Histórias de Usuário

Por que e como escrever requisitos de forma ágil?

4ª edição

Rafael Helm

wildtech.com.br

“Uma documentação ruim vai gerar um software ruim.”

Prefácio 1: Guilherme Lacerda

Quando pensamos em registrar as funcionalidades de um software, talvez, você já tenha refletido sobre alguns questionamentos: Como os analistas do meu time registram os requisitos investigados? Que ferramentas/técnicas eles usam? Isso ajuda, de fato, no desenvolvimento do software do meu time?

Ao avaliar estes questionamentos, também, talvez você pense: Eles usam Casos de Uso. Legal! Só lembrando que o Caso de Uso é um artefato informal, textual, que descreve um processo de negócio sobre a perspectiva de um ator. De nada tem haver com aquela representação visual da UML. É importante fazer este registro.

Mas, no pior dos casos, simplesmente o time não tem ferramenta ou técnica que os ajude no dia a dia.

Tanto os Casos de Uso quanto as Histórias de Usuário tem a mesma natureza. Porém, a História de Usuário traz em seu DNA alguns elementos que, de uma forma simples, busquem responder às seguintes perguntas:

1) Para quem é esta funcionalidade?
2) Qual o ganho/benefício de usá-la?
3) Como posso avaliá-la e testá-la?

O primeiro questionamento, faz com que a gente reflita que um requisito não funciona de forma isolada. Ele é para alguém. Alguém precisa usar aquela funcionalidade, por algum motivo. Isso nos leva ao segundo questionamento. Esta funcionalidade, para ter valor para o negócio, precisa trazer ganhos para quem a usará. E, finalmente, o terceiro questionamento remete aos critérios de aceitação. Isso, se você perceber bem, traz a visão de testabilidade da funcionalidade em sua definição. Estes critérios, além de ajudar a perceber as diferentes situações no qual esta funcionalidade deve se enquadrar, complementam as suas informações.

Estes elementos representam a essência das Histórias de Usuário.

Esta técnica ganhou reconhecimento juntamente com os Métodos Ágeis e, hoje, ela tem vida própria.

Independente do papel que você desempenhe (analistas de sistemas, analistas de negócio, product owners, desenvolvedores ou até mesmo cliente), ela pode ajudá-lo.
E o mais legal: você pode combiná-la com outras técnicas. A minha preferida são os protótipos. E a sua, qual é?

Neste livro, o Rafael traz uma abordagem pragmática de como usar esta técnica de forma eficaz dentro da sua equipe. Ele tem usado esta técnica de forma diligente em seus projetos, além de ensinar em treinamentos presenciais e online com públicos dos mais diferentes segmentos (setor público, privado e indústria).

Ao longo do tempo, sempre aprendemos novas técnicas e ferramentas de desenvolvimento de software. E ao aprendê-las, as colocamos na nossa caixa de ferramentas, para em algum momento oportuno, usá-las. Certamente, Histórias de Usuário é uma ferramenta que deve constar na sua caixa de ferramentas.

Aproveite a leitura e a experiência do Rafael. ;)

Guilherme Lacerda
Consultor e Professor
https://www.linkedin.com/in/guilhermelacerda

Prefácio 2: Thiago Esser

1) Desenvolver software com uma documentação extremamente técnica e longa está para usar um produto pela primeira vez e só conseguir ativá-lo após ler cuidadosamente o manual de instruções, assim como 2) desenvolver software a partir de Histórias de Usuários (HU) está para usar um produto pela primeira vez e descobrir intuitivamente, pelo seu design, como esse produto funciona.

Deixa eu tentar melhorar, contando uma história.

Certa feita, eu e o Rafael Helm fomos dar um curso sobre Métodos Ágeis (Agile) e User Experience (UX) em Brasília. Ele falando mais de Agile, e eu de UX. A partir do retorno dos alunos, fizemos ajustes para os dois assuntos ficarem mais bem integrados. Também descobrimos que tínhamos uma grande afinidade, nos tornamos grandes amigos, além de sócios. Temos habilidades complementares: o cara é um baita assador de churrasco, e eu mando bem no desenho.

