



INSTITUTO DE GESTÃO E TECNOLOGIA
DA INFORMAÇÃO

Habilitando o Backlog do Produto

Rafael Mazaro

2022

Habilitando o Backlog do Produto

Rafael Mazaro

© Copyright do Instituto de Gestão e Tecnologia da Informação.

Todos os direitos reservados.

Sumário

Capítulo 1.	Dual Track Agile	Erro! Indicador não definido.
1.1.	Como funciona a esteira Dual Track	Erro! Indicador não definido.
1.2.	Expandindo as árvores de oportunidade	Erro! Indicador não definido.
Capítulo 2.	Temas, Épicos e Histórias	Erro! Indicador não definido.
2.1.	Introdução	Erro! Indicador não definido.
2.2.	Épico x Histórias ágeis	Erro! Indicador não definido.
2.3.	Épico x Iniciativas ágeis	Erro! Indicador não definido.
2.4.	Épico x Temas	Erro! Indicador não definido.
Capítulo 3.	A importância do Business Owner no time de produtos	Erro! Indicador não definido.
3.1.	Quem é o Business Owner	Erro! Indicador não definido.
3.2.	Apoio na construção do produto	Erro! Indicador não definido.
3.3.	Disponibilidade no ciclo da entrega	Erro! Indicador não definido.
Capítulo 4.	MVP com técnicas de User Story Mapping	Erro! Indicador não definido.
4.1.	Introdução MVP com Story Mapping	Erro! Indicador não definido.
4.2.	Objetivos do User Story Mapping	Erro! Indicador não definido.
4.3.	Mapeando com os stakeholders	Erro! Indicador não definido.
4.4.	Organizando por épicos	Erro! Indicador não definido.
4.5.	Composição das histórias	Erro! Indicador não definido.
4.6.	Priorizando o backlog (MVP)	Erro! Indicador não definido.
Capítulo 5.	Técnica INVEST	Erro! Indicador não definido.
5.1.	Introdução ao conceito INVEST	Erro! Indicador não definido.

5.2. INDEPENDENT (Independente).....	Erro! Indicador não definido.
5.3. NEGOTIABLE (Negociável).....	Erro! Indicador não definido.
5.4. VALUABLE (Valiosa).....	Erro! Indicador não definido.
5.5. ESTIMABLE (Estimável)	Erro! Indicador não definido.
5.6. SMALL (Pequena)	Erro! Indicador não definido.
5.7. TESTABLE (Testável)	Erro! Indicador não definido.
5.8. Melhoria contínua.....	Erro! Indicador não definido.
Capítulo 6. Fundamentos de BDD e TDD	Erro! Indicador não definido.
6.1. Introdução BDD e TDD	Erro! Indicador não definido.
6.2. BDD (Behavior Driven Development)	Erro! Indicador não definido.
6.3. TDD (Test Driven Development)	Erro! Indicador não definido.
6.3.1. Novo Teste.....	Erro! Indicador não definido.
6.3.2. Teste falhando	Erro! Indicador não definido.
6.3.3. Criar funcionalidade	Erro! Indicador não definido.
6.3.4. Teste	Erro! Indicador não definido.
6.3.5. Refatorar.....	Erro! Indicador não definido.
Capítulo 7. Criando histórias de usuário	Erro! Indicador não definido.
7.1. Introdução a escritas de histórias de usuário	Erro! Indicador não definido.
7.2. Técnicas para escritas de histórias	Erro! Indicador não definido.
7.3. O dia a dia das histórias de usuário	Erro! Indicador não definido.
7.4. Benefícios de histórias com BDD	Erro! Indicador não definido.
Capítulo 8. Autonomia do PO no time de produtos	Erro! Indicador não definido.
8.1. Chave para o sucesso	Erro! Indicador não definido.

Referências.....Erro! Indicador não definido.

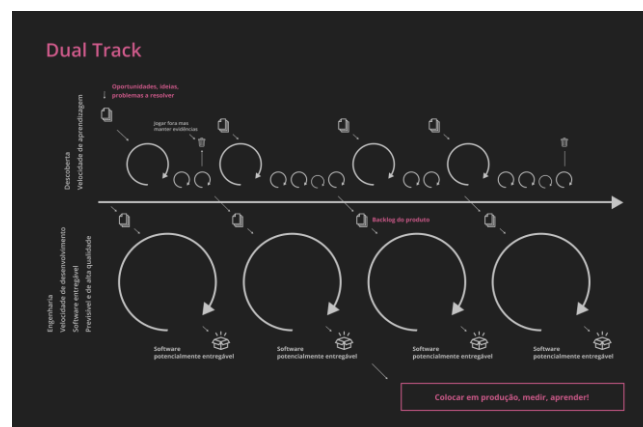
Capítulo 1. Dual Track Agile

O Dual Track Agile funciona como um alimentador de Sprints de entrega, através de todas as hipóteses que foram validadas na esteira de discovery. Por que isso é importante? Esse processo permite que a equipe melhore e preveja melhor o sucesso dos recursos do produto, maximizando, assim, a saliência e minimizando ainda mais o risco. O Discovery se torna o principal motor da estratégia de experiência do cliente. Ou seja, com o Dual Track Agile, sua equipe usa aprendizado contínuo para informar a entrega e melhoria contínua. É sua *máquina de inovação*.

1.1. Como funciona a esteira Dual Track

O backlog da esteira de Discovery é normalmente carregado com todas as ações (UX) — investigações e hipóteses a serem pesquisadas, novos clientes, tendências ou recursos de produtos a serem explorados, e falhas a serem estudadas. Em seguida, a Discovery Sprints usa o processo Design Thinking para explorar possibilidades e realizar pequenos experimentos para testar ideias. A equipe de Entrega, então, conecta essas ideias validadas em seu backlog e startu o ciclo contínuo.

Figura 1 – Esteira Dual Track Agile



Fonte: <https://brasil.uxdesign.cc> - Conhecendo o Dual Track.

Provavelmente, você já entendeu que o Discovery é o segredo de sucesso para que o produto esteja vários passos à frente de um produto tradicional, olhando constantemente para os movimentos de mercado, tendências culturais, tecnologia emergente, novas necessidades e desejos dos clientes. Sendo simplista, o processo Dual Track reforça que todos os insights dos clientes de nosso produto fossem colocados em contato constante com nossas equipes de entrega.

