

Projeto de *Software* com Métodos Ágeis



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



**Contextualização do Desenvolvimento
de Projetos de *Software* com Métodos Ágeis**

Responsável pelo Conteúdo:

Prof. Me. Artur Marques

Revisão Textual:

Prof.^a Esp. Kelciane da Rocha Campos

UNIDADE

Contextualização do Desenvolvimento de Projetos de *Software* com Métodos Ágeis



- Introdução;
- Revisão de Projeto de *Software*;
- Iniciação do Projeto;
- Situação-problema;
- Visão Geral e Fases;
- Preparação;
- Documentos e Protótipos para o Desenvolvimento Ágil.



OBJETIVO DE APRENDIZADO

- Apresentar os desafios de organização, orquestração e preparação de planejamento de um projeto ágil.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Introdução

Desenvolvimento Ágil é uma abordagem iterativa para o desenvolvimento de *software*, onde equipes de analistas de negócios, *designers* e desenvolvedores trabalham em colaboração com o *product owner* – dono do produto, designado pelo cliente para garantir que a visão, os objetivos de negócios e as necessidades do usuário sejam atendidos.

Para que funcione, ele depende de *feedback* contínuo dos usuários comerciais, os caras para quem você está desenvolvendo algo, em última instância, os usuários. Isso porque as estruturas e ferramentas modernas de tecnologia de *software* de hoje facilitam mudanças em todo o processo de desenvolvimento de *software* para criar exatamente o que a empresa precisa.

Dessa maneira, o desenvolvimento Ágil se tornou muito preferido aos métodos tradicionais de desenvolvimento de *software*, que exigiam documentação extensa antes de qualquer *software* ser entregue. Mas, não se engane, o problema nunca foi o fato de a documentação ser ou não extensa, isso tem mais a ver com preguiça do que com qualquer outra coisa. Isso tem a ver com o **escopo** e com os **requisitos** do *software*.

Afinal, o que você vai presenciar aqui nesta disciplina é um ataque direto ao problema; e sim, vamos fazer documentação ágil e não é pouca, porém somente o essencial.

No mundo VUCA – Volátil, Incerto, Complexo e Ambíguo, que estamos vivendo daqui por diante, você, com certeza, já deve ter percebido que é extremamente raro que um cliente nos apresente um documento especificando exatamente o que ele precisa construir. Na maioria das vezes, é uma ideia ou visão sobre a solução de um problema comercial.

Portanto, da mesma forma que muitos de nós deliramos querendo que o sistema de informação, sua codificação e testes sejam criados quase que por magia, sem esforço de codificação nenhum, o usuário também quer que nós saibamos o que ele quer do sistema. Nada mais justo!

O desenvolvimento ágil de *software* é um processo que transforma a visão de uma necessidade comercial em soluções. Usamos agilidade porque resulta em *software* que suporta melhor as necessidades de nossos clientes em uma linha do tempo mais rápida. E também porque os negócios não podem parar e cada dia de atraso nas ações e desenvolvimento de sistemas pode causar milhões em dinheiro que deixam de entrar para a empresa. Bem, você deveria estar bastante preocupado(a), afinal é daí que virá o dinheiro para pagar seu salário ou seu contrato de prestação de serviços.

De uma forma ou de outra, você, além disso, deverá ter Métodos Ágeis como livros de cabeceira e deverá praticar todos os dias a agilidade, ela é o mantra entoado pelos quatro cantos do planeta, agora que com esses “cisnes negros” segue um exército de desempregados (funcionais e também das crises), que estão dispostos a tudo para encontrar e tirar o tempo perdido para disputar vagas com você.

Mas fique tranquilo(a), você está ainda em vantagem, o importante é aprender tudo isso e “partir para a luta”.

Revisão de Projeto de Software

Nosso foco será projeto de *software* ágil, então utilizaremos *scrum* e um pouco de quadros de orientação visual *kanban*, entre outros. Portanto, é bom recordamos os benefícios que esses métodos, paradigmas ou processos de *software* nos proporcionam.

Esses benefícios foram resumidos no famoso Manifesto Ágil de 2001, que inclui 12 princípios:

- satisfação do cliente através da entrega antecipada e contínua de *software*;
- acomodação dos requisitos de mudança ao longo do processo de desenvolvimento;
- entrega frequente de *software* de trabalho;
- colaboração entre as partes interessadas do negócio e os desenvolvedores ao longo do projeto;
- apoiar, confiar e motivar as pessoas;
- ativar interações cara a cara;
- o *software* de trabalho é a principal medida de progresso;
- os processos ágeis suportam um ritmo consistente de desenvolvimento;
- a atenção aos detalhes técnicos e ao *design* aumenta a agilidade;
- simplicidade;
- as equipes auto-organizadas incentivam grandes arquiteturas, requisitos e projetos.
- reflexões regulares sobre como se tornar mais eficaz.

