THE BDD BOOKS

Formulação

Documentando exemplos com Dado/Quando/Então

> Leitura obrigatória para qualquer pessoa que esteja embarcando em sua jornada BDD
>
> — Daniel Terhorst-North

Seb Rose

e Gáspár Nagy

Apresentação por Angie Jones
e Daniel Terhorst-North

Tradução de Tula Valle Machado
e Vinícius Rodrigues Nunes

The BDD Books - Formulação

Documentando exemplos com cenários Gherkin

Seb Rose, Gáspár Nagy, Tula Valle Machado, e Vinícius Rodrigues Nunes

Esse livro está à venda em http://leanpub.com/bddbooks-formulation-pt

Essa versão foi publicada em 2024-12-03

Tweet Sobre Esse Livro!

Por favor ajude Seb Rose, Gáspár Nagy, Tula Valle Machado, e Vinícius Rodrigues Nunes a divulgar esse livro no Twitter!

A hashtag sugerida para esse livro é #bddbooks.

Descubra o que as outras pessoas estão falando sobre esse livro clicando nesse link para buscar a hashtag no Twitter:

#bddbooks

Conteúdo

Prefácio por Angie Jones

Sou uma grande defensora da frase “mostre, não conte”, especialmente no que diz respeito ao desenvolvimento de software. Muitos especialistas estão constantemente nos dizendo o que deveríamos estar fazendo, mas oferecem poucos exemplos que realmente nos mostram como fazer isso.

Com o Gherkin não é diferente. Uma simples pesquisa na internet resulta em artigo após artigo explicando como as equipes afirmam praticar o desenvolvimento guiado por comportamento (BDD), mas falham no objetivo principal desse processo, que é comunicar-se cedo no processo de desenvolvimento para garantir que a equipe esteja construindo a coisa certa. Poucos desses artigos abordam como fazer isso de forma eficaz.

Fiquei encantada ao saber que Seb e Gáspár aceitaram o desafio de abordar essa lacuna, proporcionando um livro inteiro explicando como escrever cenários em Gherkin de maneira eficaz, o que ajudará nos seus esforços gerais de BDD.

Eles fazem um trabalho maravilhoso ao compartilhar histórias para dar contexto ao seu raciocínio. Eles tiram proveito de sua vasta experiência para cobrir minuciosamente a rica linguagem Gherkin. Eles não poupam esforços ao explicar o como e o porquê.

Este livro é para equipes comprometidas em colaborar de maneira eficaz para construir software de qualidade. Se aplicadas, essas lições abrangentes certamente colocarão qualquer equipe de software no caminho certo, não importa qual tenha sido sua experiência anterior com BDD.

Prepare-se! Você está prestes a embarcar em uma jornada emocionante!

Angie Jones

Prefácio por Daniel Terhorst-North

Em 2003, quando tentei pela primeira vez enquadrar o desenvolvimento guiado por testes sem usar a palavra "teste"", eu não fazia ideia de onde isso levaria. Ao longo do caminho, tive a sorte de conhecer pessoas brilhantes como Seb Rose e Gáspár Nagy que, ao longo de quase duas décadas, foram fundamentais na evolução do BDD em um panorama rico e detalhado de colaboração e criação de valor.

Pode-se argumentar que a formulação é a disciplina menos compreendida do BDD. Muito tem sido escrito sobre colaboração e entendimento compartilhado, e a automação é onde a maioria das pessoas começa com o BDD. Para mim, a formulação é onde reside o verdadeiro poder e alavancagem do BDD.

É necessário habilidade e experiência para capturar "o contexto, todo o contexto e nada além do contexto"" de um cenário para descrever o comportamento desejado de um produto digital. Gáspár e Seb fizeram um trabalho magistral ao fornecer um tratamento abrangente e acessível de um tópico complexo, com muitos exemplos e ilustrações para guiar o caminho.

Me peguei concordando com cada capítulo e cada história. As notas de rodapé por si só são ouro puro. Como exemplo, não sei se foi deliberado, mas a seção deles sobre o acrônimo *BRIEF* é uma mini-aula autorreferencial. Ela usa a *Linguagem de Negócios* do BDD, trabalha através de um exemplo *Real*, é *Reveladora de Intenções*, contém apenas informações *Essenciais* e é *Focada* em uma coisa.

