

POR RAFAEL HELM E DANIEL WILDT

]HU[

HISTÓRIAS

DE USUÁRIO

POR QUE E COMO ESCREVER  
REQUISITOS DE FORMA ÁGIL?

[HistoriasDeUsuario.com.br](http://HistoriasDeUsuario.com.br)

3<sup>a</sup> EDIÇÃO

# AVISOS LEGAIS

**REDISTRIBUIÇÃO:** Você concorda que não irá copiar, redistribuir ou explorar comercialmente qualquer parte deste documento sem a permissão expressa do autor.

**AUTORIA:** Rafael Helm e Daniel Wildt

**EDITOR:** Lucas Engel

Copyright © 2014 - Todos os direitos reservados

3<sup>a</sup> edição publicada em novembro de 2014

[historiasdeusuario.com.br](http://historiasdeusuario.com.br)

*“Qualidade de software  
começa na especificação.”*

- Rafael Helm

## SOBRE OS AUTORES

Rafael Helm e Daniel Wildt são sócios da Wildtech, que é uma empresa de treinamento e consultoria de práticas ligadas ao desenvolvimento ágil de software.

Seu principal objetivo é ajudar pessoas a serem melhores profissionais, a realizarem mais e irem em busca daquilo que gera felicidade, além de ajudar times a melhorarem continuamente e organizações a se tornarem conscientes e em busca de aprendizado contínuo.

Para falar com Rafael e Daniel basta encontrá-los no twitter [@rafaelhelm](https://twitter.com/rafaelhelm) e [@dwildt](https://twitter.com/dwildt).

Ou se preferir mande email para [contato@wildtech.com.br](mailto:contato@wildtech.com.br)

# AGRADECIMENTOS

Assim que liberamos a primeira edição do livro já começamos a receber ótimos feedbacks por email. E foi assim que chegaram até nós valiosas críticas construtivas.

Lemos todos os emails, e cada um contribuiu de alguma forma para o lançamento desta segunda edição, revisada e ampliada.

Então nada mais justo do que agradecer algumas pessoas que enviaram feedbacks que nos levaram a melhorar o livro, (em ordem alfabética).

- **Ademílson F. Tonato**
- **Fabrício de Royes Mello**
- **Frederico Macedo**
- **Hugo Estevam Longo**
- **Mateus Leonardi**
- **Vanessa Me Tonini**

E um special thanks também ao **Lucas Engel** pelo brilhante trabalho realizado na diagramação e capa desta terceira edição.

Agora chega de emoção e vai ler livro! :)

# CONTEÚDO

Avisos Legais.....	2
Sobre os Autores.....	4
Agradecimentos .....	5
Introdução .....	7
Por que escrever histórias de usuário? .....	9
Existe um padrão para escrever? .....	10
Como testar? BDD!.....	11
O conceito INVEST.....	13
Cartão, conversação, Confirmação! O conceito 3C....	14
Bugs também viram histórias de usuário? .....	15
Exemplo 1: Saque no caixa eletrônico.....	19
Exemplo 2: Validando tamanho de arquivo .....	23
Priorizando funcionalidades .....	25
Alguns Lembretes valiosos.....	27
Terminei o livro, e agora?.....	28
Alguns links quentes sobre histórias de usuários.....	29

# 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;
- 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 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 o 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.

Mas é totalmente 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 se você já tem uma noção sobre o assunto (ou já leu o livro), então você poderá navegar diretamente até determinado capítulo para relembrar conceitos e 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 que você faça certo 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 negócios 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 negócios.

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 estimá-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, CONVERSÃO, 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 **conversão**”.

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 possa 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.

# **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 dando 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**

Rpare 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**.

Rpare 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.

Rpare 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.

# PRIORIZANDO FUNCIONALIDADES

Temos uma lista de funcionalidades. E a grande dúvida é saber como priorizar as funcionalidades e por onde priorizar. Existe alguma regra?

É muito comum ouvir dos clientes que tanto faz a ordem. Eles querem tudo. Só que é uma questão de tempo para os clientes entenderem que não é bem assim. Uma das nossas tarefas ao trabalhar na adoção de metodologias ágeis é ajudar nossos clientes a entenderem que eles tem uma escolha diferente com relação a simplesmente “fazer o que precisa ser feito”! A opinião deles tem valor e ajuda no desenvolvimento do projeto.