1.2. Expandindo as árvores de oportunidade

Outra razão para adotar o Dual Track Agil é expandir suas árvores de oportunidade. A tendência das equipes de entrega ágeis é ficar ocupadas agitando as primeiras ideias. É da natureza humana colocar em nossa correção chapéus e ir com ideias iniciais que poderiam ou iriam funcionar. Usamos nossa função lógica e crítica do cérebro para encontrar uma solução e ... feito! E afinal, se falhar, tudo bem, porque ainda estamos aprendendo. Todavia, como o Dual Track usa o processo de design thinking, suas equipes serão treinadas para usar o pensamento convergente e divergente para construir suas árvores de oportunidade em árvores reais, não em alguns galhos. Ele ajuda a evitar soluções muito rapidamente, oferecendo, à equipe, múltiplas opções de comparação. Como você pode saber qual direção tomar sem ter algo para comparar? Ter muitas opções é um problema melhor de tomada de decisão do que um punhado de falhas meses depois.

Com a esteira clara e implementada, o start da esteira de entrega de produtos fará muito sentido devido ao foco ser sempre no que descobrimos com nossos clientes.

Capítulo 2. Temas, Épicos e Histórias

2.1. Introdução

Digamos que você e sua equipe querem fazer algo ambicioso, como lançar um foguete no espaço. Para fazer isso, será necessário estruturar seu trabalho: dos maiores objetivos até os mais detalhados. Você desejará responder à mudança, reportar seu progresso e seguir um plano. Épicos, histórias, temas e iniciativas são, precisamente, as ferramentas que te ajudarão nesta jornada.

O que são histórias, épicos, iniciativas e temas?

- **Histórias**, também chamadas de "histórias de usuários", são requisitos pequenos ou solicitações escritas da perspectiva de um usuário final.
- **Épicos** são grandes partes de trabalho que podem ser divididas em várias tarefas menores (chamadas histórias).
- **Iniciativas** são coleções de épicos que têm um objetivo comum.
- **Temas** são grandes áreas de foco que abrangem a organização.



2.2. Épico X Histórias ágeis

De certo modo, histórias e épicos, no método ágil, são semelhantes a histórias e épicos em filmes e livros. Uma história é uma narrativa simples, uma série de histórias interdependentes e relacionadas que compõem um épico. O mesmo é verdadeiro para seu gerenciamento de trabalho, no qual a conclusão de histórias relacionadas leva à conclusão de um épico. As histórias contam o arco de trabalho completo, enquanto os épicos compartilham uma visão de alto nível do objetivo unificador.

Em uma equipe ágil, as histórias são algo que a equipe pode se comprometer a finalizar em um sprint de uma ou até quatro semanas. Muitas vezes, os desenvolvedores trabalham em várias histórias por mês. Os épicos, em contrapartida, são poucos e demoram mais para serem concluídos. Normalmente, as equipes trabalham em dois ou três épicos por trimestre.

Se sua empresa está lançando foguetes no espaço e deseja aprimorar o serviço de streaming para seus lançamentos, é necessário estruturar suas histórias como as histórias abaixo.

Exemplos de uma história ágil:

- Os usuários de iPhone precisam acessar uma visão vertical do feed ativo ao usar o aplicativo móvel.
- Os usuários de desktop precisam de um botão de "visualizar em tela cheia" na extremidade direita inferior do player de vídeo.
- Os usuários do Android devem ser vinculados à Apple Store.

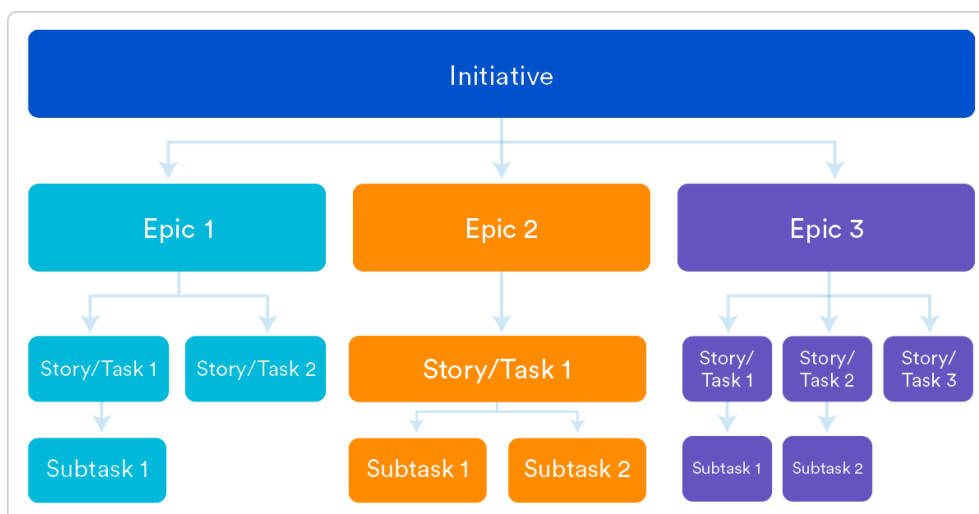
As histórias acima estão relacionadas. Todas podem ser consideradas tarefas individuais que levam à conclusão de uma parte maior do trabalho (épico).

Nesse caso, o épico pode ser **"Melhorar o serviço de streaming para lançamento no primeiro trimestre"**.

Organizar o trabalho em histórias e épicos também ajuda você e sua equipe a se comunicarem efetivamente na organização. Se estava reportando o progresso de sua equipe ao Product Manager, estava falando em épicos. Se estava falando com um colega na sua equipe de desenvolvimento, estava falando no nível da história.

2.3. Épico X Iniciativas ágeis

Do mesmo modo que os épicos são compostos de histórias, as iniciativas são compostas de épicos. As iniciativas oferecem outro nível de organização, acima deles. Em muitos casos, uma iniciativa compila épicos de várias equipes para obter um objetivo maior e mais amplo do que qualquer um deles. Enquanto um épico é algo que você deve concluir em um mês ou trimestre, as iniciativas, normalmente, são concluídas em vários trimestres ou um ano.



