BEHAVIOR DRIVEN DEVELOPMENT

Sumário

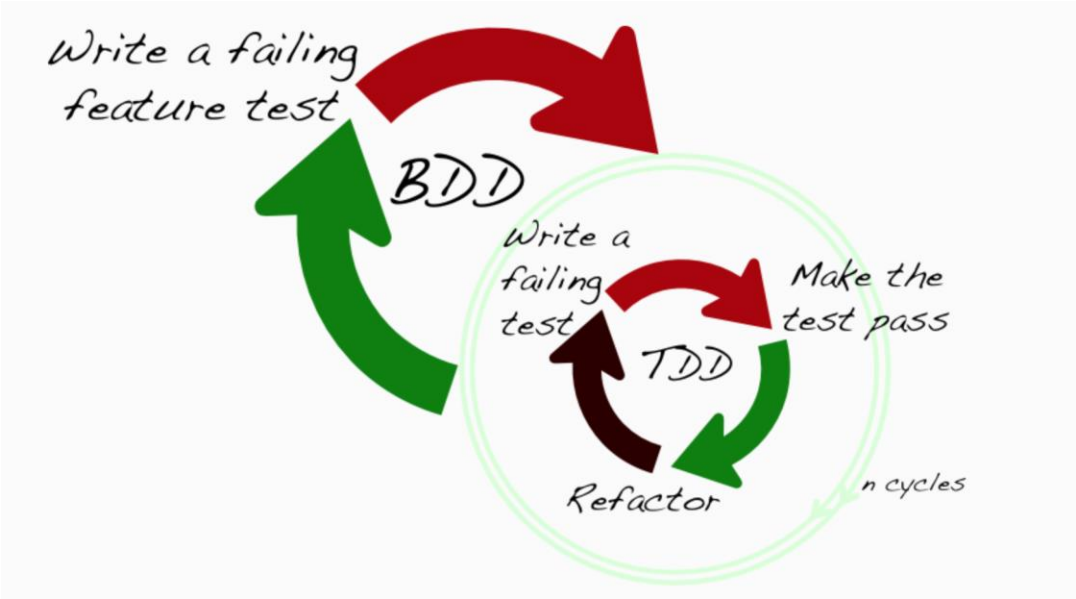
1. Conceitos e BDD
2. Boas Práticas
3. Ciclo de Vida do BDD

#1

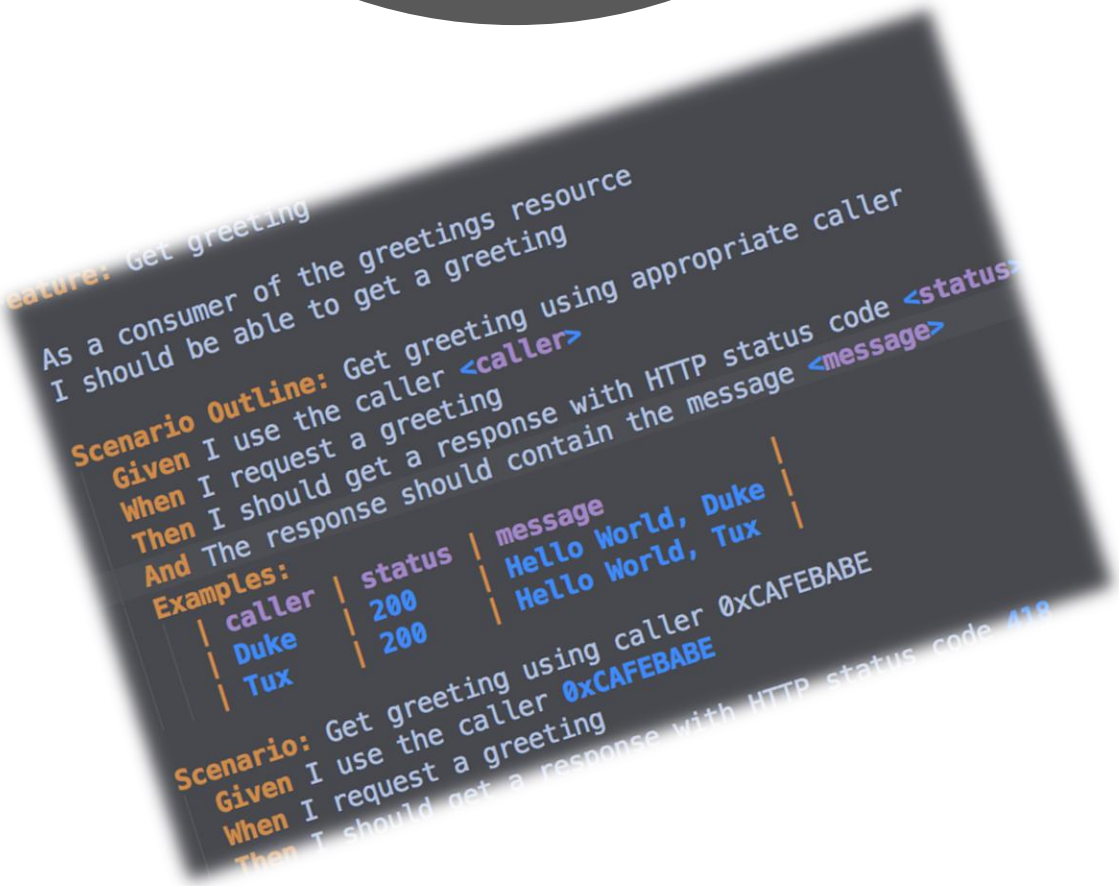
CONCEITOS E BDD

Conceitos

- **Test Driven Development (TDD)**
  - Encoraja uma **abordagem evolutiva** para o projeto e desenvolvimento de software.
  - Incentiva os programadores a **escrever testes antes de escrever o código**.
  - Os testes são o meio de **focar a atenção do programador no problema** que eles pretendem resolver e não na solução.
- **Behavior Driven Development (BDD)**
  - O software é especificado e projetado descrevendo **como seu comportamento deve ser**.
  - Tem como objetivo **concentrar** o desenvolvimento **na entrega de valor para o negócio**.
  - Baseia-se no uso de um vocabulário muito específico e comum (linguagem ubíqua) para **minimizar a falta de comunicação**.



