BEHAVIOR DRIVEN DEVELOPMENT (BDD)

Manual de Boas Práticas

Sumário

Histórico de Revisão

Data	Autor	Comentário
15/06/2018	José Paulo Almeida	Versão inicial

Introdução

O Behavior Driven Development (BDD), desenvolvimento orientado a comportamento é uma técnica de desenvolvimento de sistemas que incorpora algumas da melhores práticas de várias metodologias, com o intuito de melhorar a colaboração entre desenvolvedores, QAs e pessoas das áreas de negócio de um projeto de software.

O BDD traz uma linguagem simples, que deve ser facilmente compreendida por qualquer pessoa envolvida com o projeto, servindo como ferramenta para escrever histórias, cenários de teste e detalhar comportamentos esperados.

Para escrever os cenários usamos Gherkin, uma linguagem orientada a espaços (como Python e YAML), que usa indentação para a definição da estrutura. As linhas iniciam com uma palavra-chave e os finais das linhas fecham as declarações. Abaixo vamos analisar os principais pontos do Gherkin.

Dado, Quando, Então

A estrutura mais básica do Gherkin é formada por etapas e estas iniciam-se por três palavras-chaves: **Dado**, **Quando** e **Então**.

A etapa **Dado** é onde são definidas as pré-condições de um cenário, ou seja, todos os passos necessários para colocar o sistema em um estado conhecido, antes do usuário iniciar sua interação.

O **Quando** é a etapa usada para descrever a ação-chave executada pelo usuário. Pode ser uma ação apenas, ou um conjunto de ações que culminam em um ponto onde os resultados serão visíveis.

Estes resultados são verificados na etapa **Então**, onde observamos as saídas do cenário. Devemos validar neste ponto os comportamentos apresentados pelo sistema após as ações realizadas na etapa Quando.

Geralmente descrevemos na etapa Quando as ações ativas do usuário (exemplo: clico, preencho, seleciono etc.), enquanto na etapa Então são descritas as ações passivas (exemplo: vejo, recebo, visualizo etc.).

```
Dado que eu esteja na tela de Clientes
Quando clico no botão "Novo"
Então vejo o formulário de cadastro de clientes.
```

E, Mas

Quando o cenário é mais longo e tem várias etapas **Dado**, **Quando** ou **Então**, podemos usar as palavras-chave **E** e **Mas** como substitutas complementares, facilitando a leitura. Ambas tem exatamente a mesma função e a escolha do uso de uma ou outra é do redator do cenário, procurando sempre a melhor compreensão do comportamento descrito.

É importante salientar que repetir as palavras-chave **Dado**, **Quando** ou **Então** não é errado, porém substituí-las por **E** e **Mas** torna o cenário mais legível e compreensível. Abaixo um exemplo em que há a repetição de **Dado**, **Quando** e **Então** e em seguida o mesmo caso substituindo as palavras duplicadas por **E** e **Mas**.

```
Dado que eu esteja na tela de Login
Dado que eu possua acesso ao sistema
Quando preencho o campo "usuário" corretamente
Quando preencho uma senha inválida
Quando clico no botão "Entrar"
Então vejo a mensagem de senha inválida
Então vejo o campo "senha" marcado em vermelho.
```

```
Dado que eu esteja na tela de Login
E que eu possua acesso ao sistema
Quando preencho o campo "usuário" corretamente
Mas preencho uma senha inválida
E clico no botão "Entrar"
Então vejo a mensagem de senha inválida
E vejo o campo "senha" marcado em vermelho.
```

Cenário

A estrutura Cenário é um conjunto completo de etapas (Dado, Quando, Então) que define um dos comportamentos da funcionalidade. Todo cenário deve iniciar com a palavra-chave "**Cenário:"**, seguida do título descritivo do cenário.