Digamos que sua empresa de foguetes queira diminuir o custo por lançamento em 5% neste ano. É uma ótima adequação para uma iniciativa, pois um épico sozinho provavelmente não alcançaria essa meta. Nessa iniciativa, haveria um épico como "Diminuir o consumo de combustível na fase de lançamento em 1%", "Aumentar os lançamentos por trimestre de 3 para 4" e "Diminuir os termostatos de 71 a 69 graus #Dadmode".

Obs.: Iniciativas estão fortemente conectadas à estratégia de produtos, esse assunto será abordado no Bootcamp de Product Manager no módulo “Práticas para construção do produto digital”.

2.4. Épico X Temas

Em muitas organizações, os fundadores e a equipe de gerenciamento encorajam a busca por um destino de aspiração. Essas são as metas (às vezes muito antiquadas) anunciadas a cada ano ou trimestre, e os temas são como isso é monitorado.

As iniciativas têm um design estrutural. Elas abrigam os épicos, e a conclusão desses épicos ocasionará a conclusão da iniciativa. Os temas são uma ferramenta organizacional que permite que você rotule os itens da lista de pendências, os épicos e as iniciativas para entender quais trabalhos contribuem com quais metas organizacionais. Os temas devem inspirar a criação de épicos e iniciativas, mas não tenha um relacionamento rígido de "um para um" com eles. Um tema para uma empresa de lançamento de foguetes deve ser algo como "Segurança em primeiro lugar".

Capítulo 3. A importância do Business Owner no time de produtos

3.1. Quem é o Business Owner

O Business Owner tem um papel fundamental na estratégia, ele atua como parte interessada na construção de produtos digitais e sua responsabilidade é garantir a técnica e a governança das regras de negócio implementadas pelo time de desenvolvimento.

O termo Business Owner é o líder do negócio, aquele que entende do contexto do produto e suas regras. Esse profissional é fundamental na construção dos produtos.

3.2. Apoio na construção do produto

Imaginem a construção de um produto cujo foco é automatizar o processo de faturamento. Algumas regras de negócio desse produto sugerem que sejam implementadas funcionalidades que estejam em conformidade regulatória. Apesar do Product Owner trazer as regras que deverão ser implementadas, é fundamental que o Business Owner traga a sua visão e que garanta a implementação correta.

3.3. Disponibilidade no ciclo da entrega

Como vocês já devem ter entendido, o Business Owner é o especialista de negócios. Em uma empresa, esta figura é o analista do financeiro, o comprador, o vendedor etc. Na construção/evolução de produtos ágeis, o Business Owner compõe o time de produtos e não atua apenas na construção das regras, mas tira dúvidas do time de desenvolvedores e no processo de validação do produto.

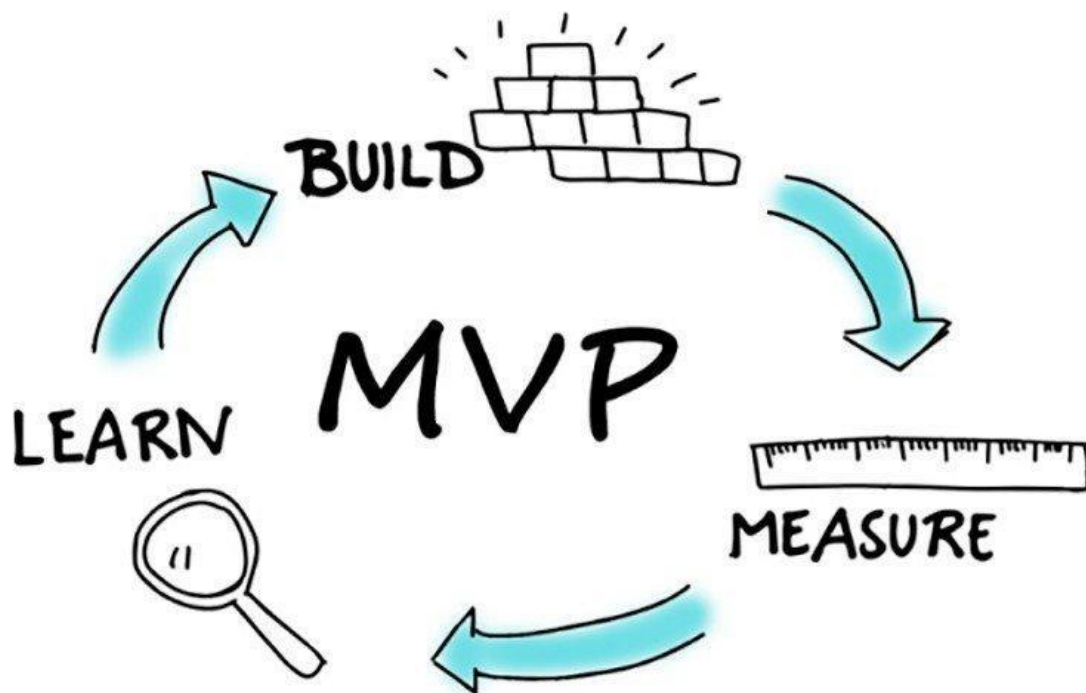


Capítulo 4. MVP com técnicas de User Story Mapping

4.1. Introdução MVP com Story Mapping

A partir deste capítulo, começaremos a falar sobre o MVP na esteira de delivery, onde deve ser considerado o que especificamente agrega valor para o cliente, mas que haja um benefício para a Cia. Para isso, existe uma técnica bastante eficiente: a utilização do User Story Mapping.

Esse é o grande foco deste curso, ajudar profissionais de produtos a construir soluções em que seus clientes amam e que traga um retorno financeiro para a empresa.