Reconheço muito da minha própria experiência nestas páginas, mas além disso, uma articulação de coisas que são "óbvias quando você as diz em voz alta"". Este livro deve ser leitura obrigatória para qualquer um que esteja começando sua jornada com BDD e servirá como um refresco oportuno e uma referência indispensável para aqueles que estão mais adiantados.

Daniel Terhorst-North

Prefácio

Desenvolvimento Guiado por Comportamento (BDD) é uma abordagem ágil para entregar software, compreendendo três *práticas centrais do BDD*: descoberta, formulação e automação.

- Descoberta – cria uma compreensão compartilhada dos requisitos através da colaboração, geralmente alcançada por meio de uma conversa estruturada centrada em regras e exemplos.
- Formulação – exemplos do comportamento do sistema são documentados usando terminologia de negócios.
- Automação – a documentação é automatizada, criando uma documentação viva que verifica o comportamento do sistema.

A **série de livros BDD Books** guia você na adoção completa do BDD, incluindo práticas específicas necessárias para conduzir o desenvolvimento usando especificações escritas colaborativamente e documentação viva. Este é o segundo livro da série de livros de BDD e apresenta uma imersão profunda na prática de formulação do BDD – a escrita de especificações executáveis em um formato legível por pessoas de negócios.

O primeiro livro cobriu a prática de descoberta do BDD e recomendamos fortemente que você leia primeiro o *Descoberta – Explorar comportamento usando exemplos (Livro 1)* [Nagy2018]. Uma vez que você tenha praticado a formulação usando os princípios que delineamos neste livro, você pode ler *Automação com SpecFlow (Livro 3)* [NagyInPrep].

Muitas equipes de desenvolvimento adotam o BDD com o desejo de melhorar sua automação de testes. Melhorar a automação de testes é um dos resultados significativos de seguir a abordagem BDD, mas é um resultado *secundário*. A menos que você adote as práticas na ordem descrita (*descoberta*, *formulação*, *automação*), você não obterá os benefícios esperados.

Por outro lado, você alcançará melhorias significativas em suas atividades de desenvolvimento de software apenas praticando *descoberta* por si só. Adicione *formulação* e você obterá benefícios adicionais ao desenvolver uma linguagem verdadeiramente ubíqua através de um processo ativo de revisão e feedback. *Automação* então transforma os cenários em uma especificação executável que guia o desenvolvimento e fornece uma rede de segurança durante a manutenção, além de documentação viva legível por pessoas de negócios que é garantida estar atualizada.

Equipes que seguem o conselho que apresentamos têm a melhor chance de criar uma documentação viva valiosa que guiará o desenvolvimento, envolverá stakeholders de negócios e reduzirá o custo de manutenção e melhorias.

O projeto WIMP

A série de livros de BDD segue uma equipe imaginária desenvolvendo o *Onde Está Minha Pizza* (WIMP) O WIMP é um aplicativo de gerenciamento de entrega de pizzas para uma grande empresa de pizzas. O aplicativo permitirá que os clientes rastreiem a localização em tempo real de seus pedidos. A equipe WIMP é composta pelos seguintes membros, sendo a primeira letra de seus nomes indicativa de seu papel na equipe:

- **P**atricia – dono do produto (PO, no inglês)
- **D**aniel – desenvolvedor
- **D**aiana – desenvolvedor
- **E**va – estagiário
- **T**ula – testador (QA, no inglês)
- **U**lisses – experiência do usuário (UX, no inglês)

Em *Formulação*, seguimos a equipe WIMP enquanto eles praticam suas habilidades de formulação usando Gherkin, o formato de especificação compreendido pelo Cucumber, Reqnroll e SpecFlow.

Para quem é este livro

Este livro é escrito para todos envolvidos na especificação e entrega de software (incluindo donos de produtos, analistas de negócios, desenvolvedores e testadores).

Você não precisa de experiência prévia com BDD. O livro descreve como todos os stakeholders precisam estar envolvidos na criação da especificação de um produto. Como você se envolve dependerá de suas habilidades, seus outros compromissos e uma série de outros fatores, mas o envolvimento de todos é essencial. Se você cria as palavras, digita ou fornece feedback construtivo, encontrará este livro indispensável.