Gherkin Language

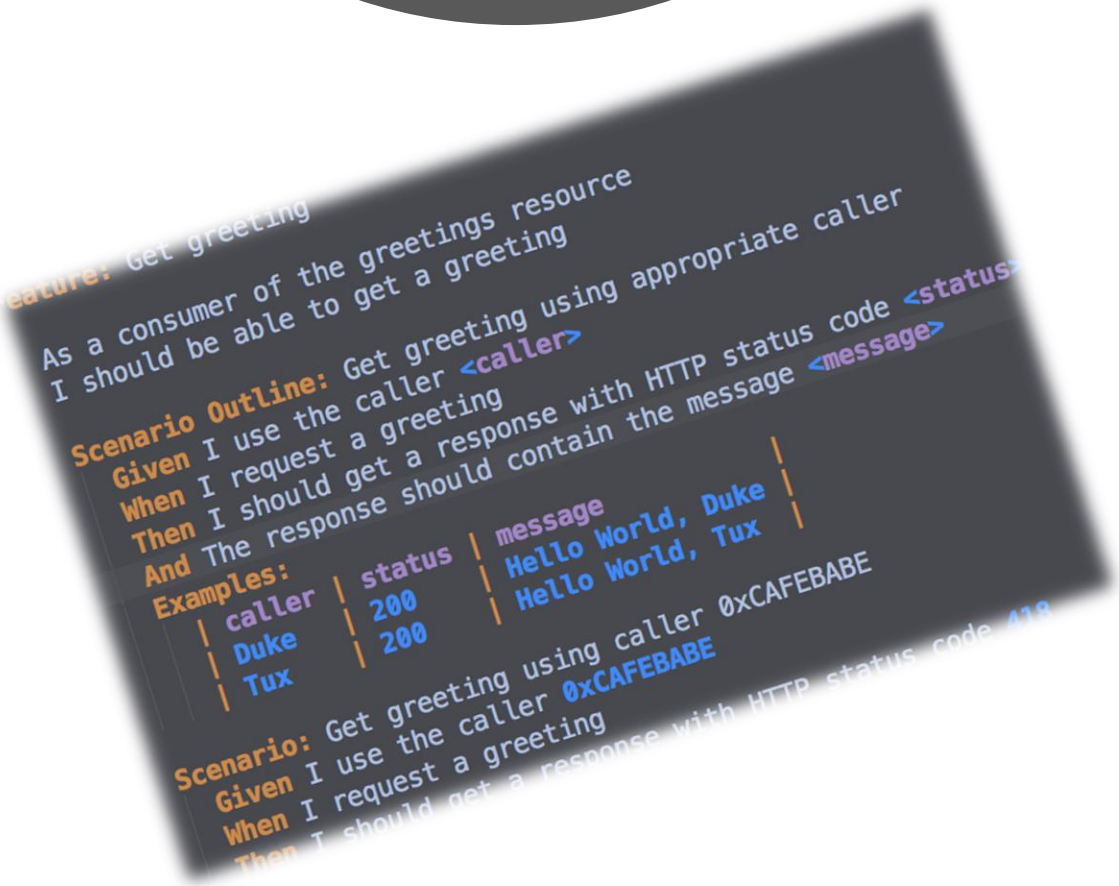


É uma linguagem de **texto simples** projetado para ser **fácil de aprender** por não-programadores...

...ainda que **estruturado o suficiente** para permitir a descrição concisa de exemplos para ilustrar as regras de negócios.

Permite descrever o comportamento do software sem detalhar como esse comportamento é implementado.

Gherkin Language



- É uma linguagem orientada a linha, ela usa indentação para definir a estrutura.
  - Os fins de linha encerram as declarações (denominados passos).
- Os espaços ou tabulações também podem ser usados para indentação (mas espaços são melhores para portabilidade).

A maioria das linhas iniciam com uma palavra-chave especial. As palavras-chave em Gherkin são:

- `Feature`
- `Scenario`
- `Given, When, Then, And, But (passos)`
- `Background`
- `Scenario Outline`
- `Examples`

Feature

- **`Features`** são:
  - Uma **narrativa**;
  - **NÃO** é **executável**;
  - Descreve o **VALOR DO NEGÓCIO**;
  - É o que o usuário necessita e espera ter quando o desenvolvimento for concluído.

- Cada arquivo ***.feature** consiste em uma funcionalidade única.

```
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta
```

```
Feature: <Título>
    As a (Como um) <tipo de usuário/papel>
    I want (Eu quero) <realizar uma ação>
    In order to (Para que) <possa atingir um objetivo - relacionado ao valor do negócio>
```

Scenario

- Uma **`Feature`** contém uma lista de **`Scenarios`**.

- Um **`Scenario`** é um exemplo concreto que ilustra uma regra de negócios.
- **`Scenario`** deve testar um comportamento único.

`Scenario:` <Título>

`<passos>`

```
Scenario: Autenticar informando dado válido
  Given Eu acesso a página de login
  When Eu preencho os campos com dados válidos
  And Eu clico em Login
  Then O sistema deve realizar o login com sucesso
```

Passos

- **Given** (Dado que) <pré-requisitos>
  - Colocar o sistema em um estado conhecido antes do usuário iniciar a interação com o sistema.
- **When** (Quando) <ação>
  - Descrever a ação chave que o usuário executa.
- **Then** (Então) <resultado esperado>
  - Verificar/validar a saída do sistema.