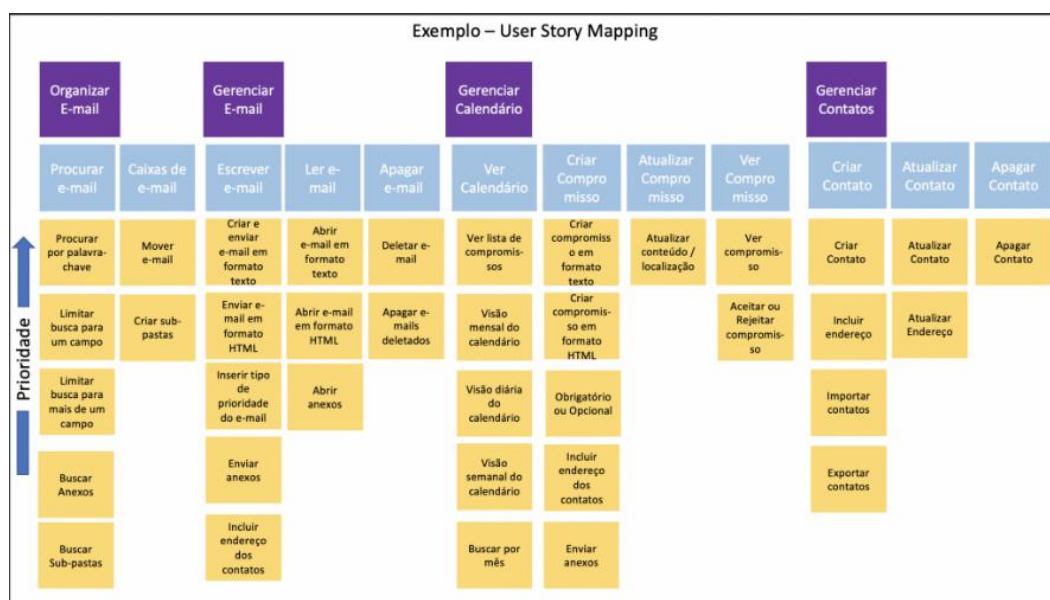
4.2. Objetivos do User Story Mapping

O mapeamento de histórias tem um objetivo simples: Manter os envolvidos na construção do produto, focados no que ativará valor sobre a experiência para o cliente e o retorno sobre o investimento para a Cia.

A técnica do User Story Mapping ajuda na facilitação dessa transmissão de informações para uma equipe. Quando as histórias são escritas e lidas pelo time de desenvolvimento, podem gerar dúvidas e interpretações diferentes.

A partir do momento que há uma visualização de todas as atividades, por meio de uma jornada, onde você pode percorrer toda a estrutura do sistema, dando a visibilidade de onde se quer chegar, isso facilita toda a parte de comunicação entre as partes.

Outro ponto importante são as prioridades que ficam muito mais claras, com as histórias sendo priorizadas com base na estrutura que o mapeamento de atividades foi montado, não ficando apenas uma única lista de histórias.



4.3. Mapeando com os stakeholders

O envolvimento do time, como desenvolvedores e designers, além da inclusão de tomadores de decisão de outras áreas, será muito útil para a discussão e entendimento sobre a visão e objetivo do produto, além de apoiar em pontos que possam necessitar de um aprofundamento maior, seja do ponto de vista técnico ou de negócio.

4.4. Organizando por épicos

No processo de Discovery, passamos a conhecer os épicos que compõem o produto, organizar esses épicos em jornada é o primeiro passo do User Story Mapping.



4.5. Composição das histórias

Com base na criação dos épicos, crie todas as histórias. Neste momento, não pense ainda em MVP, mas em algo voltado ao desejável, e também não precisa entrar no detalhe, pois ela servirá mais como apoio e passará por refinamento posterior. Além disso, organize as histórias em uma sequência lógica e que seja possível todos entenderem a sequência.

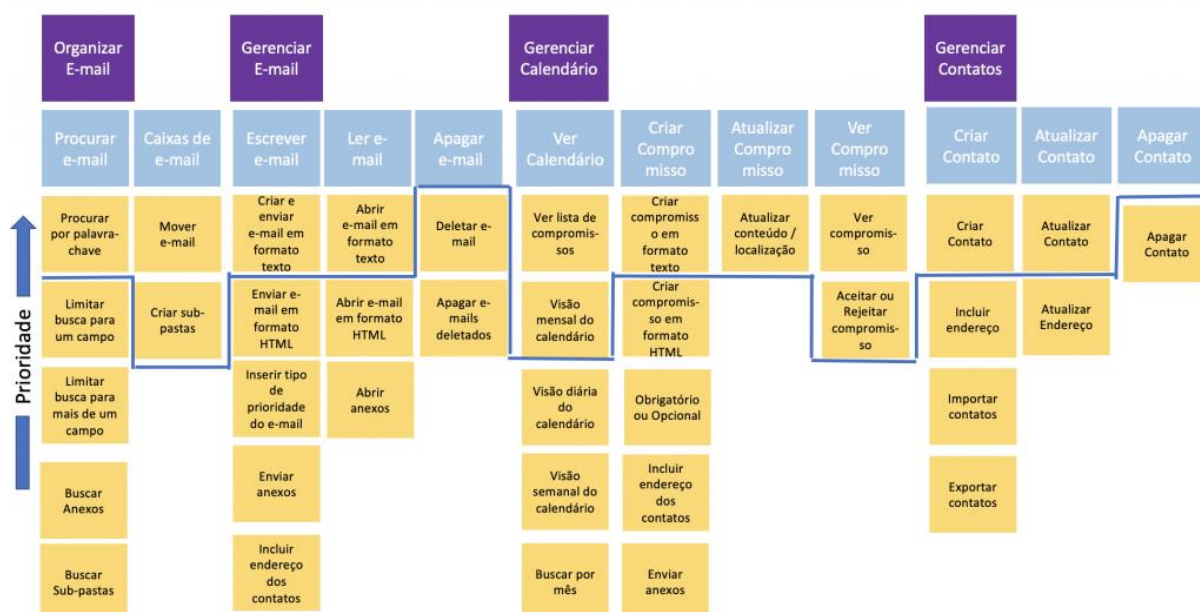
Procurar por palavra-chave	Mover e-mail	Criar e enviar e-mail em formato texto	Abrir e-mail em formato texto	Deletar e-mail	Ver lista de compromissos	Criar compromisso em formato texto	Atualizar conteúdo / localização	Ver compromisso	Criar Contato	Atualizar Contato	Apagar Contato
Limitar busca para um campo	Criar sub-pastas	Enviar e-mail em formato HTML	Abrir e-mail em formato HTML	Apagar e-mails deletados	Visão mensal do calendário	Criar compromisso em formato HTML		Aceitar ou Rejeitar compromisso	Incluir endereço	Atualizar Endereço	
Limitar busca para mais de um campo		Inserir tipo de prioridade do e-mail	Abrir anexos		Visão diária do calendário	Obrigatório ou Opcional			Importar contatos		
Buscar Anexos		Enviar anexos			Visão semanal do calendário	Incluir endereço dos contatos			Exportar contatos		
Buscar Sub-pastas		Incluir endereço dos contatos			Buscar por mês	Enviar anexos					

4.6. Priorizando o backlog (MVP)

Após a criação das histórias, chegou a hora de realizar a priorização delas. Como o mapa pode ter muitas histórias que foram incluídas, nada melhor que criar “cortes” para chegarmos à criação do MVP. Esses cortes são chamados de incrementos.