Quando conseguimos começar a trabalhar com clientes e mostrar o quanto eles podem ganhar com um trabalho organizado de priorização, isto não tem preço. Ou melhor, sim, tem preço e é percebido na qualidade do trabalho gerado no final de cada ciclo. É diretamente ligado ao alinhamento que o time de projeto consegue ter com o negócio em questão.

O desenvolvimento de software é um processo. E este processo funciona muito com o ritmo das entregas e o valor que conseguimos perceber nas entrevistas e no processo de análise de negócio. O ponto pacífico é que queremos entregar funcionalidades que tenham valor para o negócio do cliente.

Agora, como identificar o que é valor? Ou qual funcionalidade que tem valor devemos priorizar? Se montarmos um quadrante analisando risco e valor, olhando alto e baixo risco, poderemos descobrir algumas coisas interessantes.

O que mais se quer são funcionalidades com alto valor e baixo risco. São atividades que podemos trabalhar “sem medo”, que tem a tecnologia relacionada bem resolvida, e que vão trazer benefício direto para os clientes.

Mas... focar primeiro nestas funcionalidades pode nos levar para uma situação de risco, justamente por não dar atenção aos itens que

possuem risco mais alto. Uma grande dica neste processo é trabalhar em tarefas que possuem risco alto, para justamente remover o risco técnico ou de negócio de uma determinada atividade.

Funcionalidades de alto risco e alto valor pode ser uma boa estratégia para garantir que o que precisa ser feito de verdade no projeto e que possui grandes chances de falha, está sendo analisado primeiro. Queremos eliminar o risco.

E olha que interessante. Olhando o que tem risco alto e valor alto, e depois olhando o que possui risco baixo e valor alto, estamos garantindo que estamos fazendo o que o projeto “pede”. Com estas duas categorias estamos fazendo o que importa.

Agora, acontece de termos que focar em atividades de risco alto e baixo valor. Não é indicado mas pode aparecer em alguma atividade regulatória, para adequar a empresa em alguma lei, por exemplo. É o tipo de atividade que vai ser feita somente em virtude do impacto financeiro que ela causa, caso não seja realizada. E aqui a importância de um bom processo de conversa com os clientes. Muitas vezes estas atividades acabam ficando esquecidas e se tornam urgências dentro do time. Então é muito interessante identificar se o projeto que está sendo desenvolvido tem este contexto de regulação e ficar atento as mudanças, para que possamos responder quando for necessário.

E as de baixo valor e baixo risco? Faça por último, se for realmente necessário. Normalmente serão algumas telas de apoio para cadastro. De todo modo, é o caso de identificar se o projeto ainda tem orçamento e pensar em fazer novas funcionalidades ou até em começar um novo projeto.

## 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.

# TERMINEI O LIVRO, E AGORA?

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

Isto provavelmente significa que 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 nós vamos deixar algumas sugestões sobre o que você pode fazer a partir de agora:

Se você gostou do livro então nos ajude a divulgá-lo e compartilhe o link <http://historiasdeusuario.com.br> no Facebook, Twitter, LinkedIn, Tumbler e etc. Junte-se a nós e vamos tornar as especificações de requisitos de software mais amigáveis, objetivas e divertidas.

Mas se você não gostou do livro então nos mande email dizendo o motivo, nós prometemos não ficar magoados. Pode escrever para [contato@wildtech.com.br](mailto:contato@wildtech.com.br)

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

Nós estamos no twitter, nos siga e manda um alô: [@rafaelhelm / @dwildt](https://twitter.com/rafaelhelm)

# ALGUNS LINKS QUENTES SOBRE HISTÓRIAS DE USUÁRIOS

## **Artigo:**

William Wake - Invest in Good User Stories and Smart Tasks

<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

## **Artigo:**

Ron Jeffries - How should stories be written

<http://xprogramming.com/blog/how-should-user-stories-be-written/>

## **Livro:**

Mike Cohn - User Stories Applied

<http://amzn.to/1hB6GAK>

## **Livro:**

Mike Cohn - Agile Estimating and Planning

<http://amzn.to/1heiAjA>

Vá e conte as  
histórias dos  
seus usuários.

:)