As etapas que compõem o cenário devem estar recuadas (indentadas), pois isso define a estrutura do algoritmo.

```
Cenário: Acessar o Sistema
Dado que eu esteja na tela de Login
E que eu possua acesso ao sistema
Quando preencho o campo "usuário" corretamente
Mas preencho uma senha inválida
E clico no botão "Entrar"
Então vejo a mensagem de senha inválida
E vejo o campo "senha" marcado em vermelho.
```

Funcionalidade

A estrutura **Funcionalidade** serve para reunir os cenários de uma *feature*. Ela é iniciada com a palavra-chave "**Funcionalidade:"**, seguido de um texto descritivo. Recomenda-se utilizar o código da história no Jira e seu título, para auxiliar na rastreabilidade.

Abaixo desta, indentado, pode-se incluir uma descrição breve da história, para facilitar a compreensão dos cenários que serão descritos. Preferencialmente deve ser escrito no formato tradicional de *user stories*: **Eu como** <papel>, **desejo** <objetivo>, **para** <benefício>.

```
1   Funcionalidade: PRODSAS-001 - Desenvolvimento da tela de Login
2
3   Eu como gestor
4   Desejo uma tela de Login
5   Para controlar as permissões de acesso no sistema.
6
7       Cenário: Acessar o Sistema
8       Dado que eu esteja na tela de Login
9       E que eu possua acesso ao sistema
10      Quando preencho o campo "usuário" corretamente
11      Mas preencho uma senha inválida
12      E clico no botão "Entrar"
13      Então vejo a mensagem de senha inválida
14      E vejo o campo "senha" marcado em vermelho.
```

Delineação do Cenário

Quando há vários cenários em que os passo são muito parecidos, com apenas alguns detalhes diferentes ou quando um mesmo teste deve ser executado várias vezes usando diferentes massas, usamos a estrutura de **Delineação do Cenário**.

Ela é muito semelhante ao **Cenário** comum; a principal diferença é que nas etapas são incluídas variáveis e no final é incluída uma tabela com os valores que devem ser utilizados.

Uma **Delineação do Cenário** é executada uma vez para cada linha da tabela da seção de exemplo, exceto pela primeira linha que é o cabeçalho.

Abaixo um exemplo de uma mesma funcionalidade escrita com três cenários e logo após com uma delineação do cenário.

```
Funcionalidade: PRODSAS-001 - Desenvolvimento da tela de Login

Eu como gestor
Desejo uma tela de Login
Para controlar as permissões de acesso no sistema.

 Cenário: Acessar o sistema
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" corretamente
    E preencho o campo "senha" corretamente
    E clico no botão "Entrar"
    Então vejo a tela inicial do sistema.

 Cenário: Usuário Inválido
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" com um valor inválido
    E preencho o campo "senha"
    E clico no botão "Entrar"
    Então vejo a mensagem "Usuário desconhecido".

  Cenário: Senha Incorreta
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" corretamente
    E preencho o campo "senha" com a senha incorreta
    E clico no botão "Entrar"
    Então vejo a mensagem "Senha incorreta".
```

```
Funcionalidade: PRODSAS-001 - Desenvolvimento da tela de Login

Eu como gestor
Desejo uma tela de Login
Para controlar as permissões de acesso no sistema.

 Delineacao do Cenario: Acessar o sistema
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" com <user>
    E preencho o campo "senha" com <password>
    E clico no botão "Entrar"
    Então vejo <resultado>

    Exemplos:
    | user       | password   | resultado                          |
    | sysadmin   | 1234567    | a tela inicial do sistema          |
    | xxx        | 123456     | a mensagem "Usuário desconhecido"  |
    | sysadmin   | 1          | a mensagem "Senha incorreta"       |
```

Um outro exemplo abaixo, em que um mesmo teste deve ser executado com diferentes usuários, obtendo diferentes resultados.

```
Funcionalidade: PRODSAS-002 - Controle de Acesso

Eu como gestor
Desejo uma tela de Login
Para controlar as permissões de acesso no sistema.

 Delineacao do Cenario: Acessar o sistema
    Dado que eu esteja na tela de Login
    Quando logo com um usuário do perfil <user>
    Então vejo <resultado>.

    Exemplos:
    | perfil          |  resultado                                |
    | financeiro      | a tela inicial com o menu Finanças        |
    | operacional     | a tela inicial com o menu Operações       |
    | atendimento     | a tela inicial com o menu Atendimento     |
    | sysadmin        | a tela inicial com todos os menus         |
```