Pense um pouco sobre as tarefas que você quer ajudar o seu usuário a cumprir com o seu produto. Vá criando cortes incrementais (linha tracejada em azul no exemplo) e incluindo as histórias que menos impactam o usuário a cumprir com a sua tarefa/objetivo.

Após realizar alguns cortes, é possível que você chegue a um corte que seja definido como o MVP da solução que está construindo.



Capítulo 5. Técnica INVEST

5.1. Introdução ao conceito INVEST

Uma das principais falhas na construção das funcionalidades de um produto digital está relacionada à qualidade em que os requisitos são construídos. Na hora de escrever os critérios de aceite, Product Owners se empolgam e geram conteúdos abrangentes que dificultam a clareza do time, consequentemente a estimativa e validação dessas histórias ficam prejudicadas.

Então, para ajudar na construção de boas histórias, existem uma técnica chamada INVEST (Independente, Negociável, Valor, Estimável, Pequena e Testável).

5.2. INDEPENDENT (Independente)

Para que as suas User Stories sejam independents, elas não podem depender de nenhuma outra para ser implementada. Se não respeitarmos esse critério, corremos o risco de carregar uma série de outras histórias de usuário à medida em que o Product Owner prioriza uma determinada história.

Agora, sejamos francos, é fácil atingirmos esse critério? Acredito que não. No entanto, vale a pena colocar alguma energia para chegar o mais próximo possível desta tal independência.

5.3. NEGOTIABLE (Negociável)

Quando escrevemos User Stories, partimos do princípio que nem todos os detalhes não estarão escritos. E mais importante do que isso é manter o foco na necessidade.

Outro aspecto muito relevante é que as histórias podem ser escritas a quatro mãos.

Em algum momento, criamos uma ideia equivocada de que a escrita é responsabilidade exclusiva do Product Owner, porém, é um desperdício não aproveitarmos todo o capital intelectual do próprio time para também escrever e apoiar a escrita de histórias, e é por isso que a negociação é importante.

5.4. VALUABLE (Valiosa)

Todas as User Stories têm que gerar valor para o usuário! Sim! Para o usuário. Afinal de contas, estamos falando de histórias de usuários.

De todas as letras que compõem este acrônimo, este talvez seja o mais importante. Esqueça aquela ideia de escrever histórias de front-end, banco de dados e coisas do gênero. Podemos tratar isto através da escrita de outros artefatos.

5.5. ESTIMABLE (Estimável)

Os times têm que ser capazes de estimar as histórias e, para isso, elas precisam ser claras e pequenas.

Então, o foco não é exatamente a sua estimativa, mas sim o quanto somos capazes de fazer isso ao ler e discutir a história.

5.6. SMALL (Pequena)

Ser PEQUENA se conecta diretamente com ESTIMÁVEL que comentamos anteriormente.

Partindo do princípio de quanto menor o lote de trabalho, melhor a sua previsibilidade, se a sua história obedecer a esse critério, o time pode ter mais facilidade em estimar.

Pensar SMALL te orienta a uma escrita mais objetiva e concisa.

Aqui, há um benefício direto para o Product Owner. À medida em que suas User Stories ficam prontas, ele já pode degustar, digamos assim, daquela pequena implementação.

Então, se a história é SMALL, a tendência é que ela chegue mais rápido nas mãos dele enquanto o time se concentra em outras pequenas partes.

5.7. TESTABLE (Testável)

Se a sua história pode ser testada em toda a sua plenitude, então ela atende a esse requisito.

As condições necessárias para fazer a sua validação normalmente estão indicadas nos critérios de aceitação.

Follow the INVEST
guidelines for good
user stories!



one | 80
SERVICES



5.8. Melhoria contínua

Honestamente, não podemos pensar o acrônimo *INVEST* como um check list onde a sua história só pode ir para uma “planning” do time quando atende a 100% desses critérios. Mas, para dar um empurrãozinho na velocidade em que as suas User Stories melhoram, eu adicionaria a técnica INVEST no seu DoR (definition of ready).

Fazendo isso, você terá que refletir a escrita das histórias antes de colocá-las na arena.

Mais importante do que a técnica em si, é você trazer um dos pilares do Mindset Ágil aqui neste contexto: Melhoria contínua. Por isso que pensar em um trabalho evolutivo é mais interessante do que em um check list.

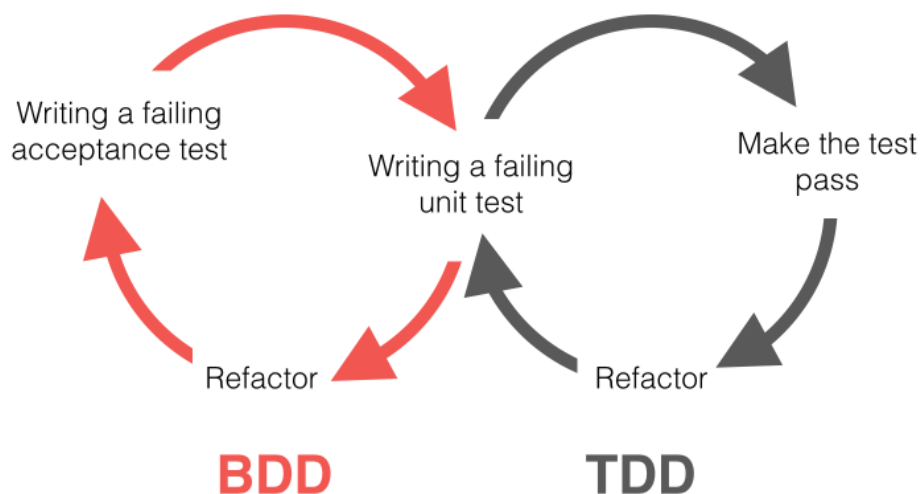
À medida em que você vai aprendendo a respeito do seu produto e alimentando o seu Backlog, a tendência é que a qualidade na escrita das User Stories melhore ao longo do tempo.