Portanto, como você pode ver, a ascensão dos processos ágeis no mundo foi a consequência da frustração da indústria e comércio em geral nos anos 90 do século XX. O enorme intervalo de tempo entre os requisitos de negócios, que nada mais eram do que os aplicativos e recursos que os clientes estavam solicitando, e a entrega da tecnologia que atendia a essas necessidades levou ao cancelamento de muitos projetos. Afinal, os negócios, os requisitos e os requisitos do cliente foram alterados durante esse período de atraso, e o produto atenderia às necessidades da época em que foi pedido, mas não as necessidades da data em que foi entregue – lá no futuro. Então, a chave era rapidez e não documentação, percebeu?!

Apesar do Manifesto Ágil ter 17 signatários, os processos que usamos hoje existem por causa das seguintes pessoas: Jon Kern, Kent Beck, Ward Cunningham, Arie van Bennekum e Alistair Cockburn.

Aqui temos os 4 valores do processo Ágil:

- **Indivíduos e interações sobre processos e ferramentas:** é fácil entender a valorização das pessoas mais do que processos ou ferramentas, pois são as pessoas que respondem às necessidades da empresa e conduzem o processo de desenvolvimento;
- **Software de trabalho com documentação abrangente:** o Ágil não elimina a documentação, mas a otimiza de uma forma que fornece ao desenvolvedor o que é necessário para fazer o trabalho sem ficar atolado em minúcias. Requisitos de documentos ágeis, como histórias de usuário, ajudam muito para que um desenvolvedor

de *software* inicie a tarefa de criar uma funcionalidade no *software*. O Manifesto Ágil valoriza a documentação, mas valoriza mais ainda o *software* em funcionamento;

- **Colaboração do cliente sobre negociação de contrato:** negociação é o período em que o cliente e o gerente de produto elaboram os detalhes de uma entrega, com pontos ao longo do caminho em que os detalhes podem ser renegociados. Porém, a colaboração é diferente. O Manifesto descreve um cliente que está envolvido e colabora durante todo o processo de desenvolvimento; portanto, o cliente está lá, com os desenvolvedores, ou ao menos podem incluir o cliente em intervalos de tempos, por exemplo, para demonstrações periódicas;
- **Respondendo à mudança após um plano:** mudança deixa de ser apenas uma despesa. Quando falamos de mudança em Ágil, as prioridades podem ser alteradas de iteração para iteração e novos recursos podem ser adicionados à próxima iteração. A visão Ágil é que as mudanças sempre melhoram um projeto; alterações fornecem valor adicional. Portanto, abrace a mudança. As metodologias ágeis permitem que a equipe Ágil modifique o processo e faça com que ele se ajuste à equipe, e não o contrário.

Bem, para você que está chegando agora, perceba que a abordagem de Desenvolvimento Ágil é caracterizada pela divisão do projeto de *software* em fases curtas, mas que sempre gerarão *software* de trabalho (pronto para executar alguma funcionalidade, algo que não deixe o cliente esperando), isso, logo de cara, permite que o *feedback* inicial seja coletado para ajustar os recursos necessários e melhorar a usabilidade rapidamente, por exemplo. A terminologia associada ao desenvolvimento ágil está diretamente relacionada à abordagem da equipe e ao ritmo da atividade de desenvolvimento.

O *scrum* tem seu termo emprestado do *Rugby* e refere-se a desenvolvimento de sistemas, a uma equipe profissional de desenvolvimento que trabalha bem, automotivada e em conjunto, com foco em manter os processos de colaboração simples e organizados. O trabalho no desenvolvimento ágil de *software* é dividido em *sprints*, que nada mais são que “corridas” curtas (em tempo) que permitem reavaliações e adaptações frequentes.