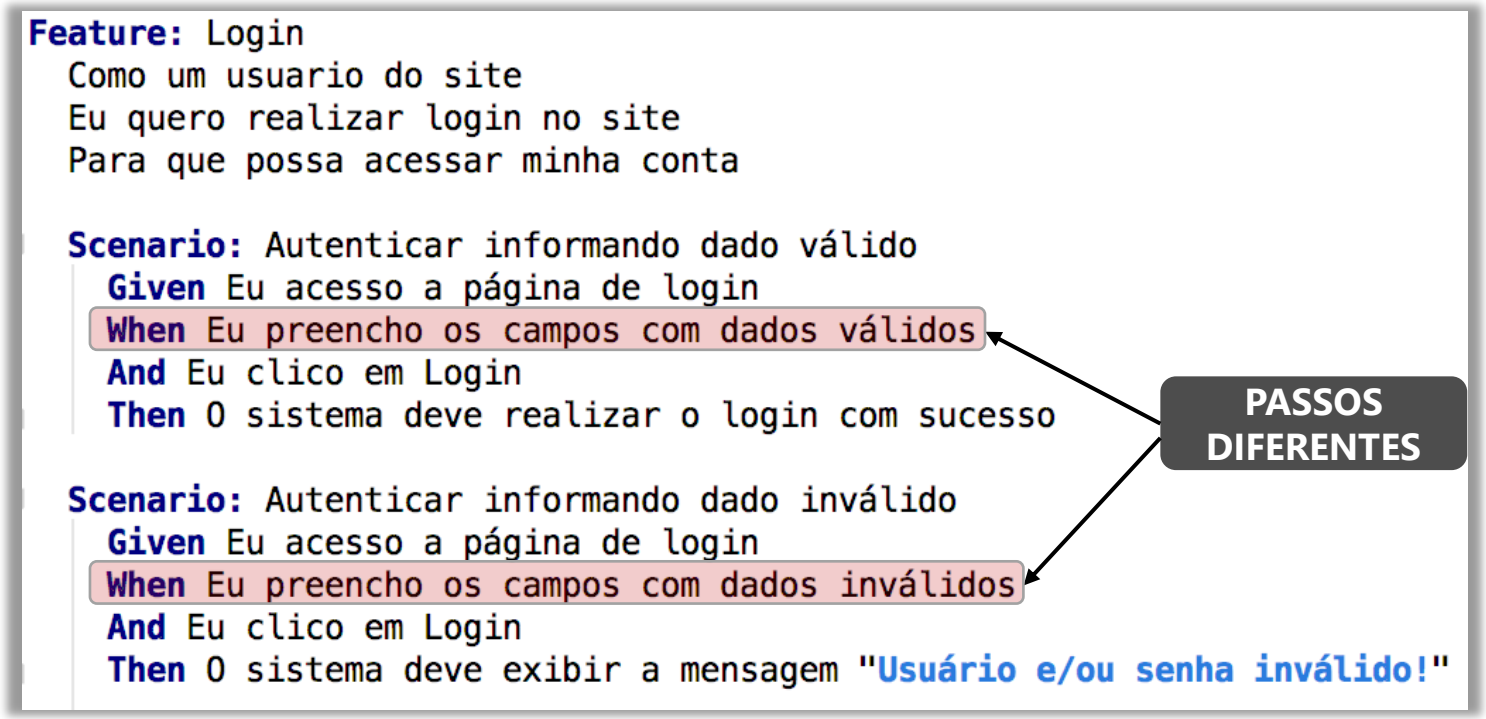
- Se houver vários Given/When/Then pode-se usar And (E) ou But (Mas), permitindo uma leitura mais fluente do cenário.
  - Existe ainda a opção de colocar * (asterisco), palavra-chave desafia todo o propósito de se ter as outras palavras-chaves.

- **Não existe distinção técnica** entre nenhum destes passos, mas como boa prática isso deve ser seguido. Essas palavras-chaves foram selecionadas para o seu propósito.

Parâmetros

- É possível que **passos individuais "reapareçam" em diferentes cenários** ou funcionalidades.

- Podem não ser totalmente idênticas, mas diferem ligeiramente em detalhes, como um valor para preencher em um campo de formulário.



Parâmetros

- A **manutenção de dois passos** (ou mesmo implementações de passos independentes com muitos códigos duplicados) **causam muito trabalho** a longo prazo.
- Portanto


```
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  Scenario: Autenticar informando dado válido
    Given Eu acesso a página de login
    When Eu preencho os campos com fulano e fulano2018
    And Eu clico em Login
    Then O sistema deve realizar o login com sucesso

  Scenario: Autenticar informando dado inválido
    Given Eu acesso a página de login
    When Eu preencho os campos com fulano e 12345
    And Eu clico em Login
    Then O sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

REUTILIZAÇÃO DO PASSO

Expressões Regulares

- A correspondência de palavras específicas é simples, mas muitas vezes você quer flexibilidade para combinar uma variedade de *strings*.

- As expressões regulares são a **chave para a flexibilidade**.
- As expressões regulares bem elaboradas permitem que você **reutilize** as definições de passos, **evitando a duplicação** e mantendo seus testes sustentáveis.

- **Âncoras**
- A presença de âncoras garante que seja tratado como uma Expressão Regular.
  - Use ^ (circunflexo) no início e finalize com $ (cifrão).

Expressões Regulares

Wildcards (Curingas)

.*	**qualquer caractere** (exceto uma nova linha) 0 ou mais vezes
.+	pelo menos um de **qualquer coisa** (mesmo do anterior, **exceto vazio**)
[0-9]* ou **\d***	corresponde a uma **série de dígitos** (ou nada)
[0-9]+ ou **\d+**	corresponde a **um ou mais dígitos** (mesmo do anterior, **exceto vazio**)
"[^\"]*" ou **".*"**	corresponde a alguma coisa (ou nada) **entre aspas** duplas
?	faz do **último caractere ou grupo opcional**, por exemplo uma?
|	**OR lógico**, por exemplo logado|logada
(padrão)	usado para **capturar uma substring** para um parâmetro
(?:padrão)	usado em um grupo que **não é capturado como parâmetro**

Referência: *http://agileforall.com/wp-content/uploads/2011/08/Cucumber-Regular-Expressions-Cheat-Sheet.pdf*

Expressões Regulares


```
Given Eu tenho 10 usuários cadastrados
```

```
@Given("^Eu tenho (\\d+) usuários cadastrados$")
public void euTenhoUsuariosCadastrados(int usuarios) {
}
```

```
Given eu quero comprar um coca-cola de 1.5 litros
```

```
@Given("^eu quero comprar um coca-cola de (\\d+.\\d+) litros$")
public void euQueroComprarUmCocaColaDeLitros(double litros) {
}
```

```
Given Eu estou logado como um administrador
```

```
@Given("^Eu estou logado como um (.*)")
public void euEstouLogadoComoUmPerfil(String perfil) {
}
```

```
Given eu estou logado como um "administrador"
```

```
@Given("^eu estou logado como um \"(.*)\"")
public void euEstouLogadoComoUmPerfil(String perfil) {
}
```

```
Given Eu estou logada como uma usuária
```

```
@Given("^Eu estou logad(?:o|a) como uma? (.*)")
public void euEstouLogadoComoUmAdministrador(String perfil) {
}
```

Scenario Outline

- Copiar e colar cenários para usar diferentes valores pode ser muito tedioso e repetitivo.
- O `Scenario Outline` permite **criar cenários parametrizados** (como um *template*).

