USER STORY

User Story: Representa uma potencial entrega de funcionalidade para o cliente final. Recomenda-se a quebra de uma iniciativa em entregas menores, para dividir o risco e estimular a experimentação das funcionalidades o mais cedo possível. Recomendamos que uma histórias sejam:

- **I.N.V.E.S.T.**:
- **I** = Independente
- **N** = Negociável
- **V** = Com valor (valuable)
- **E** = Estimável
- **S** = Pequena (small)
- **T** = Testável

Recomenda-se o uso no formato BDD (Behavior Driven Development) Uma História é composta por uma ou mais Sub-Tarefas.

USER STORY	
COMO UM EU QUERO PARA QUE	CONTEXTO
CRITÉRIOS DE ACEITE	
VALOR ENTREGUE	VALOR PERCEBIDO

História de usuário é um jeito bastante simples e eficiente de representar um requisito de software ou descrever uma funcionalidade, pois foca na linguagem de negócio e não se preocupa em detalhar a parte técnica, muito menos em tentar descrever o “como" uma funcionalidade será implementada.

Sua principal característica é representar um requisito de software sob a perspectiva do usuário final, incluindo os diferentes tipos de usuários e focando no resultado que este usuário está tentando atingir, bem como no valor que aquilo trará àquele usuário.

APLICAÇÃO Histórias de usuário são mais indicadas quando o usuário ou tipo de usuário é o foco central da ação e geralmente ocorrem em situações onde deseja-se criar algo novo, novos requisitos ou novas funcionalidades de software.

As histórias de usuário são excelentes alternativas para facilitar a conversa entre o *Product Owne*r (Dono do produto) e o time, permitindo que o foco dessa conversa seja no resultado e no valor que uma determinada funcionalidade trará ao usuário final, ao invés de no 'como' uma funcionalidade será implementada tecnicamente.

IMPROVEMENT STORY

Improvement Stories**,** ou “histórias para melhoria”, são alternativas bastante interessantes às histórias de usuário e funcionam muito bem em situações em que deseja-se melhorar uma funcionalidade existente, desde que essa melhoria seja relativamente pequena e bastante óbvia, pois focam na solução daquela situação.

IMPROVEMENT STORY	
NÓS TEMOS NÓS QUEREMOS TER PARA QUE	CONTEXTO
CRITÉRIOS DE ACEITE	
VALOR ENTREGUE	VALOR PERCEBIDO

APLICAÇÃO As *Improvement Stories* são recomendas em produtos de software mais “maduros”, bem depois da entrega do MVP (*Minimum Viable Product* - Produto Minimamente Viável), geralmente em suporte ou BAU (*Business as Usual*).

As *Improvement Stories* são rápidas de se criar e bem fáceis de ser entendidas, o que ajuda a facilitar o entendimento do time, e, ao mesmo tempo, evita que o *Product Owner* (Dono do Produto) se esforce escrevendo *User Stories* mais completas, que não adicionariam muito valor nestes casos específicos, já que não há muita margem para negociação, nestes casos.

DICA EXTRA ***Improvement Stories também podem ser utilizadas em situações que não se têm certeza se algo é bug, evitando todo aquela discussão se “é bug ou não”. Afinal é mais importante que o time resolva aquela situação ou implemente aquela funcionalidade o mais rápido possível, do que perca tempo discutindo ou tentando achar um culpado.***

JOB STORY

Pois bem, se você estiver em uma situação em que você se sinta meio que 'obrigado' a identificar um tipo de usuário só porque faz parte do template, ou a definir um usuário 'genérico' só pra não deixar o "*Como um….*" sem preencher, talvez você deva tentar substituir essa história de usuário por *Job Story*.

As ***Job Stories*** são uma alternativa interessante às *User Stories*, principalmente quando se precisa focar mais na **situação** que será o gatilho daquela ação, do que no **tipo de usuário** que quer algo. Isso, obviamente, pode variar de item pra item, a depender da necessidade, podendo resultar em um *product backlog* misto, com histórias de usuário **E** *Job Stories* no mesmo backlog, sem problemas.