É provável que o primeiro parágrafo tenha exigido muito mais neurônios e não tenha chamado tanto sua atenção quando a história que acabei de contar. Ela ilustra bem o que aprendi sobre as HU com esse livro do Rafael:

- São universais. Não há grandes pré-requisitos para entendê-las e se conectar com elas.

- São visuais. Trazem um imagem mais viva do que conceitos abstratos.

- São colaborativas. Elas podem ser enriquecidas: a HU da Product Owner (PO) é complementada pelo protótipo da Designer; a pessoa dos testes cria cenários que servem para o desenvolvimento...

- São iterativas. Podem receber incrementos e ajustes a todo instante.

Mas agora chega de história! É hora de escrever a sua a partir do que irá aprender nas páginas a seguir.

Thiago Esser
UX Designer e co-fundador da UXConf BR.
https://www.linkedin.com/in/thiagoesser

Prefácio 3: Daniel Wildt

Eu sempre gostei de padrões. Gosto quando consigo combinar com quem trabalha comigo o que vamos fazer e como vamos fazer. Uma certa estrutura e contexto me ajudam a funcionar melhor.

Lá por 1998 comecei a conhecer estratégias como Casos de Uso, e depois pensamento em serviços, que me ajudavam a pensar em modelagem de sistemas como jornadas. Isso lá no início dos anos 2000. Lá naquela época já entendia que os documentos eram padrões que ajudavam uma equipe no seu processo de funcionamento. Era uma estrutura. E poderia ser simples.

E foi o que mais me chamou atenção no extreme programming e nas metodologias ágeis, quando conheci em 2003. Muita gente fala dos mais diversos princípios, mas eu sou um preguiçoso. E as metodologias ágeis conectavam com muito do que buscava fazer no meu dia a dia profissional, principalmente quando passo pelo princípio da simplicidade: a arte de maximizar o trabalho que não precisa ser feito. Isso envolve conectar com o necessário, com o que entrega valor e resolve o problema. Nada mais.

Nesta época eu começo também a participar de um grupo que buscava encontrar padrões para funcionar quando se pensava em especificação de sistemas. Eu tinha dois aprendizados que trazia das minhas buscas e estudos:

1. O conceito 3C que aprendi com Ron Jeffries, um dos criadores do eXtreme Programming: cartão, conversação e confirmação. O cartão é um lembrete para conversa, as conversas servem para descobrir e explorar. E a partir disso vem as confirmações, e o nosso aprendizado sobre algo. Assim fica mais fácil fazer a validação do que vai ser construído.
2. O modelo 5W2H, onde faço perguntas de precisão para enquadrar um problema ou atividade. E assim eu quero saber o que precisa ser feito (lembrete para conversa), quem vai ser beneficiado por isso (entrega de valor), porque eu preciso deixar o meu sofá para trabalhar (preguiça me ajudando a entender porque deveria fazer algo) e principalmente me indique como eu sei que isso está pronto?

Este modelo de pensamento era totalmente aplicável nos documentos de caso de uso, já que eu poderia organizar as seções que poderiam fazer sentido para mim, para fins de documentação. E a cada sessão de conversa com clientes, conseguia avançar na escrita e mantendo um histórico de conversas.

As histórias de usuário me ajudaram a entender que o foco não é na funcionalidade, mas sim na necessidade. Isso quebrou muitos paradigmas e até muito da minha estrutura ao desenvolveu software. Passo a ter mais respeito por cada linha de código que adiciono em um sistema.

Agora a minha preocupação era entender benefícios de fazer algo, e principalmente entender como validar "o critério de preguiça". Me mostra porque algo precisa ser feito!? Adoro a pergunta: e se isso não for feito, o que acontece? E assim a jornada conectava com a necessidade real, me permitindo ir além e somente a partir disso pensar na entrega funcional.

Histórias nos permitem focar em quem é a personagem da mesma. Permite conectar com as pessoas que são importantes nas jornadas, e permite manter a curiosidade sempre presente e alta. Quem atua com histórias de usuário atua com a curiosidade. Com perguntas, explorando opções e buscando qual é a real necessidade das pessoas que estamos ajudando.