```
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  Scenario Outline: Autenticar informando dado inválido
    Given eu acesso a página de login
    When eu preencho os campos com <usuario> e <senha>
    And eu clico em Login
    Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"

    Examples:
      | usuario  | senha |
      | ciclano  | 12345 |
      | beltrano |       |
```

- Neste modelo o `Scenario Outline` será executado uma vez para cada linha na seção de exemplos abaixo dela (exceto para a primeira linha que é o cabeçalho).
- Para utilizar o valor do exemplo em um passo basta utilizar <texto>, onde o

Scenario Outline

- **Múltiplas tabelas** também são permitidas em um `Scenario Outline`.
  - O que permite agrupar diferentes tipos de exemplos, se desejar.
- Quando você tem um grande conjunto de exemplos, dividi-lo em várias tabelas pode facilitar o entendimento de um leitor.

```
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  Scenario Outline: Autenticar informando dado inválido
    Given eu acesso a página de login
    When eu preencho os campos com <usuario> e <senha>
    And eu clico em Login
    Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"

    Examples: Usuários inválidos
      | usuario | senha |
      | 3       | 12345 |
      | ful     | 12345 |

    Examples: Senhas inválidas
      | usuario  | senha |
      | ciclano  | 12345 |
      | beltrano |       |
```

Background

- **`Background`** permite adicionar algum contexto a todos os cenários de uma funcionalidade.
  - É como um `Scenario` sem título, que contém uma série de passos.

- A diferença ocorre quando ele é executado:
  - O `Background` será executado antes de cada `Scenario` (dentro da mesma `Feature`), mas depois do *hook* `Before`.

```
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  Background:
    Given eu acesso a página de login

  Scenario: Autenticar informando dado válido
    When eu preencho os campos com fulano e senhaCorreta
    And eu clico em Login
    Then o sistema deve realizar o login com sucesso

  Scenario Outline: Autenticar informando dado inválido
    When eu preencho os campos com <usuario> e <senha>
    And eu clico em Login
    Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"

    Examples:
      | usuario  | senha |
      | ciclano  | 12345 |
      | beltrano |       |
```

Data Tables

- **`Data Table`** permite passar múltiplos parâmetros (um grande conjunto de dados) em um passo.
- Não confunda `Data Table` com `Scenario Outline`.
  - São muito parecidos, mas eles **tem propósitos diferentes**.
  - `Scenario Outline` → declaram diferentes valores múltiplos ao mesmo cenário.
  - `Data Table` → são usadas para passar um conjunto de dados a um determinado passo.

```
Feature: Pesquisar usuário
  Como um usuario do site
  Eu quero pesquisar por usuários cadastrados
  Para que possa visualizar dados de usuários

  Scenario: Pesquisar por usuário existente
    Given eu tenho os seguintes usuários no sistema
      | Nome     | Usuario  | Email               |
      | Fulano   | fulano   | fulano@gmail.com    |
      | Ciclano  | ciclano  | ciclano@gmail.com   |
      | Beltrano | beltrano | beltrano@gmail.com  |
    When eu realizo a pesquisa por fulano
    Then eu devo visualizar os seguintes dados
      | Nome    | Fulano           |
      | Usuario | fulano           |
      | Email   | fulano@gmail.com |
```

Data Tables


```
Feature: Pesquisar usuário
  Como um usuario do site
  Eu quero pesquisar por usuários cadastrados
  Para que possa visualizar dados de usuários

  Scenario: Pesquisar por usuário existente
    Given eu tenho os seguintes usuários no sistema
      | Nome     | Usuario  | Email              |
      | Fulano   | fulano   | fulano@gmail.com   |
      | Ciclano  | ciclano  | ciclano@gmail.com  |
      | Beltrano | beltrano | beltrano@gmail.com |
    When eu realizo a pesquisa por fulano
    Then eu devo visualizar os seguintes dados
      | Nome    | Fulano           |
      | Usuario | fulano           |
      | Email   | fulano@gmail.com |
```

```
import lombok.Data;

@Data
public class Usuario {
  private String nome;
  private String usuario;
  private String email;
}
```

```
@Given("^eu tenho os seguintes usuários no sistema$")
public void euTenhoOsSeguintesUsuariosNoSistema(DataTable usuarios) {
  List<List<String>> data = usuarios.raw();
  data.forEach(s -> System.out.println(s.get(0) + " - " + s.get(1) + " - " + s.get(2)));
}

@Given("^eu tenho os seguintes usuários no sistema$")
public void euTenhoOsSeguintesUsuariosNoSistema(List<Usuario> usuarios) {
  usuarios.forEach(
      u -> System.out.println(u.getNome() + " - " + u.getUsuario() + " - " + u.getEmail()));
}
```

```
@Then("^eu devo visualizar os seguintes dados$")
public void euDevoVisualizarOsSeguintesDados(DataTable usuario) {
  List<List<String>> data = usuario.transpose().raw();
  data.forEach(s -> System.out.println(s.get(0) + " - " + s.get(1) + " - " + s.get(2)));
}

@Then("^eu devo visualizar os seguintes dados$")
public void euDevoVisualizarOsSeguintesDados(@Transpose List<Usuario> usuario) {
  usuario.forEach(
      u -> System.out.println(u.getNome() + " - " + u.getUsuario() + " - " + u.getEmail()));
}
```

Doc Strings

- `Doc Strings` é útil para passar um texto maior para um passo.
- O texto deve ser delimitado por três marcas de dupla citações em linhas próprias:

```
Feature: Exportar dados de usuários
  Como um usuario do site
  Eu quero exportar dados de usuários
  Para que possa visualizá-los em planilha

  Scenario: Exportar dados de usuários em CSV
    Given eu tenho os seguintes usuários no sistema
      | Nome     | Usuario  | Email               |
      | Fulano   | fulano   | fulano@gmail.com    |
      | Ciclano  | ciclano  | ciclano@gmail.com   |
      | Beltrano | beltrano | beltrano@gmail.com  |
    When eu clico em Exportar
    Then eu devo ver um arquivo conforme:
      """
      Fulano;fulano;fulano@gmail.com
      Ciclano;ciclano;ciclano@gmail.com
      Beltrano;beltrano;beltrano@gmail.com
      """
```

Tags e Comentários

- **`Tags`** são uma ótima forma de organizar funcionalidades e cenários.
- Um `Scenario` ou `Feature` pode ter **quantas tags quiser**, basta separá-las com espaços:
  - @smoke @cadastrousuario
  - @login
- Se uma tag existe em uma `Feature`, ela será atribuída para todos os `Scenarios` existentes no arquivo.

- Ao executar os testes pode-se configurar um filtro para executar apenas os testes com uma determinada tag (ou sem uma determinada tag).
  - @ignore
  - @wip

- Para colocar **comentários** no arquivo basta usar # (hashtag) para que a linha seja ignorada.

Tags e Comentários

Quais cenários serão executados?