Contexto ou Cenário de Fundo

Contexto ou **Cenário de Fundo** é um conjunto de etapas que são comuns para dois ou mais cenários e que são agrupados para aumentar a agilidade da redação. No Contexto podem ter etapas de **Dado** ou de **Dado** e **Quando**, desde que todo o bloco de etapas seja comum a todos os cenários que o compõem.

Ou seja, se todos os cenários compartilham um ou mais condições de **Dado**, estas podem ser incluídas no **Contexto**. Caso um **Cenário** tenha mais condições de **Dado**, esta é incluída diretamente no cenário, iniciando com a palavra-chave **E** ou **Mas**, sem **Dado**.

Se todas as condições de **Dado** de todos os cenários forem exatamente as mesmas, é possível incluir no **Contexto** também uma ou mais etapas de **Quando**. Isso vai acontecer quando todas as pré-condições e algumas ações são comuns a todos. Neste caso os **Cenário** vai ser iniciado com a palavra-chave **E** ou **Mas**, sem **Dado** nem **Quando**.

A estrutura **Contexto** é iniciada com a palavra-chave "**Contexto:"** ou "**Cenário de Fundo:"**, seguido de um texto descritivo que resuma o ponto em comum entre todos os cenários que a compõem. Quando dentro de uma **Funcionalidade** existe apenas um **Contexto**, o texto descritivo é facultativo.

Como o Gherkin é uma linguagem indentada, todos os cenários que compõem um **Contexto** devem estar tabulados à direita. Os cenários que não estiverem indentados não são considerados como parte desta estrutura, porém é recomendado criar um novo **Contexto**, mesmo que sem etapas, para organizar o código.

Abaixo dois exemplos:

```
Funcionalidade: PRODSAS-003 - CRUD de Clientes

Eu como gestor
Desejo uma clientes
Para cadastrar, buscar, editar e excluir clientes.

 Contexto: Clientes
    Dado que eu esteja logado como sysadmin
    E que esteja na tela de Clientes

    Cenário: Filtrar
    Quando preencho os filtros
    E clico no botão "Buscar"
    Então vejo os resultados da busca na lista.

    Cenário: Cadastrar
    E que eu tenha permissão de cadastro
    Quando clico no botão "Novo"
    E preencho todos os campos
    E clico no botão "Gravar"
    Então vejo a mensagem "Cadastro com sucesso"
    E vejo o novo cliente na lista.
```

```
Funcionalidade: PRODSAS-004 - Relatório Mensal

Eu como gestor
Desejo uma relatório mensal
Para verificar os resultados mês a mês.

 Contexto: Imprimir Relatório Consolidado e Detalhado
    Dado que eu esteja logado como sysadmin
    E que esteja na tela de Relatório Mensal
    Quando eu preencho o campo "Mês"

    Cenário: Consolidado
    E seleciono a opção "Consolidado"
    E clico no botão "Imprimir"
    Então eu vejo o relatório consolidado referente ao mês selecionado.

    Cenário: Detalhado
    E seleciono a opção "Detalhado"
    E clico no botão "Imprimir"
    Então eu vejo o relatório detalhado referente ao mês selecionado.
```

Tabela de Dados

A estrutura de **Tabela de Dados**, também chamada de **Data Table** é utilizada para agrupar um conjunto de dados que será utilizado na validação do cenário. É uma ferramenta muito versátil, porém não muito intuitiva e portanto é necessário atenção para a utilizar.

Também é muito comum confundir a **Tabela de Dados** com **Delineação do Cenário**, no entanto são estruturas com funções completamente diferentes. Enquanto a **Delineação do Cenário** declara diferentes valores múltiplos para o cenário (e portanto para cada linha uma

nova execução é necessária), a **Tabela de Dados** é usada para contemplar um único conjunto de dados (que serão consumidos em uma única validação).