—

O Rafael Helm me acompanha nas jornadas profissionais faz algum tempo, desde 2001. Fundamos grupo de usuários de tecnologia (Delphi!), trabalhamos em diferentes clientes e projetos. Um dia o Rafael disse que queria palestrar, e antes dele notar estava em um palco palestrando. Um dia ele disse que queria dar aulas, e antes dele notar estávamos sócios na mesma empresa dando aulas juntos. Fizemos um livro juntos, e fizemos um ebook sobre histórias que segue ajudando e sendo baixado por muitas pessoas. Talvez tenha sido o que trouxe você até este livro. :)

Sobre o autor

Rafael Helm é Consultor, Instrutor e sócio na Wildtech (http://wildtech.com.br), empresa de treinamento e consultoria de práticas ligadas ao desenvolvimento ágil de software e UX.

Seu principal objetivo é ajudar empresas e equipes na entrega de software de valor sem abrir mão da qualidade, respeitando acordos de custo e prazo pré estabelecidos sempre atendendo ~~o escopo do projeto~~ as expectativas do cliente e do negócio. Na busca destes objetivos a disciplina, o trabalho padronizado, e o ritmo sustentável não são itens opcionais.

Rafael é coautor de um livro sobre eXtreme Programming e é cofundador da UXConf BR, relevante conferência sobre UX Design.

Para falar com o autor basta encontrá-lo no twitter @rafaelhelm, no linkedin (https://www.linkedin.com/in/rafaelhelm) ou por e-mail em rhelm@wildtech.com.br.

Sumário

Introdução

Por mais que as tecnologias de desenvolvimento estejam evoluindo cada vez mais rápido, o desenvolvimento de software ainda é um processo complexo. São muitas fases envolvidas:

- Análise de negócios;
- Análise de requisitos;
- Design;
- Projeto de banco de dados;
- Desenvolvimento;
- Testes;
- Implantação.

Dependendo da realidade da sua empresa ou equipe, o seu processo pode ser mais simples ou mais complexo do que o citado acima, mas pelo menos duas fases geralmente todos os processos de desenvolvimento de software possuem: Especificação e desenvolvimento.

Ao contrário do que muitos acreditam, o desenvolvimento de software não começa através das mãos do desenvolvedor quando elas iniciam a digitar linhas de código. O desenvolvimento de software começa na fase de análise, e principalmente na especificação dos requisitos.

Não basta que sua equipe possua desenvolvedores altamente capacitados e responsáveis se a especificação que eles recebem é incompleta, superficial, ou burocrática demais.

Se a especificação do software é ruim, o resultado do trabalho provavelmente será um software igualmente ruim.

Saiba que é possível especificar software de uma forma muito mais efetiva, simples e até divertida do que o mercado normalmente tem feito ao longo dos anos.

Essa forma de especificação de requisitos mais eficiente se chama **Histórias de Usuário** (user stories), que é uma prática ágil de desenvolvimento de software, e é o tema central deste livro.

Ao longo dos capítulos vamos apresentar a motivação para escrever requisitos utilizando histórias de usuário, também vamos ensinar como escrever, além de citar ricos exemplos que poderão ser utilizados por você como guias durante suas primeiras histórias de usuário.

Importante:

Se você não tem nenhum conhecimento prévio sobre histórias de usuário, sugerimos que você leia o livro seguindo sua sequência natural. Mas caso você já tenha uma noção sobre o

assunto, então fique a vontade para navegar diretamente até determinado capítulo para relembrar conceitos e/ou tirar dúvidas.

Boa leitura!

Por que escrever histórias de usuário?

Ao longo dos anos temos visto muitas empresas tratarem seus desenvolvedores como funcionários de uma linha de montagem. Ou seja, algumas empresas ainda acham que o trabalho de desenvolvimento de software é algo repetitivo, e acreditam que apenas dizer ao desenvolvedor **o que fazer** é suficiente.

Mas acontece que o desenvolvimento de software é um processo complexo, e na maioria das vezes não se trata de um trabalho repetitivo. É comum um desenvolvedor encontrar várias formas de desenvolver uma mesma funcionalidade, e para que ele possa tomar uma decisão correta ele precisa de mais informações do que apenas saber o que fazer.