```
@login
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  @smoke
  Scenario: Autenticar informando dado válido
    Given eu acesso a página de login
    When eu preencho os campos com fulano e fulano2018
    And eu clico em Login
    Then o sistema deve realizar o login com sucesso

  @wip
  # Precisamos verificar qual é a mensagem de erro correta.
  Scenario: Autenticar informando dado inválido
    Given eu acesso a página de login
    When eu preencho os campos com fulano e 12345
    And eu clico em Login
    #Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@login"}
)
public class CucumberRunner {}
```

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@login", "~@wip"}
)
public class CucumberRunner {}
```

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@smoke"}
)
public class CucumberRunner {}
```

Tags e Comentários

```
@login
Feature: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  @smoke
  Scenario: Autenticar informando dado válido
    Given eu acesso a página de login
    When eu preencho os campos com fulano e fulano2018
    And eu clico em Login
    Then o sistema deve realizar o login com sucesso

  @wip
  # Precisamos verificar qual é a mensagem de erro correta.
  Scenario: Autenticar informando dado inválido
    Given eu acesso a página de login
    When eu preencho os campos com fulano e 12345
    And eu clico em Login
    #Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

TODOS

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@login"}
)
public class CucumberRunner {}
```

SOMENTE O PRIMEIRO

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@login", "~@wip"}
)
public class CucumberRunner {}
```

SOMENTE O PRIMEIRO

```
@RunWith(Cucumber.class)
@CucumberOptions(
  glue = "stepdefinitions",
  features = "src/test/resources",
  tags = {"@smoke"}
)
public class CucumberRunner {}
```

Idiomas

- O Gherkin está **disponível em muitas linguagens**, permitindo você escrever histórias usando as palavras-chave de um idioma específico.
- Qualquer linguagem diferente do inglês (en) **precisa ser explicitada** com um comentário `#language: <idioma>` no início de seu arquivo *.feature.

```
#language: pt
Funcionalidade: Login
  Como um usuario do site
  Eu quero realizar login no site
  Para que possa acessar minha conta

  Cenário: Autenticar informando dado válido
    Dado que eu acesso a página de login
    Quando eu preencho os campos com dados válidos
    E eu clico em Login
    Então o sistema deve realizar o login com sucesso

  Cenário: Autenticar informando dado inválido
    Dado que eu acesso a página de login
    Quando eu preencho os campos com dados inválidos
    E eu clico em Login
    Então o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

#2

BOAS PRÁTICAS

Boas Práticas

- Os cenários não devem conter detalhes técnicos.
  - Foque no "o quê" e evite o "como".

- Faça cenário mais declarativos (enxuto e conciso) e menos imperativos (detalhados).

Se o Gherkin é difícil de entender, é uma documentação ruim.

```
Scenario: Com detalhes técnicos
  Given que uma conta de administrador está configurada no banco de dados
  And eu estou logado como administrador

Scenario: Sem detalhes técnicos
  Given eu estou logado como administrador
```

```
Scenario: Imperativo
  Given que Fulano registra nos serviços bancários online
  And que Fulano abre as seguintes contas:
    | conta  | tipo     | saldo |
    | 123456 | poupanca | 1000  |
    | 123457 | corrente | 100   |
  When Fulano inicia a sessão
  And Fulano vai para a página inicial
  And Fulano vê suas contas
  Then Fulano deveria ver uma lista de suas contas:
    | conta  | tipo     | saldo |
    | 123456 | poupanca | 1000  |
    | 123457 | corrente | 100   |

Scenario: Declarativo
  Given que Fulano está cadastrado nos serviços bancários online
  And que Fulano abriu as seguintes contas:
    | conta  | tipo     | saldo |
    | 123456 | poupanca | 1000  |
    | 123457 | corrente | 100   |
  When Fulano vê o resumo da sua conta
  Then Fulano deveria ver uma lista de suas contas
```

Boas Práticas

- Formate o *.feature corretamente...
  - Indentação
  - Os passos começam com letra minúscula (exceto em caso de substantivos)
  - Não pontuar os passos

- Evitar descrições, títulos e passos longos (sugestão: limite a 80-120 caracteres)

- Normalmente se tem um *.feature por funcionalidade e todos os seus respectivos cenários, mas:
  - Limite o número de Scenarios por Feature. Ninguém quer um arquivo com mil linhas.
  - Limite o número de passos por Scenario (i.e., máximo 10).

Boas Práticas

- Não só organize os cenários em pastas e arquivos *.feature, mas também **utilize tags**.
  - Adote um conjunto padrão de tags e evite duplicar (i.e., duas tags com o mesmo sentido).
    - @smoke e @críticos (talvez só um deles)
  - Escreva tags em minúsculo e use hifens ("-") para separar as palavras.
    - @cadastro-usuario
  - Limite o comprimento das tags (sugestão: 20-30 caracteres).

Boas Práticas

- Don’t Repeat Yourself (DRY)
  - Reutilizar passos ao máximo (faça uso das **expressões regulares**).
  - `Scenario Outline` deve ser sempre usado quando o mesmo cenário precisar ser executado para múltiplos dados.
  - Utilize o `Background` e *hook* `Before` sempre que possível
- Evite dependências entre cenários.
- Tenha uma boa arquitetura de automação (organização, uso de padrões, etc.).

Exemplos Anti-padrão

```
# Exemplo 1
Given que uma conta de administrador está configurada no banco de dados
And eu estou logado como administrador
When eu abrir a página de cadastro de usuário
And informar 'Fulano' no campo Nome
And informar 'fulano@gmail.com' no campo Email
And informar 'senha' no campo Senha
And clicar em Salvar
Then o usuário deve ser registrado com sucesso
```

- Detalhes técnicos;
- Cenário muito imperativo.