Capítulo 6. Fundamentos de BDD e TDD

6.1. Introdução BDD e TDD

Com a evolução das ferramentas de desenvolvimento para produtos digitais, foram criadas novas formas para escrita e desenvolvimento de histórias de usuário. O BDD (Behavior Driven Development) é uma técnica que utiliza, como premissa, o comportamento do usuário.

O TDD (Test Driven Development) é uma técnica de desenvolvimento de software que utiliza o conceito de verificar e validar o código constantemente, permitindo a implementação de automações de testes.



6.2. BDD (Behavior Driven Development)

Descrevendo o comportamento

Seus cenários devem descrever o comportamento pretendido do sistema, não a implementação. Em outras palavras, deve descrever o *quê*, não *como*.

Por exemplo, para um cenário de autenticação, você deve escrever:

```
When "Bob" logs in
```

Em vez de:

```
Given I visit "/login"  
When I enter "Bob" in the "user name" field  
    And I enter "tester" in the "password" field  
    And I press the "login" button  
Then I should see the "welcome" page
```

O primeiro exemplo, **quando “Bob” efetua login**, é um *requisito funcional*. O segundo exemplo, muito mais longo, é uma *referência procedimental*. Requisitos funcionais são recursos, mas os procedimentos pertencem aos detalhes de implementação.

Dessa forma, quando a implementação de um recurso for alterada, você só precisará alterar as etapas do processo nos bastidores. O comportamento não precisa mudar apenas porque a implementação muda. Na verdade, uma boa pergunta a se fazer ao escrever uma cláusula de recurso é: “Esse texto precisará ser alterado se a implementação mudar?”.

Se a resposta for “Sim”, você deve retrabalhar evitando detalhes específicos de implementação. Como benefício colateral, consequentemente seus cenários serão muito mais curtos e muito mais fáceis de seguir e entender.

Uma maneira de tornar os cenários mais fáceis de manter, e menos frágeis, é usar um estilo declarativo. O estilo declarativo descreve o comportamento do aplicativo, em vez dos detalhes de implementação. Cenários declarativos podem ser lidos melhor como “documentação viva”. Um estilo declarativo ajuda você a se concentrar no valor que o cliente está obtendo, em vez dos pressionamentos de tecla que eles usarão.

Os testes imperativos comunicam detalhes e, em alguns contextos, esse estilo de teste é apropriado. Por outro lado, por estarem tão intimamente ligados à mecânica da IU atual, costumam exigir mais trabalho de manutenção. Sempre que a implementação muda, os testes também precisam ser atualizados.

Aqui está um exemplo de um recurso em um estilo imperativo:

```
Feature: Subscribers see different sets of stock images based on their
subscription level
```

```
Scenario: Free subscribers see only the free articles
```

```
    Given users with a free subscription can access "FreeArticle1" but
    not "PaidArticle1"
```

```
        When I type "freeFrieda@example.com" in the email field
```

```
        And I type "validPassword123" in the password field
```

```
        And I press the "Submit" button
```

```
        Then I see "FreeArticle1" on the home page
```

```
        And I do not see "PaidArticle1" on the home page
```

```
Scenario: Subscriber with a paid subscription can access
"FreeArticle1" and "PaidArticle1"
```

```
    Given I am on the login page
```

```
        When I type "paidPattya@example.com" in the email field
```

```
        And I type "validPassword123" in the password field
```

```
And I press the "Submit" button
```

```
Then I see "FreeArticle1" and "PaidArticle1" on the home page
```

Cada etapa é uma instrução precisa. As entradas e os resultados esperados são especificados exatamente. Mas é fácil imaginar mudanças no aplicativo que exigiriam a alteração desses testes. As opções disponíveis para assinaturas gratuitas e pagas podem mudar. Até os meios de login podem mudar. E se, no futuro, os usuários fizerem login com uma interface de voz ou impressão digital?

Um estilo mais declarativo oculta os detalhes de como os recursos do aplicativo são implementados.

```
Feature: Subscribers see different sets of stock images based on their subscription level
```

```
Scenario: Free subscribers see only the free articles
```

```
Given Free Frieda has a free subscription
```

```
When Free Frieda logs in with her valid credentials
```

```
Then she sees a Free article on the home page
```

```
Scenario: Subscriber with a paid subscription can access both free and paid articles
```

```
Given Paid Patty has a basic-level paid subscription
```

```
When Paid Patty logs in with her valid credentials
```

```
Then she sees a Free article and a Paid article on the home page
```

Com um estilo declarativo, cada etapa comunica uma ideia, mas os valores exatos não são especificados. São especificados nas definições da etapa (o código de automação que interage com o sistema) os detalhes de *como* o usuário interage com o

sistema, quais artigos específicos são gratuitos ou pagos e o nível de assinatura de diferentes usuários de teste, por exemplo.

Os pacotes de assinatura podem mudar no futuro e a empresa pode alterar o conteúdo disponível para assinantes em planos gratuitos e pagos, sem ter que alterar este e outros cenários que usam as mesmas definições de etapa. Se outro nível de assinatura for adicionado posteriormente, é fácil adicionar um cenário para isso. Ao evitar termos como “clicar em um botão” que sugere implementação, o cenário é mais resiliente aos detalhes de implementação da UI (User Interface). A intenção permanece a mesma, mesmo que a implementação seja alterada posteriormente.

6.3. TDD (Test Driven Development)

A princípio, pode parecer estranho escrever um teste sem ao menos ter a respectiva função pronta. Mas a ideia por trás disso é justamente facilitar a criação do código e evitar a inserção de erros e bugs no sistema.

Afinal, o fato do teste já estar implementado faz com que a pessoa desenvolvedora tenha mais facilidade para entender o que supostamente o código deve fazer e possa implementar a funcionalidade já pensando em como o teste vai passar. Por consequência, o processo se torna mais rápido, já que são criados códigos mais simples.

Fora isso, utilizando o TDD, é possível garantir que cada pequena parte do sistema tenha um teste que atesta o seu funcionamento, o que melhora a qualidade final do software.