É importante que ele saiba **para quem** está sendo criada a nova funcionalidade.
Por exemplo, se o desenvolvedor souber que está desenvolvendo um recurso que será usado por vendedores que realizam em média 50 visitas por dia, é bem provável que ele desenvolva um design pensando mais em produtividade do que em elegância.
Também é vital que ele saiba o motivo desta funcionalidade, ou seja, por que esta funcionalidade está sendo desenvolvida.

Dando mais um exemplo: Se um desenvolvedor sabe que está alterando uma funcionalidade por que é necessário reduzir o tempo médio de um atendimento, ao terminar o desenvolvimento ele vai se preocupar em verificar quanto tempo leva efetivamente um atendimento com a nova interface versus o tempo deste mesmo atendimento executado na interface antiga.

Ou seja, repare que nos exemplos citados anteriormente, informar **para quem** e **por que** a funcionalidade está sendo desenvolvida ajudou o desenvolvedor a tomar decisões mais alinhadas com a necessidade do cliente. Isto tem como consequência um ganho significativo de qualidade!

Então por que escrever histórias de usuário?
Porque nós queremos maximizar a possibilidade de você e seu time desenvolver e entregar corretamente na primeira tentativa! E o seu cliente também. :)

Existe um padrão para escrever?

Sim existem alguns padrões, mas isto não importa!
Como assim? O que importa é você entender a estrutura base de uma história de usuário, ou seja, as informações fundamentais que precisam constar numa boa especificação de requisitos.

Como já vimos no capítulo anterior existem 3 informações que são fundamentais nas histórias de usuário, são elas:

- **Quem?** Para quem estamos desenvolvendo a funcionalidade.

- **O que?** Uma descrição resumida da funcionalidade em si.

- **Por que?** O motivo pelo qual o cliente precisa desta funcionalidade. Se possível citando o valor de negócio obtido.

Normalmente para responder as três perguntas citadas acima nós usamos o SENDO... POSSO... PARA QUE...

Um exemplo:
SENDO um vendedor que realiza 50 visitas por dia
POSSO consultar as últimas compras de cada cliente
PARA QUE ao chegar no cliente eu possa consultar qual foi sua última compra, e assim conseguir negociar com ele estando melhor informado.

Repare que no SENDO nós identificamos o perfil do usuário que vai usar a funcionalidade, no POSSO a funcionalidade em si que precisa ser desenvolvida e no PARA QUE a motivação da funcionalidade, incluindo o valor de negócio.

Com estas informações, o desenvolvedor vai conseguir trabalhar “mais armado”, e provavelmente vai criar uma funcionalidade mais bem elaborada do que se recebesse apenas a necessidade do cliente, sem o detalhamento de **quem** vai usar e **por que** vai usar.
Entendido? Mas ainda falta uma informação muito importante, que é o **como testar?** Veremos isto no próximo capítulo.

Como testar? BDD!

No capítulo anterior entendemos melhor a importância do **quem**, **o que**, e **por que**, mas ainda falta um ponto muito importante para fecharmos a estrutura de uma boa história de usuário: **O como testar?**
Para isto podemos usar a técnica do BDD (Behavior Driven Development) de Dan North, onde as palavras chave Dado que... Quando... Então... nos apoiam na criação de ricos cenários de teste.

Exemplos:

Cenário 1: Estoque disponível
Dado que o estoque da coca-cola é de 50 unidades
Quando informo uma venda de 40 unidades
Então a venda é registrada
    E o estoque passa a ser de 10 unidades

Cenário 2: Estoque indisponível
Dado que o estoque da coca-cola é de 50 unidades
Quando informo uma venda de 60 unidades
Então a venda não é registrada
    E é exibida na tela a mensagem “estoque insuficiente!”
Repare que nos exemplos anteriores nós usamos o “Dado que” para indicar o cenário atual, o “quando” para indicar a ação do usuário, e o “Então” para indicar como o software vai reagir.
Podemos também usar o “E” e o “OU” para criar cenários de teste ainda mais ricos.