Abaixo alguns diferentes exemplos de como utilizar esta estrutura.

```
Cenário: Emitir Relatório de Total de Vendas
Dado que eu tenha feito vendas nos seguintes valores
    | R$ 100,00  |
    | R$ 250,00  |
    | R$ 500,00  |
Quando eu imprimo o Relatório de Total de Vendas
Eu vejo o total de R$ 850,00.
```

```
Cenário: Emitir Relatório de Clientes
Dado que os seguintes clientes existem
    | nome       | email           | cidade          |
    | José       | jose@bdd.com    | Florianópolis   |
    | Dora       | dora@bdd.com    | Curitiba        |
    | Paola      | paola@bdd.com   | São Paulo       |
Quando emito o relatório de clientes
Então vejo o nome, e-mail e cidade dos clientes.
```

```
Cenário: Acessar o sistema
Dado que eu esteja na tela de Login
Quando preencho os campos
    | usuário    | sysadmin        |
    | senha      | 123456          |
    | sistema    | Sascar          |
E clico no botão "Entrar"
Então vejo a tela inicial do sistema
E vejo o nome do usuário logado no canto superior direito.
```

Strings Multilinhas

O Gherkin é uma linguagem orientada a espaços e o final das linhas fecham as declarações. Quando é necessário declarar strings com mais de uma linha, usa-se uma estrutura chamada String Multilinha ou Pystring, que pode ser utilizada dentro de qualquer etapa (Dado, Quando ou Então). A sintaxe para isso é iniciar e encerrar a string com três aspas duplas (""") e preferencialmente indentá-la com dois espaços.

```
Funcionalidade: PRODSAS-006 - Gerar Boleto

    Cenário: Validar Instrução do Boleto
    Dado um título financeiro a vencer
    Quando gero o boleto
    Então vejo na tela o boleto na tela com a instrução:
      """
      Pagável em qualquer banco até o Vencimento.
      Após o vencimento cobrar juros de 2% e mora de 1% ao mês.

      Para emitir segunda via, acesse o Portal Sascar.
      """
    E recebo o boleto por e-mail.
```

Tags

Para organizar e classificar as **Funcionalidades**, **Cenários** e **Delineações do Cenário**, utiliza-se uma estrutura chamada **Tag**. Sua sintaxe é utilizando o caractere arroba (@) seguido de uma palavra-chave, sendo possível incluir diversas tags, bastando separá-las com um espaço.

Quando a **Tag** é incluída na linha acima de um Cenário ou Delineação do Cenário, é válido apenas para este. Já se for incluída na linha acima da Funcionalidade, vale para todos os Cenários e Delineações de Cenário que a compõem.

```
@Login @Regressao
Funcionalidade: PRODSAS-001 - Desenvolvimento da tela de Login

Eu como gestor
Desejo uma tela de Login
Para controlar as permissões de acesso no sistema.

 @High
 Cenário: Acessar o sistema
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" corretamente
    E preencho o campo "senha" corretamente
    E clico no botão "Entrar"
    Então vejo a tela inicial do sistema.

 @Medium
 Cenário: Usuário Inválido
    Dado que eu esteja na tela de Login
    Quando preencho o campo "usuário" com um valor inválido
    E preencho o campo "senha"
    E clico no botão "Entrar"
    Então vejo a mensagem "Usuário desconhecido".
```

Comentários

Na linguagem Gherkin utiliza-se o símbolo cerquilha (#) para indicar comentários. Quando este for incluído, todo o conteúdo da linha à sua direita é considerado um comentário.

Language

O idioma padrão do Gherkin é o inglês, mas existem versões desta linguagem ubíqua para mais de 70 idiomas e dialetos, incluindo opções comuns como o português, italiano, francês, alemão e também opções curiosas como klingon, emoji, inglês arcaico e inglês de piratas, por exemplo.