Como funciona o TDD e seu ciclo

Como foi dito, o TDD se inicia com a criação de um teste para cada nova funcionalidade da aplicação. Mas isso não é tudo, o ciclo completo envolve cinco etapas diferentes que explicaremos logo abaixo.

6.3.1. Novo teste

Ao identificar uma nova funcionalidade para o sistema, a pessoa desenvolvedora deve iniciar a implementação de um teste específico para ela. Nesta etapa, é muito importante entender exatamente qual é a finalidade da função, seus requisitos, entradas e saídas, pois só assim você conseguirá criar o teste corretamente.

6.3.2. Teste falhando

Neste momento, o teste já está pronto. No entanto, uma vez que o código da função ainda não foi implementado, ao ser executado, o teste obviamente vai falhar. Esse resultado é totalmente esperado e a execução é feita apenas para demonstrar que algo novo foi acrescentado ao conjunto de testes do sistema.

6.3.3. Criar funcionalidade

Após a execução e falha do teste, finalmente é chegado o momento de escrever o código da nova funcionalidade. Nesse ponto, a única preocupação que deve ter em mente é fazer o teste passar. Ou seja, você pode criar o código da sua maneira, sem se prender às boas práticas e design patterns, o importante é que o teste passe sem que outros quebrem com a adição da nova funcionalidade.

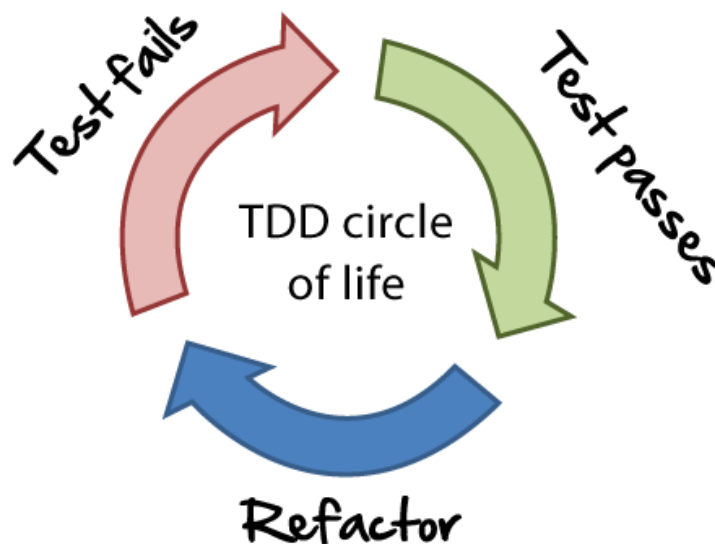
6.3.4. Teste

Com a funcionalidade pronta, o teste é executado novamente. Dessa vez, no entanto, é esperado que ele passe. Porém, mesmo que o código esteja funcionando, é

preciso lembrar que ele foi feito da forma mais simples possível e sem considerar as boas práticas de desenvolvimento. Isso nos leva ao próximo passo.

6.3.5. Refatorar

As boas práticas de desenvolvimento nos ensinam que um bom código deve ser simples, claro, coeso e menos acoplado possível. A refatoração é usada exatamente para esse objetivo. Nesta etapa, você deve melhorar o seu código, extraindo classes e interfaces, reduzindo o acoplamento, retirando as duplicidades e fazendo qualquer alteração que traga alguma otimização, desde que elas não insiram erros no código.



Capítulo 7. Criando histórias de usuário

7.1. Introdução a escritas de histórias de usuário

Uma história do usuário é uma explicação informal e geral sobre um recurso de software escrita a partir da perspectiva do usuário final. Seu objetivo é articular como um recurso de software pode gerar valor para o cliente. É tentador pensar que histórias de usuário são, de grosso modo, requisitos de sistema do software. Mas não são.

Um componente-chave do desenvolvimento de software ágil é colocar as pessoas em primeiro lugar, e as histórias de usuário colocam os usuários finais reais sob os holofotes. Essas histórias usam linguagem não técnica para dar contexto à equipe de desenvolvimento e suas iniciativas. Depois de ler uma história de usuário, a equipe sabe porque está desenvolvendo o que está desenvolvendo e qual o valor que isso cria.

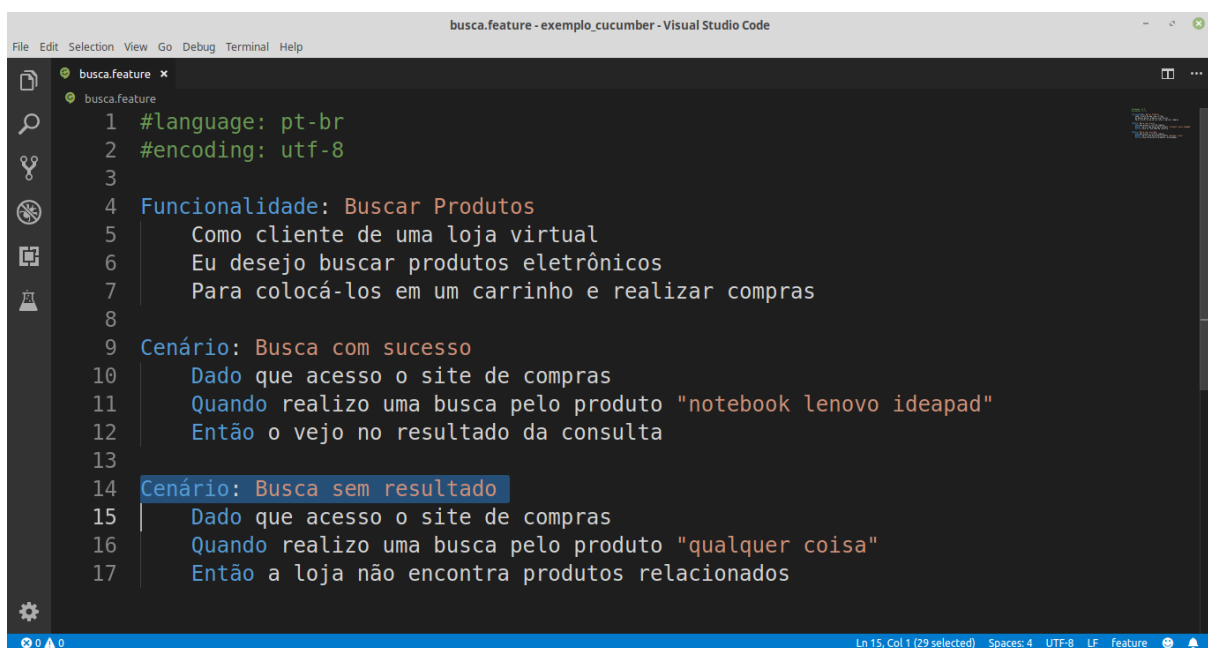
Histórias de usuários são um dos componentes principais de um programa ágil. Elas possibilitam uma estrutura centrada no usuário para o trabalho diário, o que impulsiona a colaboração, a criatividade e um produto melhor em geral.