Exemplos:

Cenário 1: Estoque disponível, venda limitada a 30
Dado que o estoque da coca-cola é de 50 unidades
    E a venda máxima por cliente é limitada a 30 unidades
Quando informo uma venda de 20 unidades
Então a venda é registrada
    E o estoque passa a ser de 30 unidades

Cenário 2: Venda com cartão indisponível para valores abaixo de 20,00
Dado que o valor da venda é de 10,00

E o valor mínimo de vendas para cartão é de 20,00
Quando informo que o meio de pagamento é cartão de crédito
OU informo que o meio de pagamento é cartão de débito
Então a venda não é registrada
E é exibida na tela a mensagem “Meio de pagamento inválido! Para valores inferiores a 20 reais somente dinheiro.”
Importante: Você não precisa escrever os critérios de aceitação exatamente desta forma. Mas é interessante que você registre de alguma forma os testes que devem ser realizados para que a história de usuário possa ser bem testada.

Nós particularmente gostamos muito de usar o “Dado que”, “quando”, “então”, mas fica a seu critério.

Para saber mais sobre BDD acesse a Wikipédia, lá você vai encontrar um ótimo artigo sobre o assunto.

O conceito INVEST

INVEST é um acrônimo (em inglês), que pode nos ajudar a revisar as histórias de usuário para verificar se elas foram bem escritas.

I ndependent (deve ser independente)

N egotiable (deve ser negociável)

V aluable (deve agregar valor para o cliente)

E stimable (deve ser possível estima-la)

S mall (deve ser pequena)

T estable (deve ser testável)

Resumindo: Uma boa história de usuário **não deve depender** de outra, deve ser possível **negociá-la** de forma que você possa alterar sua prioridade e ordem de execução com o cliente, deve **agregar valor**, deve ser **estimável**, deve ser **pequena** (até para poder ser estimada), e deve ser **testável**.
Na prática em alguns casos pode ser bem difícil escrever histórias de usuário INVEST, mas com o tempo e prática vai ficando mais fácil. Então não desista. ;)

Cartão, conversação, Confirmação! O conceito 3C.

Fichas de papel, cartões de papel ou index cards, são uma excelente forma de manter a vista novas ideias para um produto de software. E a melhor característica delas é o espaço limitado.
Como assim? Você não vai conseguir colocar toda informação necessária na ficha. E isto é bom, pode acreditar.
Em 2003 quando eu estava estudando eXtreme Programming, ouvi uma história do Ron Jeffries sobre 3C. E desde então eu aplico e ensino isto, pelo valor que esta prática agrega no dia a dia de um projeto.

O conceito do 3C é baseado em iniciar com a escrita de uma ideia em um cartão, para que possamos lembrar. O **cartão** é o primeiro C. E ele leva ao próximo, gerando um "lembrete para a **conversação**".

Que é o que precisamos gerar, conversas. O objetivo com isto é validar as ideias, com pessoas que podem ajudar no tópico. O melhor nestas conversas é criar exemplos que ajudem a validar a mesma.
Estes exemplos acabam virando depois cenários de aceitação da história. Se é um cálculo, exemplos de cálculos. Através deste processo, criamos um "cartão executável". E este é o nosso segundo C.
Ah, um cartão normalmente possui um documento auxiliar, onde o requisito em questão é documentado seguindo os padrões que a equipe utiliza.

Estas conversas ajudam o time a identificar alguns atributos para os cartões, exemplo?
- senso de valor
- prioridade
- risco associado
- qualquer-atributo-que-o-time-consiga-ver-valor.

O terceiro C é sobre **confirmação**. Através das conversas com o time e clientes poderemos entender como validar o cartão e confirmar que o que temos definido é o necessário para "fazer acontecer". E então é isto que precisamos buscar, confirmação! E dos nossos clientes! Eles irão confirmar sua ideia e ajudar a mesma a crescer.

Bugs também viram histórias de usuário?

Não! Nós não escrevemos histórias de usuário para registrar erros. Histórias de usuário são uma forma ágil de especificação de **novos** requisitos, ou para especificação de **evoluções** de requisitos.
Mas isto não quer dizer que nós não vamos te mostrar uma forma efetiva de registrar relatos de bugs. ;)

Ao longo dos anos nós obtivemos muito sucesso na correção de bugs nos times que trabalhamos. Ou seja, temos conseguido resolver os bugs na primeira tentativa.
Ok, sabemos que os bugs não devem ocorrer, mas infelizmente eles ocorrem.
Então veja na página a seguir o nosso modelo para relato de bug.