```
Given eu estou logado como administrador
When eu cadastrar o usuário:
| Nome  | Fulano              |
| Email | fulano@gmail.com    |
| Senha | senha               |
Then o usuário deve ser registrado com sucesso
```

Exemplos Anti-padrão

```
# Exemplo 2
Given que Fulano quer se registrar nos serviços bancários online
When ele abre a página de registro
And ele informa 'Fulano' no campo Nome
And ele informa 'fulano@gmail.com' no campo Email
And ele informa '01/01/1980' no campo Data de Nascimento
And ele informa 'Rua dos Guajajaras, 870' no campo Endereco
And ele informa 'Belo Horizonte' no campo Cidade
And ele informa '30180-100' no campo CEP
And ele clica em Cadastrar
Then seu pedido de registro deve ser criado como pendente
And ele deve receber um email com o contrato para que seja assinado
```

- Detalhes técnicos;
- Cenário muito imperativo.

```
Given que Fulano quer se registrar nos serviços bancários online com os seguintes dados:
| Nome           | Fulano                 |
| Email          | fulano@gmail.com       |
| DataNascimento | 01/01/1980             |
| Endereco       | Rua dos Guajajaras, 870 |
| Cidade         | Belo Horizonte         |
| Cep            | 30180-100              |
When ele realiza o cadastro
Then seu pedido de registro deve ser criado como pendente
And ele deve receber um email com o contrato para que seja assinado
```

Exemplos Anti-padrão

```
# Exemplo 3
Given eu estou logado no sistema
When eu clico em Pesquisar
Then eu vou ver as opções de pesquisa para pesquisar por um cliente por nome, data de nascimento e conta
```

- Cenário sem objetivo;
- Passo muito longo.

```
Given eu tenho os seguintes clientes no sistema
| Nome     | Usuario  | Email                |
| Fulano   | fulano   | fulano@gmail.com     |
| Ciclano  | ciclano  | ciclano@gmail.com    |
| Beltrano | beltrano | beltrano@gmail.com   |
When eu realizo a pesquisa por fulano
Then eu devo visualizar os seguintes dados
| Nome    | Fulano             |
| Usuario | fulano             |
| Email   | fulano@gmail.com   |
```

Exemplos Anti-padrão

```
# Exemplo 4
Given uma conta poupança que possui R$10000
When o cálculo de rendimento de 0.5% é aplicado
Then o total de rendimento deve ser 50

Given uma conta poupança que possui R$50000
When o cálculo de rendimento de 0.6% é aplicado
Then o total de rendimento deve ser 300
```

- Repetição de cenário.

```
Given uma conta poupança que possui R$<saldo>
When o cálculo de rendimento de <taxaRendimento>% é aplicado
Then o total de rendimento deve ser <rendimento>

Examples:
   | saldo | taxaRendimento | rendimento |
   | 10000 | 0.5            | 50         |
   | 50000 | 0.6            | 300        |
```

Exemplos Anti-padrão

```
# Exemplo 5
Given eu me cadastrei como Fulana e minha senha é senha
When eu verifico minha conta bancária o meu saldo é R$100
```

- Dois passos juntos;
- Não possui `Then` (junto com `When`).

```
Given o seguinte cadastro no sistema:
| Nome  | Fulana |
| Senha | senha  |
| Saldo | 100    |
When acessar a conta bancária
Then o saldo deve ser de R$100
```

Exemplos Anti-padrão

```
# Exemplo 6
Given uma conta bancária
When eu saco dinheiro
Then o saldo deve ser o saldo original menos o valor que saquei
```

- Muito alto nível;
- Pouca possibilidade de reuso (exemplo: depósito).

```
Given uma conta bancária com saldo de R$800
When eu saco R$100
Then o saldo deve ser R$700
```

Exemplos Anti-padrão

```
# Exemplo 7
Given eu clico em Exportar Relatório
When eu clico no icone CSV
Then o sistema deve gerar o arquivo no formato escolhido
```

- Não tem uma pré-condição,
- O `When` está como `Given`.

```
Given eu tenho os seguintes usuários no sistema
| Nome      | Usuario  | Email               |
| Fulano    | fulano   | fulano@gmail.com    |
| Ciclano   | ciclano  | ciclano@gmail.com   |
| Beltrano  | beltrano | beltrano@gmail.com  |
When eu clico em Exportar Relatório
And eu clico no icone CSV
Then o sistema deve gerar o arquivo conforme:
  """
  Fulano;fulano;fulano@gmail.com
  Ciclano;ciclano;ciclano@gmail.com
  Beltrano;beltrano;beltrano@gmail.com
  """
```

Exemplos Anti-padrão

```
# Exemplo 8
Given eu acesso a página de login
When eu preencho os campos com dados inválidos
And eu clico em Login
And o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

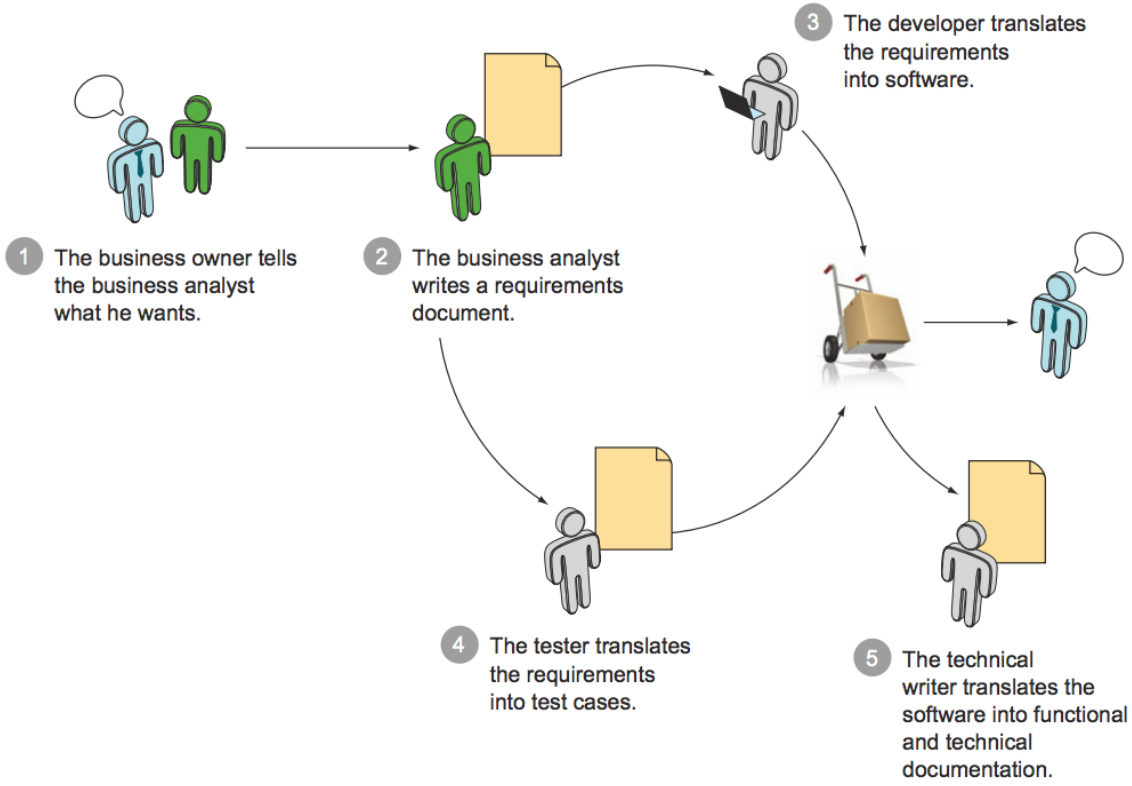
- O `Then` está como `When`.

```
Given eu acesso a página de login
When eu preencho os campos com dados inválidos
And eu clico em Login
Then o sistema deve exibir a mensagem "Usuário e/ou senha inválido!"
```

