 e saldo 2000 (0,001 s)
Quando o dono realiza o deposito no valor de 200 na conta (0,001 s)
E o dono realiza o primeiro saque no valor de 400 na conta (0,000 s)
E o dono realiza o segundo saque no valor de 200 na conta (0,001 s)
Entao o dono tem o saldo no valor de 1600 na conta (0,004 s)
cucumber.teste.BancoTeste [Runner: JUnit 4] (0,013 s)
Funcionalidade: Testar as operacoes basicas de banco (0,013 s)
Cenario: Verifica o total de contas criadas (0,002 s)
Dado que as contas sao do "Banco do Brasil" (0,001 s)
Dado o calculo do total de contas criadas (0,001 s)
Entao o total de contas e 3 (0,000 s)
Cenario: Verifica o total de dinheiro no banco (0,009 s)
Dado que as contas sao do "Banco do Brasil" (0,006 s)
Dado o calculo do total de dinheiro (0,001 s)
Entao o total de dinheiro no banco e 11055 (0,002 s)
Failure Trace

Conclusão:

- o Cucumber possui uso simplificado, baseado primariamente em texto puro, o que permite uma clara visualização dos problemas e do passo-a-passo de todo o processo e os caminhos lógicos para a obtenção dos resultados esperados.
- Por ser básico e seguir o padrão *Gherkin*, a implementação destes processos será mais fácil e inteligível para os desenvolvedores, além de posteriormente facilitar a documentação para futuros desenvolvedores, agilizando a metodologia de testes pois permite que a lógica de código seja testada de forma mais eficaz dentro dos limites estabelecidos e em concordância com as regras de negócio.
- Caso seja feita no processo de idealização de certa aplicação, poderá permitir que se analise os caminhos lógicos possíveis para cada aplicação e a sua implementação antes mesmo de qualquer parte de código ser escrita, o que possibilita a agilidade do projeto e ganhos exponenciais de produtividade.