LOCAL: Nome do Sistema - Módulo e Menu relacionado

VERSÃO:
Identificar em que versão do sistema envolvido o problema pode ser repetido. Importante identificar se o problema pode ser repetido na última versão.

PRÉ-CONDIÇÕES:
* Identifique o que deve estar configurado no ambiente para que o problema pode ser repetido;
* Pode ser uma lista de configurações a serem marcadas;
* Ou simplesmente a indicação de que uma base de dados específica deve ser usada.

PASSOS PARA REPRODUÇÃO DO ERRO:
1) Monte uma lista indicando os passos que devem ser realizados para repetir o erro;
2) Você pode ser específico e identificar o que deve ser preenchido em cada campo;
3) Principalmente se uma base de dados específica está sendo usada para trabalhar;
4) Deve ser possível para qualquer pessoa repetir o erro lendo esta lista de passos.

ERRO:
Mostrar o erro que está acontecendo. Pode ser com uma identificação do que está acontecendo de errado - e muito importante: mostrar contexto de negócio identificando porque a situação atual é um erro.

SITUAÇÃO DESEJADA:

Descreva a situação que o sistema vai mostrar, identifique configurações que não estão sendo consideradas, mostre o que deve ser modificado pensando em regras de negócio para resolver a situação.

Segue um exemplo:

LOCAL: SoftVendas – Módulo Mobile – Tela de vendas de produtos

VERSÃO:
Identificado na última versão (03.50), o problema não ocorre em versões anteriores.

PRÉ-CONDIÇÕES:
* Acessar o ambiente de homologação;
* Logar com usuário “alfredo”, senha “xyz9988”;

PASSOS PARA REPRODUÇÃO DO ERRO:
1) Uma vez já logado no sistema mobile, acesse o menu “Vendas”;
2) Selecione um cliente qualquer e abra uma nova venda;
3) Na tela de listagem de produtos, selecione qualquer produto;
4) Após selecionar um produto informe a quantidade a ser vendida (pode ser 10), e no campo desconto informe um desconto (pode ser 10% de desconto).

ERRO:
Mesmo após informar o desconto, o valor total do produto segue sendo o mesmo que era antes (sem o desconto).

SITUAÇÃO DESEJADA:
Que o valor total do produto considere o desconto aplicado, ou seja:
Valor total do produto = (Valor unitário * Quantidade) – Desconto.
Exemplo: Se valor do produto é 90,00, a quantidade informada é 10, e o percentual de desconto é informado é 10%, então o valor total do produto deve ser 810,00.

Algumas considerações sobre o exemplo citado:

Repare como as seções PRÉ CONDIÇÕES e PASSOS PARA A REPRODUÇÃO DO ERRO, são importantes para fazer o erro acontecer.
Perceba também que na seção SITUAÇÃO DESEJADA além de citar a explicação do que deve ocorrer nós também citamos um exemplo prático, neste caso com um exemplo real do cálculo de preço total do produto.

Ainda sobre o exemplo, verifique que não usamos emoção no relato do defeito, ou seja, não existe nenhuma frase parecida com “mais uma vez ocorreu um erro primário na aplicação”, ou “é inadmissível que erros como este ocorram numa funcionalidade tão importante do nosso software”.
Relatos carregados de emoção, frustração ou cobrança não são efetivos. O importante no relato de um defeito é (1) mostrar como repetir o problema, (2) detalhar o problema, (3) apresentar o comportamento esperado.

Esperamos que este modelo de relato de bug ajude você a melhorar a qualidade da especificação dos defeitos. Afinal de contas eles não devem acontecer, mas se acontecer que pelo menos eles sejam resolvidos na primeira tentativa. :)

Exemplo 1: Saque no caixa eletrônico

Vamos imaginar que você trabalha em um sistema bancário de auto atendimento (caixa eletrônico).
Seu cliente envia para você um email solicitando e explicando como funciona o saque do banco:

"Olá! Precisamos disponibilizar a operação de saque no caixa eletrônico.
Segue as regras do banco para saques em caixas eletrônicos:
- Por questões de segurança o valor máximo de cada saque é de 800,00;
- Os saques só estão liberados entre 6h00min e 22h59, em qualquer dia, útil ou não;
- O saldo do cliente não pode ficar negativo, exceto se ele possuir limite de cheque especial;
- O cliente jamais poderá ultrapassar seu limite de cheque especial;
- Deve ser impresso um comprovante de saque ao final da operação, (se o cliente assim desejar)."