Vale ressaltar que enquanto *Discovery* [Nagy2018] é completamente agnóstico em relação a ferramentas, este livro está focado em ferramentas que entendem a sintaxe Gherkin. Isso engloba uma grande quantidade de ferramentas, incluindo Cucumber[1], SpecFlow[2], Reqnroll[3], JBehave[4], Behave[5], e Behat[6].

Por que você deve ler este livro

Sua equipe pode entender o que precisa ser entregue ao sair de uma oficina de requisitos, mas e as pessoas que não estavam na reunião, ou os próprios participantes no futuro quando forem encarregados de corrigir um defeito um ano depois? Neste livro, mostramos como capturar essa compreensão usando Gherkin. Gherkin permite que você escreva especificações usando sua própria linguagem de negócios que é compreensível por todos na equipe, mas suficientemente estruturada para também ser entendida por ferramentas de automação.

Porque a estrutura do Gherkin é tão simples de escrever, é fácil, mas é mais difícil garantir que seja escrito de uma maneira fácil de entender, fácil de manter e valioso o suficiente para que membros da equipe não técnicos participem ativamente de sua criação. Essa é a arte da formulação que ensinamos neste livro.

Como ler este livro

Este livro contém muitas dicas e truques para ajudá-lo a escrever cenários de BDD melhores, além de apontar algumas práticas a serem evitadas. Para facilitar

[1] https://cucumber.io/
[2] https://specflow.org/
[3] https://reqnroll.net/
[4] https://jbehave.org/
[5] https://github.com/behave/behave
[6] https://docs.behat.org/

o entendimento, agrupamos nossos conselhos de acordo com o tipo de problema que nossa equipe WIMP está tentando resolver.

- Capítulo 1, "O que é formulação??", introduz o conceito de formulação, seu papel entre as práticas comuns de BDD (descoberta, formulação, automação) e define os elementos mais importantes de terminologia.
- Capítulo 2, "Limpando um cenário antigo", mostra como um cenário de BDD "ruim" pode ser corrigido e introduz o acrônimo BRIEF que captura os seis princípios essenciais de bons cenários.
- Capítulo 3, "Nossa primeira funcionalidade", guia você através de um arquivo de funcionalidade completo criado pela equipe WIMP. Através disso, mostramos a conexão entre os cenários de BDD e os requisitos discutidos durante a descoberta. Também aprendemos sobre os elementos fundamentais de um arquivo de funcionalidade.
- Capítulo 4, "Uma nova história de usuário", guia você pelas discussões e decisões importantes da equipe WIMP enquanto formulam um novo cenário. Mostramos como os acordos resultantes levam a um progresso mais rápido para cenários subsequentes.
- Capítulo 5, "Organizando a documentação", foca nos desafios que as equipes frequentemente enfrentam quando têm muitos arquivos de funcionalidade. Conforme a equipe WIMP estrutura seu arquivo de funcionalidade, esclarecemos a diferença entre estrutura baseada em histórias e em funcionalidade, explicamos como os cenários podem ser usados eficientemente como documentação e exploramos a formulação de funcionalidades compartilhadas e cenários de jornada.
- Capítulo 6, "Lidando com o legado", discute como o BDD pode ser introduzido em um projeto legado. Você verá quais estratégias incrementais podem funcionar e como lidar com scripts de teste manual existentes.

Em cada capítulo, a equipe WIMP enfrenta novos problemas e considera quais técnicas aplicar. Para enfatizar que não há "melhores práticas" gerais, seguimos as discussões da equipe enquanto consideram alternativas, efeitos colaterais e compensações. Lembre-se, no entanto, que este é um exemplo simplificado e, portanto, algumas discussões podem ser incompletas.

Todas as informações necessárias para acompanhar estão no texto do livro, mas se preferir, você pode baixar o código-fonte para cada capítulo (veja *Recursos online*, mais adiante neste capítulo).