<table>
<tr><th colspan="2">JOB STORY</th></tr>
<tr><td>QUANDO

EU QUERO

PARA QUE EU POSSA</td><td rowspan="2">CONTEXTO</td></tr>
<tr><td>CRITÉRIOS DE ACEITE</td></tr>
<tr><td>VALOR ENTREGUE</td><td>VALOR PERCEBIDO</td></tr>
</table>

APLICAÇÃO *Job Stories* são indicadas em situações em que é difícil conseguir identificar um usuário específico, ou ainda em cenários em que a situação é mais relevante do que o usuário que irá executar aquela ação.

As *Job Stories* são uma alternativa bastante interessante em empresas que trabalham no modelo cascata (*waterfall*), e que queiram dar o primeiro passo para iniciar com a transformação digital, mas que ainda não possuem processos maduros o suficiente para se desfazer do documento de requisitos tradicional, por exemplo.

Por fim, *Job Stories* podem também ser eficazes em projetos curtos, onde o tempo e o esforço para treinar e tentar convencer os clientes/times dos benefícios das histórias de usuário e critérios de aceitação são tão grandes, comparado ao tamanho e duração do projeto, que não justifica o esforço.

TECHNICAL STORY

TÉCNICA -, *focando no objetivo e no resultado/valor esperado"*. Porém, como o nome sugere, as *Technical Stories* são mais focadas na parte técnica da implementação e são usadas principalmente em "SPIKES", embora também possam ser usadas para identificação de requisitos não funcionais.

TECHNICAL STORY	
PARA QUE PRECISAMOS	RESULTADO ESPERADO
CRITÉRIOS DE ACEITE	
SISTEMA / FUNCIONALIDADE	MÉTRICA

De maneira resumida, SPIKE - originado no XP (*Extreme Programming*) - trata-se de um pequeno experimento, semelhante a uma investigação, que permite ao time aprender mais sobre uma determinada área da aplicação, geralmente o suficiente para poder realizar a estimativa e minimizar os riscos da implementação de uma funcionalidade.

APLICAÇÃO *Technical Stories* são recomendadas quando se deseja descrever requisitos técnicos, não funcionais e/ou SPIKES.

Uma *Technical Story* nunca deve estar sozinha no *backlog* do produto. Isso porque, não é recomendado implementar uma tarefa técnica, SPIKE, *task* ou *chore* simplesmente porque alguém interno disse que algo precisa ser feito (e.g. desenvolvedor, arquiteto, gerente, etc).

Geralmente, elas surgem a partir da necessidade de se entregar algo maior ao cliente, e por este motivo é que devem sempre estar associadas aos requisitos/desejos originais, como a uma *User Story* por exemplo. Essa associação cria uma relação hierárquica entre a *User Story* original e a *Technical Story*, como se fosse uma relação de dependência.

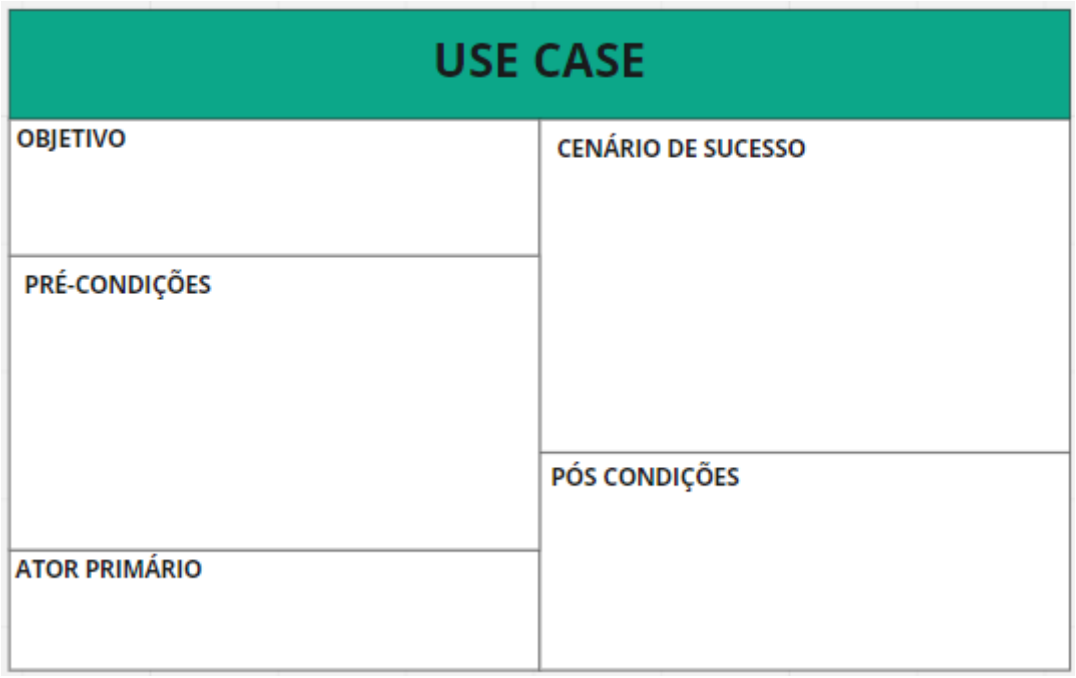
A relação de dependência entre uma história de usuário e uma história técnica, normalmente ocorre a partir da "quebra" da história de usuário original ("pai"), em uma ou mais histórias técnicas ("filhas"). Essa relação pode ocorrer basicamente de duas maneiras:

1. A primeira maneira consiste em “quebrar” a história de usuário original (“pai”) em uma ou mais histórias “filhas” usando tarefas técnicas (*chores* ou *tasks*), conforme apresentado na **Figura 1**.

2. A segunda maneira (e a mais usada) consiste em criar um SPIKE usando *Technical Story* (“filha”), a partir da *User Story* original (“pai”), conforme apresentado na **Figura 2.**

USE CASE

Essa relação de dependência impede o time de concluir a história original enquanto a história “filha” não esteja concluída. Por este motivo, recomenda-se separar as iterações, trazendo a *Technical Story* (SPIKE) em uma *Sprint* e a *User Story* original para uma *Sprint* subsequente. Desta forma, o time terá tempo suficiente de completar a SPIKE e remover a dependência, antes do início da implementação da história original.



Geralmente, eles estão relacionados a um processo bem definido e basicamente descrevem de uma maneira bem estruturada, as interações entre o sistema que será construído e seus atores, que pode ser uma pessoa, um sub-sistema ou um dispositivo físico que interage com esse sistema.

APLICAÇÃO Casos de uso podem ser usados em projetos onde seja necessário documentar todos os requisitos de uma vez e em bastante detalhe. Uma aplicação interessante de *Use Cases* fora do âmbito de requisitos é para a identificação dos principais cenários e fluxos para teste de carga (Load testing).

BUGS OU INCIDENTES

Existem variações desse modelo básico de caso de uso acima apresentado, que incluem coisas como: o nível e o escopo do caso de uso, assim como qualquer fluxo alternativo, qualquer extensão e pós-condição. Na prática, essa complexidade e detalhamento vão variar de caso a caso, dependendo da situação e necessidade.

<table>
<tr><th colspan="2">BUG</th></tr>
<tr><td colspan="2">TÍTULO</td></tr>
<tr><td rowspan="2">PASSO A PASSO</td><td>COMPORTAMENTO ATUAL</td></tr>
<tr><td>COMPORTAMENTO ESPERADO</td></tr>
<tr><td>URL:
Ambiente:
Versão do navegador:
Dispositivo:
Tipo de usuário:</td><td>Evidência:

Comentário:</td></tr>
</table>

Não é objetivo deste E-book entrar mérito do que é *bug*, defeito, falha ou erro. Porém, discutiremos um modelo genérico o bastante para poder ser utilizado em praticamente todas essas situações, e que tem se demonstrado bastante eficaz na identificação e reprodução de problemas deste gênero.

APLICAÇÃO Este *template* pode ser utilizado para identificação, reprodução e resolução de *bugs*, defeitos, falhas ou erros de software.

Este modelo é bastante simples e muito poderoso, pois reune o mínimo de informação necessária para que seja possível realizar uma investigação eficaz, evitando que o time perca tempo buscando/perguntando por informações faltantes.

DICA EXTRA ***Caso queira aumentar ainda mais a chance do seu time resolver um bug, defeito, falha ou erro sem precisar perder tempo procurando por informação faltante, adicione ao modelo os items complementares abaixo apresentados.***

OBS.: Este conteúdo é baseado nas técnicas apresentadas pelo Leandro Bodo em seu ebook "5 Maneiras de descrever requisitos de software".