Como você transformaria este email do cliente em uma história de usuário?

Segue um exemplo:

SENDO um cliente correntista do banco
POSSO sacar dinheiro em caixas eletrônicos
PARA poder comprar em estabelecimentos que não aceitam cartão de débito/crédito

Cenário 1: Horário limite
DADO QUE são 5h00
E já estou autenticado no caixa eletrônico
QUANDO solicito sacar 10,00
ENTÃO o sistema apresenta a mensagem "Os saques somente são permitidos entre 6h00min e 22h59"
E o saque não é realizado

Cenário 2: Valor máximo de saque
DADO QUE a hora atual está entre 6h00min e 22h59min
E já estou autenticado no caixa eletrônico
QUANDO solicito sacar 1.000,00
ENTÃO o sistema apresenta a mensagem "O valor de um único saque no caixa eletrônico está limitado a R$ 800,00"

E o saque não é realizado

Cenário 3: Saldo insuficiente (cliente não tem limite)
DADO QUE a hora atual está entre 6h00min e 22h59min
E já estou autenticado no caixa eletrônico
E meu saldo é +600,00
E não tenho limite de cheque especial
QUANDO solicito sacar 700,00
ENTÃO o sistema apresenta a mensagem "Saldo insuficiente"
E o saque não é realizado

Cenário 4: Saldo insuficiente (cliente tem limite)
DADO QUE a hora atual está entre 6h00min e 22h59min
E já estou autenticado no caixa eletrônico
E meu saldo é +100,00
E meu limite de cheque especial é 500,00
QUANDO solicito sacar 700,00
ENTÃO o sistema apresenta a mensagem "Saldo insuficiente"
E o saque não é realizado

Cenário 5: Saldo disponível (sem usar limite)
DADO QUE a hora atual está entre 6h00min e 22h59min
E já estou autenticado no caixa eletrônico
E meu saldo é +600,00
QUANDO solicito sacar 200,00
ENTÃO o sistema libera o dinheiro no caixa eletrônico
E meu saldo passa a ser +400,00
E a tela de emissão de impressão de recibo é exibida

Cenário 6: Saldo disponível (usando limite)
DADO QUE a hora atual está entre 6h00min e 22h59min
E já estou autenticado no caixa eletrônico
E meu saldo é +100,00
E meu limite de cheque especial é 500,00
QUANDO solicito sacar 500,00
ENTÃO o sistema libera o dinheiro no caixa eletrônico
E meu saldo passa a ser -400,00
E a tela de emissão de impressão de recibo é exibida

Cenário 7: Emissão de recibo (confirmação de impressão)
DADO QUE meu saque foi autorizado
E a tela de impressão de recibo está sendo exibida
QUANDO eu confirmo a impressão do recibo
ENTÃO o recibo é impresso
E o sistema retorna a tela inicial do caixa eletrônico

Cenário 8: Emissão de recibo (impressão ignorada)
DADO QUE meu saque foi autorizado
E a tela de impressão de recibo está sendo exibida
QUANDO eu indico não imprimir o recibo
ENTÃO o sistema retorna a tela inicial do caixa eletrônico

Repare como a história de usuário ficou mais rica do que o email do cliente.
Nos casos em que o sistema precisou emitir uma mensagem de erro, ela já estava especificada no próprio critério de aceitação.
Em todos os casos que o saldo foi manipulado nós registramos exemplos práticos nos critérios de aceitação. Isto ajuda muito no processo de teste.
Agora pare e reflita. Comparando o email do cliente com a história de usuário, qual especificação é mais passível de bugs?
Provavelmente o email, pois ele cita de forma superficial cada cenário de teste, enquanto que a história de usuário detalha melhor cada um dos cenários.