Além dos seis capítulos principais, os Apêndices contêm uma folha de referência rápida de Gherkin para fornecer uma visão geral rápida dos recursos do Gherkin. Em seguida, há duas listas de navegação que ajudam você a encontrar detalhes no livro relacionados aos elementos do Gherkin e aos "problemas de formulação". Os Apêndices também incluem os arquivos finais de funcionalidades que a equipe WIMP criou.

Regras e exemplos

A descoberta é alcançada em uma sessão colaborativa durante a qual a equipe de entrega explora sua compreensão dos requisitos de uma história usando exemplos concretos. No mínimo, as perspectivas de negócios, desenvolvimento e teste devem estar representadas. Recomendamos que a oficina utilize o *Mapeamento de Exemplos*[7].

O escopo de uma *história de usuário* é definido pelas regras que serão implementadas por sua entrega. *Regra* é um sinônimo para *requisito de negócio* e *critério de aceitação*.

Cada regra é ilustrada por vários *exemplos concretos* que garantem que não haja ambiguidade na interpretação da regra. Cada exemplo descreve uma instância concreta da aplicação da regra ao documentar o resultado esperado que deve resultar de uma ação específica ocorrendo em um determinado contexto.

Exemplos devem ser capturados em qualquer formato relevante e conciso (como listas de itens, wireframes, diagramas de fluxo ou tabelas verdade). Neste livro, mostraremos como formular exemplos concretos em uma documentação legível para negócios. Uma vez automatizados, isso se torna uma *documentação viva* que pode ser confiavelmente usada para descrever com precisão o comportamento real do sistema.

Juntas, as regras e exemplos especificam o comportamento esperado do sistema.

[7] https://cucumber.io/blog/bdd/example-mapping-introduction/

BDD precisa de testadores qualificados

Os testadores continuam desempenhando um papel crucial em equipes que adotam uma abordagem orientada pelo comportamento. Ainda assim, frequentemente ouvimos falar de organizações que parecem acreditar que a automação de testes reduz a necessidade de testadores. Sentimos que é importante reiterar que isso não é verdadeiro. Pode chegar um momento em que a inteligência artificial consiga automatizar todos os aspectos do desenvolvimento de software, mas por enquanto continuamos a depender de profissionais qualificados e humanos em todas as etapas da especificação e entrega.

Apesar dos enormes benefícios da automação de testes, as organizações devem fazer tudo o que podem para manter testadores qualificados, especialmente aqueles com um extenso conhecimento de domínio, porque:

- a experiência e perspectiva de um testador são essenciais durante a descoberta;
- existem uma ampla gama de técnicas de teste especializadas que são valiosas ao longo do ciclo de vida do desenvolvimento, e;
- o teste exploratório requer um profundo conhecimento de teste e do domínio do problema.

Certamente faz sentido oferecer treinamento em habilidades de desenvolvimento para testadores, da mesma forma que faz sentido oferecer treinamento em habilidades de teste para desenvolvedores. No entanto, a organização deve reconhecer que o conhecimento de domínio é muito valioso para ser desperdiçado. Ofertas de desenvolvimento de habilidades cruzadas devem ser voluntárias, tanto na teoria quanto na prática.

Por que você deveria nos ouvir

Gáspár é o criador do SpecFlow, o framework BDD mais amplamente utilizado para .NET.

Ele é um coach independente, treinador e especialista em automação de testes, focado em ajudar equipes a implementar BDD e SpecFlow através de sua empresa, Spec

Solutions. Com mais de 20 anos de experiência no desenvolvimento de software empresarial, atuou como arquiteto e coach ágil de desenvolvedores.

le compartilha dicas úteis relacionadas ao BDD e automação de testes em seu blog[8] e no Twitter (@gasparnagy). Ele também edita um boletim mensal[9] com artigos interessantes, vídeos e notícias sobre BDD, SpecFlow e Cucumber.

Ele também trabalha em uma extensão de código aberto para Visual Studio para SpecFlow, chamada Deveroom[10] e em uma ferramenta que pode sincronizar cenários com o Azure DevOps, chamada SpecSync[11].