7.2. Técnicas para escritas de histórias

Uma história de usuário utiliza cenários que capturam os critérios de aceitação da história. Usando essa técnica, você garante que os requisitos de desenvolvimento, bem como a perspectiva de negócios e as expectativas de experiência do usuário final, sejam levados em consideração. E como os critérios de aceitação são escritos em uma linguagem clara e concreta que todos podem entender, você pode avaliar com mais precisão os candidatos a lançamento em relação aos critérios de aceitação. Simplificando, há menos espaço para mal-entendidos.

Para adicionar um cenário de aceitação a uma história, usamos Gherkin, uma linguagem legível para definir casos de teste. Usando o formato simples Dado-Quando-Então, você pode capturar em primeira pessoa a partir da perspectiva de cada um dos membros da equipe, assim:

- **Dado** é a configuração; por exemplo, “DADA que o fluxo de trabalho exista”;
- **Quando** é a ação ou gatilho; por exemplo, “QUANDO eu solicitar uma API GET”;
- **Então** é a afirmação; por exemplo, “ENTÃO o fluxo de trabalho deve executar seus trabalhos”.



```

1 #language: pt-br
2 #encoding: utf-8
3
4 Funcionalidade: Buscar Produtos
5     Como cliente de uma loja virtual
6     Eu desejo buscar produtos eletrônicos
7     Para colocá-los em um carrinho e realizar compras
8
9 Cenário: Busca com sucesso
10    Dado que acesso o site de compras
11    Quando realizo uma busca pelo produto "notebook lenovo ideapad"
12    Então o vejo no resultado da consulta
13
14 Cenário: Busca sem resultado
15    Dado que acesso o site de compras
16    Quando realizo uma busca pelo produto "qualquer coisa"
17    Então a loja não encontra produtos relacionados
  
```

7.3. O dia a dia das histórias de usuário

Analistas de negócios: Coletam requisitos de software na forma de histórias de usuários, com foco no comportamento do sistema e nas necessidades do cliente.

Desenvolvedores: são aprovados em cada um dos cenários de teste criados para cada história e em todos os cenários de teste.

Garantia de qualidade: Confirma que todos os cenários de teste foram implementados, entendendo os itens específicos que eles precisam validar.

Donos do produto: entendem as necessidades do usuário (voz do cliente) e as expectativas de comportamento do sistema, fornecendo subsídios para a criação de todos os cenários.

Business Owners podem colaborar mais estreitamente adicionando exemplos do comportamento esperado na forma desses cenários de critérios de aceitação ao sistema.

7.4. Benefícios de histórias com BDD

Existem várias formas de escrever histórias de usuários e critérios de aceite, porém, neste curso, optamos em trazer as técnicas de BDD por ser uma técnica extremamente avançada na construção de produtos digitais, além de trazer vários benefícios, como:

Melhor colaboração da equipe: o uso do BDD para fortalecer suas histórias não deixa espaço para dúvidas e suposições que, às vezes, podem retardar o processo. Em vez disso, você obtém as informações diretamente das partes interessadas em um formato simples, claro e de fácil compreensão, tornando muito mais fácil entender os requisitos.

Visibilidade aprimorada: quando você usa uma linguagem coloquial para escrever suas histórias, permite que todos entendam o andamento do projeto e torna mais fácil obter uma visão de 360 graus do projeto.

Custos reduzidos: a produção de código de qualidade superior que requer menos correção de bugs acelera o tempo de lançamento no mercado e reduz custos, produzindo uma entrega de software mais robusta.

Maior adaptabilidade: o BDD torna muito mais fácil modificar cenários conforme as necessidades mudam.

Capítulo 8. Autonomia do PO no time de produtos

8.1. Chave para o sucesso

Em linhas gerais, o Product Owner é a chave de sucesso na operação de um produto digital. Sua autonomia e responsabilidade principal é garantir que todo o backlog priorizado seja revertido na melhor experiência para os clientes com o maior ROI (retorno sobre o investimento) possível.

Talvez essa realidade deva doer para você que está lendo, mas nossa proposta é trazer verdades. O Product Owner é um profissional Sr., ou seja, não existe PO Jr., PO Pleno; existe o dono do produto que deve ter total autonomia para decidir o que entra ou não na esteira de entrega. Quando não há essa autonomia, o papel do Product Owner está totalmente comprometido, e, com certeza, demandas “Top Down” estão furando a esteira de entregas de maneira constante.



Referências

CAGAN, Marty. **Inspirado** – Como criar produtos de Tecnologia que os clients amam. Rio de Janeiro: Alta Books, 2020.

MÉTODO ÁGIL. **Product Owner**: garante o máximo de ROI. Belo Horizonte, 2021. Disponível em: <https://www.metodoagil.com/product-owner/>. Acesso em: 23 fev. 2022.

PATTON, Jeff. **User Story Mapping**: discover the whole story, build the right product. Massachusetts: O'Reilly Media, 2014.

SALVADOR, Leonardo. Conhecendo o Dual Track: foco na entrega de valor. **Ux Collective Brasil**, 18 de junho de 2020. Disponível em: <https://brasil.uxdesign.cc/conhecendo-o-dual-track-foco-na-entrega-de-valor-7ef90ff13bee>. Acesso em: 23 fev. 2022.

SCRUM.ORG™. **Mastering the Product Owner Stances**. Burlington, 2021. Disponível em: <https://www.scrum.org/courses/professional-scrum-product-owner-advanced-mastering-product-owner-stances-training>. Acesso em: 23 fev. 2022.