Exemplo 2: Validando tamanho de arquivo

Imagine que você é responsável por manter um serviço de importação de arquivos, chamado **SIA** (Serviço de Importação de Arquivos).
Este serviço roda em background no servidor e não possui interface gráfica.
O serviço fica monitorando um diretório FTP e importando todos os arquivos que "caem" neste diretório.
Agora imagine que o responsável pela infraestrutura dos servidores te mandou o seguinte email.

"Olá! Nossos servidores não estão conta do recado quando arquivos de importação muito grandes são enviados pelos usuários.
Então não podemos processar arquivos com mais de 10Mb, para que o processador do servidor não seja sobrecarregado."

Como você transformaria este email do responsável pela infraestrutura em uma história de usuário?

Segue um exemplo:

SENDO o módulo SIA
NÃO POSSO processar arquivos com mais de 10Mb
PARA que os servidores não sejam sobrecarregados

Cenário 1: Arquivo com tamanho OK
DADO QUE o arquivo possui até 10mb
    E seu layout é válido
QUANDO o SIA for processar o arquivo
ENTÃO o arquivo é processado normalmente

Cenário 2: Arquivo muito grande
DADO QUE o arquivo possui mais de 10mb
    E seu layout é válido
QUANDO o SIA for processar o arquivo
ENTÃO o arquivo não é processado
    E um log é gerado no sistema indicando que o arquivo não foi processado
    E um email é enviado para o usuário que submeteu o arquivo via FTP

Repare em algumas características interessantes desta história:
Como estamos realizando uma mudança no sistema solicitada pelo pessoal de infraestrutura, e este pessoal não é usuário do sistema nós acabamos utilizando no SENDO o próprio SIA.
Quando o benefício da história não se refere a negócio você pode citar o próprio módulo do sistema na seção SENDO.
Repare que usamos o NÃO POSSO para indicar o que sistema não deverá mais fazer. Usamos a negação por achar que ficaria mais claro do que usar uma afirmação. Esta decisão fica ao critério de quem escrever a história, o importante é que fique claro para quem for ler a história.
No PARA nós informamos a motivação que solicitou a validação de tamanho máximo de arquivos antes do processamento.

Sobre os cenários:
O cenário 1 é bem óbvio e previsível, mas vale a pena analisarmos o cenário 2.
Repare que nós incluímos a gravação de um log e o envio de um email, nos casos em que o arquivo não foi processado, mesmo que isto não tenha sido solicitado pelo responsável da infraestrutura.
Nós fizemos isto porque mesmo sendo um requisito não funcional, nós acabaríamos afetando os usuários nos casos em que os arquivos com mais de 10Mb fossem ignorados, então adicionamos o log e o email para avisar os usuários.

Alguns Lembretes valiosos

- Qualidade de software começa na especificação.
- Se a especificação do software é ruim, o resultado do trabalho provavelmente será um software igualmente ruim.
- Além de “o que fazer” o desenvolvedor também merece saber “para quem” e “por que” cada funcionalidade será desenvolvida.
- Dedique atenção especial aos critérios de aceitação. Eles estão diretamente ligados a como seu software será testado.
- Ao acabar de escrever uma história de usuário, verifique se ela ficou INVEST.
- Evite ao máximo escrever histórias de usuário grandes e sempre pergunte a si mesmo se uma história pode ser “quebrada”.

Terminei o livro, e agora?

Legal você não ter abandonado a leitura do livro. Muito obrigado pela consideração!

Isto provavelmente significa que especificar requisitos no formato histórias de usuário devem ter feito algum sentido para você. Ou quem sabe você apenas não tinha nada melhor para fazer mesmo. :)

De qualquer forma deixarei algumas sugestões sobre o que você pode fazer a partir de agora:

- Se você gostou do livro então então ajude a divulgá-lo e compartilhe o link <LINK DO LIVRO NA EDITORA CAROLI> no facebook, twitter, linkedin, e etc. M apoie nesta caminhada para tornar as especificações de requisitos de software mais simples, objetivas e amigáveis.

- Mas se você não gostou do livro então me envie um email dizendo o motivo, eu prometo não ficar magoado. Pode escrever para rhelm@wildtech.com.br

- Pratique! Tente escrever algumas histórias de usuário. Comece pelos requisitos mais simples. Use os exemplos do livro como guias de referência.

Agora vá econte boas histórias de usuário para o seu time de desenvolvimento. :)