Seb tem sido consultor, coach, designer, analista e desenvolvedor por mais de 40 anos. Ele esteve envolvido no ciclo completo de desenvolvimento, com experiência que vai desde arquitetura até suporte técnico, e trabalhou para empresas grandes (como IBM e Amazon) e pequenas. Seb possui ampla experiência em projetos que não tiveram sucesso. Atualmente, ele é líder de Melhoria Contínua na SmartBear, ajudando a aplicar as lições que aprendeu nas práticas internas de desenvolvimento e nos roteiros de produtos.

Seb é palestrante regular em conferências, autor colaborador do livro *97 Things Every Programmer Should Know* (O'Reilly) e autor principal do *The Cucumber for Java Book* (Pragmatic Programmers).

Ele escreve em seu blog em cucumber.io[12] e compartilha conteúdo no Twitter como @sebrose.

Juntos, Seb e Gáspár possuem mais de 60 anos de experiência em software, que eles utilizam para desenvolver e fornecer treinamento e coaching para organizações em todo o mundo. Se você estiver interessado nos serviços que eles oferecem, entre em contato pelo email services@bddbooks.com.

Recursos online

- Série de Livros BDD: http://bddbooks.com

[8] http://gasparnagy.com
[9] http://bddaddict.com
[10] https://github.com/specsolutions/deveroom-visualstudio
[11] https://www.specsolutions.eu/services/specsync/
[12] https://cucumber.io/blog

- Recursos para este livro: http://bddbooks.com/resources/formulation
- Figuras do livro: http://bddbooks.com/resources/formulation/figures
- Arquivos do projeto WIMP (arquivos de funcionalidade): https://github.com/bddbooks/bddbooks-formulation-wimp

Agradecimentos

Este livro não teria sido possível sem a ajuda de: Gojko Adzic, Emily Bache, Abby Bangser, Lisa Crispin, Gary Fleming, Markus Gärtner, Janet Gregory, John Ferguson Smart, Aslak Hellesøy, Claude Hanhart, Kevlin Henney, Angie Jones, Heidi Kinsey, Liz Keogh, Ailsa Laing, Cyrille Martaire, Rob McBryde, Ken Pugh, Tom Roden, Johanna Rothman, Daniel Terhorst-North, Joe Wright, Matt Wynne.

Esta tradução não teria sido possível sem a ajuda de nossos revisores:

- Jonas Lima Fleck
- Marcos Machado Duarte

Seb Rose e Gáspár Nagy, 2021

Capítulo 2 – Limpando um cenário antigo

A linguagem Gherkin é usada por milhares de organizações em todo o mundo, e é comum encontrar cenários longos, complexos e ilegíveis em seus projetos. Infelizmente cenários longos, complexos e ilegíveis não promovem entendimento compartilhado em sua organização e exigem da sua equipe esforços constantes para mantê-los. Limpar cenários como esses atenuam os problemas citados e também oferecem uma boa oportunidade para aprender mais sobre o seu domínio.

Nesse capítulo você vai aprender os princípios básicos da formulação ao seguir a equipe WIMP enquanto eles pegam um cenário mal formulado e o melhoram. Nós olharemos mais a fundo a aplicação do BDD em projetos legados no Capítulo 6, "Lidando com o legado".

2.1 – O cenário antigo

A aplicação Onde Está Minha Pizza (WIMP, no inglês) permite que os clientes façam pedidos e os busque no restaurante (retirada pelo cliente)". Também é possível pagar o pedido apenas na retirada (pagamento na retirada). Houve alguns problemas com pedidos que nunca foram retirados ou pagos. Patricia, a dona do produto (PO, no inglês), recebeu a tarefa de desenvolver uma funcionalidade que reduza este problema, então ela está tentando entender a implementação existente das funcionalidades de retirada pelo cliente e pagamento na retirada.

Na preparação para a oficina de requisitos Patricia, Tula e Daiana se encontraram para identificar os cenários existentes que ilustram o processo de pedido em que o cliente faz a retirada. Elas encontraram somente um cenário (ver Listagem 1 abaixo), que foi escrito quando o projeto usou cenários para testes, ao invés de usá-los para facilitar a colaboração e o BDD.

Membros da equipe WIMP

Para tornar mais fácil acompanhar, as iniciais dos membros descrevem seus papéis:

- **D**aniel – desenvolvedor
- **D**aiana – desenvolvedora
- **E**va – estagiária
- **P**atricia – dona do produto (PO, no inglês)
- **T**ula – testadora (QA, no inglês)
- **U**lisses – experiência de usuário (UX, no inglês)

Patricia projeta o cenário para todos lerem.