#3

CICLO DE VIDA DO BDD

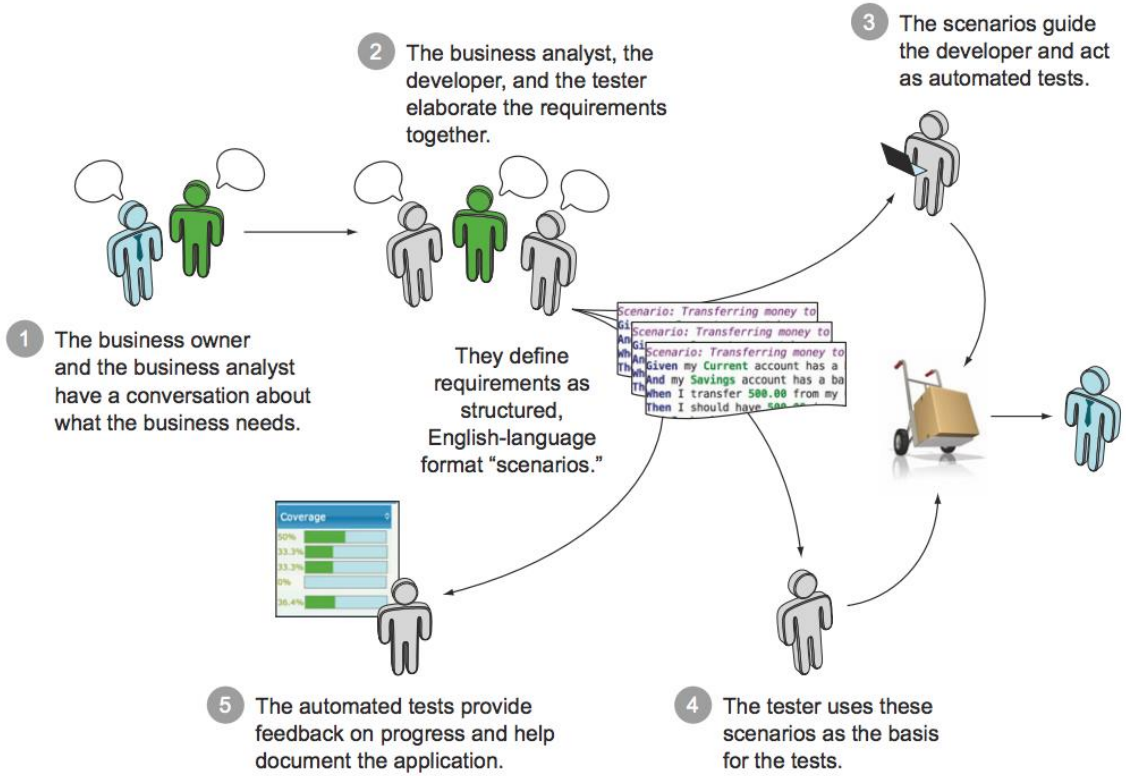
Ciclo de Vida sem BDD



- Considere...
  - uma Especificação de Requisitos de Negócios,
  - além de uma Especificação Funcional,
  - além de uma Especificação Técnica e,
  - ainda, uma Especificação de Testes...

- Um efeito colateral é o do **TELEFONE SEM FIO**, onde a perda da fidelidade dos requisitos se desenvolve naturalmente.