Para selecionar um idioma diferente do padrão, é necessário indicar isso na primeira linha do arquivo. Para escolher português, por exemplo, utilize a sintaxe “#language: pt”.

```gherkin
#Language: pt

Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado que eu esteja na tela de cadastro de usuários
    Quando preencho o campo corretamente
    E clico em "Salvar"
    Então vejo a mensagem "Usuário cadastrado com sucesso".
```

Boa Prática: Cenário Declarativo

O Gherkin é uma ferramenta para documentar comportamentos e deve ter o foco sempre em facilitar o entendimento dos processos por todos os envolvidos. Um ponto importante para isso é considerar a relevância das informações que comporão os cenários, para evitar passos desnecessários. Chamamos de **cenários imperativos** aqueles em que tudo é declarado nos mínimos detalhes.

```gherkin
Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado que eu esteja na tela de cadastro de usuários
    Quando preencho o campo "Nome"
    E preencho o campo "Login"
    E preencho o campo "Senha"
    E preencho o campo "E-mail"
    E preencho o campo "Telefone"
    E preencho o campo "Cidade"
    E preencho o campo "UF"
    E clico em "Salvar"
    Então vejo a mensagem "Usuário cadastrado com sucesso".
```

O problema de cenários como este acima, é que podem perder o foco do que precisa realmente ser validado. Em um ambiente ágil é comum que todos os envolvidos estejam inteirados do que se espera de cada história. Por isso é saudável utilizar **cenários declarativos**, que concentram-se no que precisa ser testado, removendo todos os passos irrelevantes.

```
Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado que eu esteja na tela de cadastro de usuários
    Quando preencho o campo corretamente
    E clico em "Salvar"
    Então vejo a mensagem "Usuário cadastrado com sucesso".
```

No exemplo acima o cenário deveria validar se a mensagem de sucesso estava sendo apresentada corretamente. Por isso não é necessário detalhar cada um dos campos que serão preenchidos; provavelmente haverá um outro teste apenas para isso. Manter o foco no que precisa ser testado em cada cenário ajuda a manter a legibilidade e o entendimento do comportamento esperado.

Boa Prática: Primeira ou Terceira Pessoa

Sendo o Gherkin uma linguagem ubíqua, é importante que ela seja compreendida por todos os envolvidos no projeto de software. Para isso, duas formas de narrativa são possíveis e corretas: narrativa em primeira pessoa ou em terceira pessoa.

Na narrativa em **terceira pessoa** o redator do código BDD refere-se ao usuário que executa os testes como uma alguém diferente de si. Os partidários desta forma defendem que permite uma perspectiva mais genérica e mais livre.

```
#Language: pt

Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado um usuário na tela de cadastro de usuários
    Quando ele preenche o campo corretamente
    E clica em "Salvar"
    Então o sistema apresenta a mensagem "Usuário cadastrado com sucesso".
```

Contudo Dan North, o criador do BDD, defende que os códigos devem ser escritos em **primeira pessoa**, pois sendo o BDD focado em comportamento, esta prática facilita que o redator se coloque no lugar do usuário e possa enxergar melhor o cenário que está sendo descrito.

```gherkin
#language: pt

Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado que eu esteja na tela de cadastro de usuários
    Quando preencho o campo corretamente
    E clico em "Salvar"
    Então vejo a mensagem "Usuário cadastrado com sucesso".
```

Ambas as formas estão corretas; o que **deve ser evitado** é uma narrativa mista, com partes em terceira pessoa e partes em primeira, pois isso dificulta a legibilidade e causa confusão na compreensão dos cenários.

```gherkin
#language: pt

Funcionalidade: PRODSAS-005 - Cadastro de Usuário

Eu como gestor
Desejo uma tela de cadastro de usuários
Para cadastrar, buscar, editar e excluir usuários.

 Cenário: Mensagem de Sucesso
    Dado que eu esteja na tela de cadastro de usuários
    Quando ele preenche o campo corretamente
    E clica em "Salvar"
    Então vejo a mensagem "Usuário cadastrado com sucesso".
```

No exemplo acima, fica claro como a narrativa mista pode ser prejudicial à compreensão de um código BDD.