Listagem 1 – O cenário antigo

```
Cenário: Teste de Pedido
  Dado que o horário é "11:00"
  Dado que o cliente acessa "http://teste.WIMP.com/"
  E preenche "Margherita" para "TextoDaBusca"
  Quando clica em "Buscar"
  Então deve ver "Marguerita" em "ResultadosDaBusca"
  E seleciona "Médio" em "Tamanho"
  Quando clica em "Adicionar ao carrinho"
  Então deve ver "1 item" em "ContadorDeItensDoCarrinho"
  Quando ele clica em "Fechar pedido"
  E ele seleciona "Retirar no restaurante" em "InstrucoesDeEntrega"
  E ele seleciona "Pagar na retirada" em "OpcoesDePagamento"
  E preenche "Marvin" para "NomeDoContato"
  E preenche "12334456" para "NumeroDeTelefoneDoContato"
  Quando ele clica em "Fechar pedido"
  Então ele deve ver "MensagemDeSucesso"
  Então ele não vê "MensagemDeErro"
  E ele deve ver "Obrigado pelo seu pedido!" em "MensagemDeSucesso"
  E ele deve ver "11:20" em "HoraDaRetirada"
  E ele deve ver "$14" em "ValorTotal"
```

Princípios básicos do Gherkin

A linguagem Gherkin permite que as organizações escrevam especificações de negócio legíveis que também podem ser usadas como testes automatizados. É escrita em *feature files*, que são arquivos de texto não formatado com a extensão `.feature`.

Cada arquivo de funcionalidade (*feature file*, em inglês) contém um ou mais *cenários*. Cada cenário é composto de um ou mais *passos*. Cada passo inicia com uma das 5 palavras chave: *Dado*, *Quando*, *Então*, *E*, *Mas*

"Dado", *"Quando"*, *"Então"* introduzem respectivamente as seções de contexto, ação e resultado esperado de um cenário. (Veja *Discovery* [Nagy2018, Capítulo 3]). *"E"* e *"Mas"* são conjunções que continuam a seção atual. Nós cobrimos a maior parte da sintaxe da linguagem Gherkin em Seção 3.1, "Arquivos de funcionalidade" e Seção 3.3, "Fundamentos do Gherkin". Veja *Lista de Atalhos do Gherkin?* in the Apêndices para mais detalhes.

"Eu não acho que eu já tenha visto este cenário antes," diz Patricia. "Ele não é fácil de ler como aqueles nós geralmente escrevemos."

"Você está certa," responde Tula, "Ele não foi escrito para ilustrar uma *regra* específica. Eu o escrevi como um teste de regressão automatizado. Há muito aqui que queremos reescrever."

"É muito longo e envolve muitas *regras* diferentes," diz Daiana. "Deve ser por essa razão que ele é difícil de manter atualizado. Na última vez que este cenário falhou nós passamos muito tempo tentando entender o que havia dado errado."

"Vamos melhorar esse cenário, então?" diz Patricia. "Talvez nós deveríamos tentar aplicar os princípios BRIEF aqui."

2.2 – Mantenha seus cenários BREVES (BRIEF, no inglês)

Ao longo dos anos que a linguagem Gherkin tem sido usada, uma proposta para escrever cenários tem evoluído. Porque a linguagem Gherkin é muito próxima da linguagem natural ela se torna fácil de aprender, mas assim como escrever relatórios e histórias, requer prática para fazer bem. Existem três objetivos principais que nós temos que ter em mente quando escrevemos cenários, que dão origem aos seis princípios encapsulados pelo acrônimo BRIEF.

Os objetivos

Cenários devem ser pensados como documentação, não como testes Nós escrevemos cenários para ilustrar e esclarecer o comportamento esperado do sistema. O objetivo é ser descritivo, não exaustivo.

Cenários devem habilitar colaboração entre negócio e entrega, não inibir. Cenários devem ser escritos de modo que possam ser entendidos por todos que contribuem para a criação e a evolução do sistema.