Ciclo de Vida com BDD

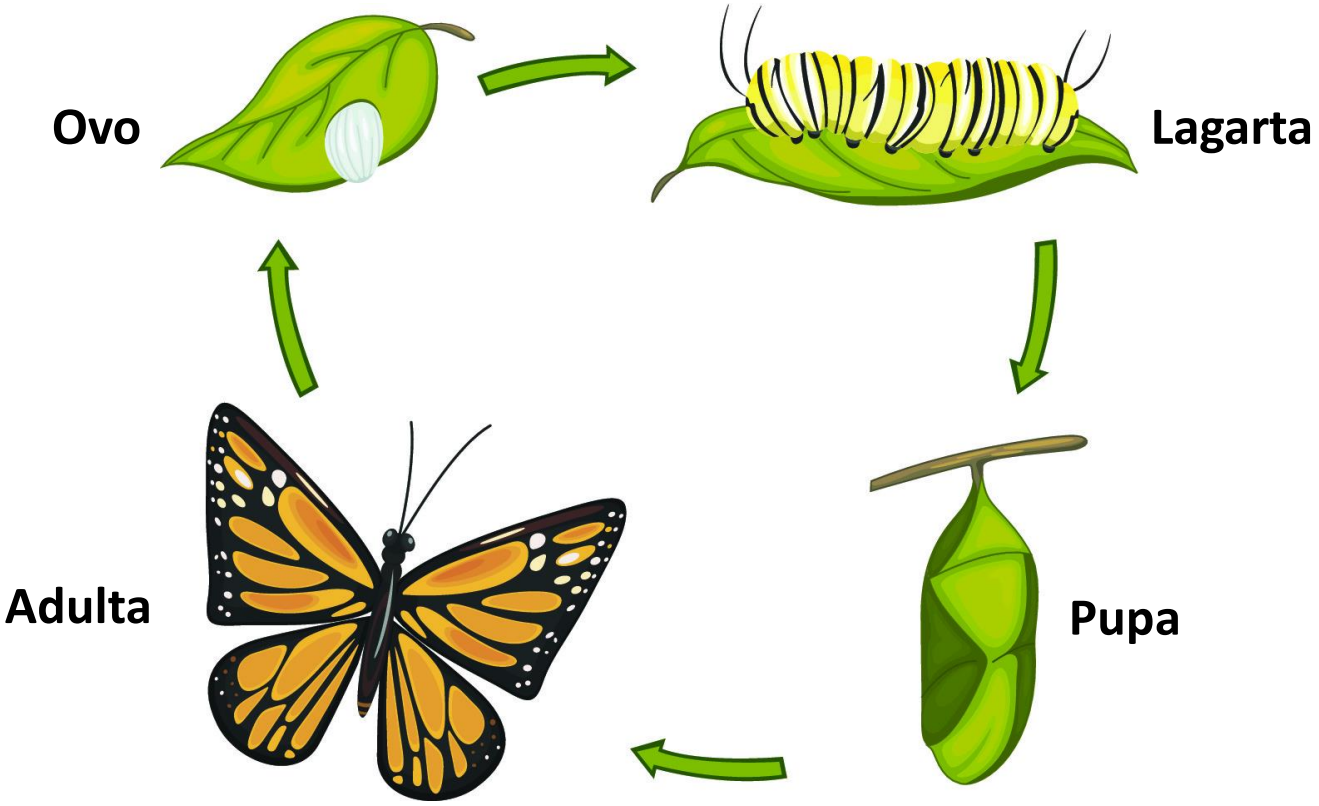


- BDD é um documento compartilhado que vive, e fornece valor, ao longo do ciclo de vida de um projeto de software.
- O BDD oferece uma maneira mais simples de colaborar e manter o conhecimento dos requisitos durante a vida útil de um projeto.

Ciclo de Vida do BDD

- O ciclo de vida do BDD se assemelha ao ciclo de vida de uma borboleta.

Os cenários (criados pelo Dono do Produto) são muito alto nível até o ponto de serem **quase inutilizáveis**.



Desenvolvedores **ajustarão os cenários** para serem mais integrados com a ferramenta escolhida.

Neste momento devemos ter um **conjunto completo** de cenários.

Testadores podem **adicionar testes** de limite e demais técnicas de testes.

Ciclo de Vida do BDD

1

```
Feature: Criar um pedido de sapatos novos
  Como encontrei o par perfeito de sapatos e são alguns dias até a festa
  Eu quero pagar por eles e enviá-los com prioridade
  Para que eu possa levá-los a tempo e não tenho que usar meu velho par.

  Scenario: Pagar por sapatos
    Given eu encontrei meu par de sapatos
    And eu informei os detalhes do meu cartão de crédito
    When eu clicar em 'Comprar'
    Then eu deveria ter um recibo de pagamento na minha caixa de entrada
```

DONO DE PRODUTO

2

```
Feature: Criar um pedido de sapatos novos
  Como encontrei o par perfeito de sapatos e são alguns dias até a festa
  Eu quero pagar por eles e enviá-los com prioridade
  Para que eu possa levá-los a tempo e não tenho que usar meu velho par.

  Background:
    Given eu estou logado
    And os detalhes do meu cartão de crédito estão salvos na minha conta

  Scenario: Pagar por sapatos
    Given eu adicionei "Nike MAGs" à minha cesta de compras
    And eu selecionei SEDEX
    When eu clicar em 'Comprar'
    Then eu deveria ver uma mensagem de transação bem-sucedida com um número de recibo
```

PRIMEIRA IMPLEMENTAÇÃO

Ciclo de Vida do BDD

3

```
Feature: Criar um pedido de sapatos novos
  Como encontrei o par perfeito de sapatos e são alguns dias até a festa
  Eu quero pagar por eles e enviá-los com prioridade
  Para que eu possa levá-los a tempo e não tenho que usar meu velho par.

  Background:
    Given eu estou logado
    And os detalhes do meu cartão de crédito estão salvos na minha conta

  Scenario: Pagar por sapatos
    Given eu adicionei "Nike MAGs" à minha cesta de compras
    And eu selecionei SEDEX
    When eu clicar em 'Comprar'
    Then eu deveria ver uma mensagem de transação bem-sucedida com um número de recibo

  Scenario: Item fora de estoque
    Given eu adicionei "Glass Slippers" à minha cesta de compras
    And eu selecionei SEDEX
    When eu clicar em 'Comprar'
    Then eu deveria ver uma mensagem fora de estoque
```

ADICIONANDO CENÁRIOS

Ciclo de Vida do BDD

4

```
Feature: Criar um pedido de sapatos novos
  Como encontrei o par perfeito de sapatos e são alguns dias até a festa
  Eu quero pagar por eles e enviá-los com prioridade
  Para que eu possa levá-los a tempo e não tenho que usar meu velho par.

  Background:
    Given eu estou logado
    And os detalhes do meu cartão de crédito estão salvos na minha conta
    And minha preferência de envio padrão é 'SEDEX'

  Scenario: Pagar por sapatos
    Given eu adicionei "Nike MAGs" à minha cesta de compras
    When eu clicar em 'Comprar'
    Then eu deveria ver uma mensagem de transação bem-sucedida com um número de recibo

  Scenario: Item fora de estoque
    Given eu adicionei "Glass Slippers" à minha cesta de compras
    When eu clicar em 'Comprar'
    Then eu deveria ver uma mensagem fora de estoque
```

REFATORAÇÃO

OBRIGADO!

Referências

Apresentação adaptada de treinamento interno das empresas Base2 e Framework

- Wynne, M., Hellesoy, A., Tooke S. The Cucumber Book. O'Reilly; Edição: 2nd ed. 2017
- Smart, J. BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle. 2014
- Cucumber Reference: https://cucumber.io/docs/reference
- BDD 101: Unit, Integration, and End-to-end Tests: https://automationpanda.com/2017/10/14/bdd-101-unit-integration-and-end-to-end-tests/
- The curious lifecycle of behaviour driven development scenarios: http://www.zero-one.io/blog/2015/04/01/the-curious-lifecycle-of-behaviour-driven-development-scenarios/