Cenários devem suportar a evolução do produto, não obstruir. Cenários que ilustram um comportamento específico não precisam ser alterados quando mudanças de comportamento não relacionadas alteram.

Os princípios

Os 6 princípios abaixo trabalham juntos para suportar os objetivos descritos acima. Para serem mais fáceis de lembrar, nós os organizamos de modo que a primeira letra de cada princípio encaixe em um acrônimo, *BRIEF*, que por sua vez representa o sexto princípio.

B	**linguagem de negócio**, **B** de *business*	Terminologia de negócio facilita a colaboração entre as disciplinas
R	**Dados reais**, **R** de *real*	Uso de dados reais ajuda a revelar suposições e casos limite
I	**Intenção revelada**	Descreve os resultados desejados, ao invés do mecanismo de como eles são alcançados
E	**Essencial**	Omite qualquer informação que não ilustra diretamente o comportamento
F	**Focado**	Cada cenário deve ilustrar uma única regra
	Breve	Cenários curtos são mais fáceis de ler, de entender e de manter

Linguagem de negócio: As palavras usadas em um cenário devem ser extraídas do domínio de negócio. Sistema de software precisam entregar valor de negócio, sendo assim os termos de negócio devem entendidos por todos os envolvidos na entrega do sistema. Portanto, nós deveríamos usar a linguagem de negócio para habilitar colaboração e garantir alinhamento.

Dados reais: Em *Discovery* [Nagy2018, Seção 3.1], nós explicamos que os exemplos devem usar dados reais e concretos. Isto ajuda a expor mais cedo no processo de desenvolvimento as condições limítrofes (também conhecidas como casos limítrofes) e as suposições subentendidas. Quando escrevendo cenários, nós deveríamos também usar dados reais quando isto ajuda a *revelar intenção.*

Intenção revelada: Cenários deveriam relevar a *intenção* que os atores no cenários estão tentando alcançar, ao invés de descrever o *mecanismo* usado para alcançar algo. Nós deveríamos iniciar definindo um nome pro cenário que revela sua intenção, e em seguida garantir que cada linha no cenário descreva *intenção, e não mecanismo.*

Essencial: O propósito de um cenário é ilustrar como uma regra deveria se comportar. Qualquer parte de um cenário que não contribui diretamente para este propósito é *incidental* e deveria ser removida. Se eles são importantes para o sistema, eles serão cobertos em outros cenários que ilustram outras regras. Adicionalmente, qualquer cenário que não adicione entendimento ao leitor sobre o comportamento esperado não tem lugar na documentação.

Focado: A maior parte dos cenários devem ser focas em ilustrar uma *única regra.*

É mais fácil atender esse princípio se você deriva seus cenários dos exemplos capturados durante uma *oficina de requisitos*.

Por último, mas não menos importante, cenários devem ser breves, tanto quanto devem ser BRIEF.

Breve: Nós sugerimos que você tente restringir a maior parte de seus cenários a cinco passos ou menos. Isto os fará mais fáceis de ler e muito mais fáceis de entender.

No restante deste capítulo, você verá a equipe aplicar estes princípios. Nos próximos capítulos, nós iremos nos aprofundar e ver o BRIEF aplicado à escrita de novos cenários. Nós indicaremos quais princípios do BRIEF a equipe aplica em cada seção ao destacá-los com este símbolo:

Linguagem de negócio (**B**), **E**ssencial

História do Seb: Acrônimos

Gáspár e eu temos explicado os conceitos por trás por muitos anos, mas nós nunca conseguimos criar um acrônimo memorável. em uma das nossas oficinas "Escrevendo Cenários BDD Melhores" na *European Testing Conference* em 2018 nós tivemos Gojko Adzic na audiência. Ele pareceu curtir a oficina, mas gastou algum tempo rabiscando em um pedaço de papel. No fim da oficina ele veio e nos contou que ele tinha tentado (mas falhado) em transformar nossos tópicos em um acrônimo. "Isto funcionou para o acrônimo INVEST[a] do Bill Wake."

Nós seguimos o conselho, e BRIEF é o resultado. Agradecemos por nos encorajar, Gojko!

[a] https://en.wikipedia.org/wiki/INVEST_(mnemonic)