Vamos resumidamente explicar o que é *scrum* e seus componentes para nivelarmos nossos conhecimentos, é bastante simples e vai ser útil quando estivermos nas outras unidades:

- **Scrum:** trata-se de uma série de práticas de gestão em cada *Sprint* que são utilizadas para envolver as equipes e o proprietário do produto durante todo o ciclo de vida do projeto;
- **Scrum diário (em pé):** o famoso *Daily Scrum*, uma reunião diária de 10 a 15 minutos, em que cada membro da equipe comunica o que trabalhou no dia anterior, no que trabalhará hoje e se possui algum bloqueador (se tem alguma dificuldade em executar o trabalho). Obs.: afirmações como: não sei fazer isso, nunca fiz isso antes, ou não gosto de fazer isso; não são bem vistas e nem devem ser feitas numa reunião, principalmente em um time *scrum*. A menos que você esteja em algum treinamento, aclimatação ou estágio; fora isso, com certeza, não iriam te jogar num time *Scrum* para desenvolver um sistema. Eu espero!
- **Scrum Master:** não é gerente de projeto e não é um programador que vai te salvar caso você esteja em apuros e não achou nada no *Google*. Ele tem a função de facilitar e gerenciar as informações trocadas e, mais que isso, ele é o guardião das

práticas e princípios *scrum* que devem ser seguidos ou postos no eixo toda vez que houver fuga desse propósito. Não há gerente de projeto em *scrum*, ok?

- **Sprint:** trata-se de um período estipulado durante o projeto, no qual o trabalho definido deve ser concluído e preparado para revisão. As *Sprints* quando têm seu tempo definido – por exemplo, 7, 15 ou 21 dias, não mudam, então se o time *Scrum* definir que uma *Sprint* terá 15 dias, não mudará e todo o trabalho definido para aquela *Sprint* deverá ser cumprido em 15 dias. Não existe em *Scrum* atrasos de entrega nem desculpas. O prometido deverá ser entregue!
- **Planejamento da Sprint:** aqui, nós priorizamos quais recursos (funcionalidades) incluiremos no próximo *sprint*;
- **Definir requisitos:** lista das funcionalidades necessárias para atingir as metas de negócios e do usuário. Portanto, não ficamos só nas histórias do usuário, isso é planejamento em alto nível. Precisamos descer em complementar o que se quer dizer quando lemos num cartão de usuário, por exemplo: Eu como gerente quero todos os usuários identificados e com acesso somente à sua área de função no sistema. Talvez você pense: ah, isso é fácil, é só um login com senha... “Só que não”!
- **Arquitetura de design:** mapas do site, jornadas do usuário, diagramas conceituais, modelos de dados, etc;
- **Wireframe:** esquemas de página e/ou mockups de telas que servem como guias visuais que representam a estrutura esquelética (arcabouço) de um aplicativo. Lembram mais um esboço em preto e branco, mas em algumas situações podem ser coloridos também;
- **Protótipo:** modelagem de um aplicativo em que o teste do usuário pode ser realizado para coletar *feedback* antes da *sprint* de desenvolvimento. Pode ser em vários níveis, desde um protótipo navegacional ou chegar em um nível de teste de uma regra de negócio e suas exceções. Nesse caso, as identidades visuais e cromáticas são geralmente adicionadas;
- **Teste do usuário:** aqui entra tudo, UX e Acessibilidade, por exemplo, guidelines do W3C, a usabilidade dos wireframes ou protótipo com usuários reais para reunir o *feedback* necessário para garantir que suas necessidades sejam atendidas;
- **Desenvolvimento de Sprint:** como escrevi anteriormente, varia de 2 a 4 semanas, em que ocorre o desenvolvimento dos recursos priorizados;
- **Desenvolver recursos:** o código é escrito; a arquitetura continua sendo construída e/ou ajustada durante todo o ciclo do projeto;
- **Recursos de teste:** precisa de automação, os testes devem ser automáticos, gerados *scripts* que abrangem todos os cenários para garantir que a funcionalidade esteja se comportando corretamente (testes unitários e outros);
- **Implantar recursos:** subir o código para um repositório central que se integre ao aplicativo ou ambiente existente. Isso deve ser feito de maneira rotineira, muitos times escolhem a cada semana ou a cada quinzena;
- **Revisão da Sprint:** serve para rever os recursos do trabalho. Testes e implantação podem ocorrer aqui;

- **Retrospectiva da Sprint:** envolvem as equipes para compartilhar os destaques e os pontos mais baixos da *sprint* para obter conhecimento sobre o que funcionou bem e o que não funcionou. Sim, é aqui que refletimos sobre as lições aprendidas a cada entrega para não repetirmos os mesmos erros do passado.

Já que falamos de *kanban*, para que utilizamos uma prática da indústria de bens de consumo, como automóveis, por exemplo, em um processo Ágil?

Bom, rapidamente, o *kanban* é uma ferramenta projetada para reduzir o tempo ocioso em um processo de produção. A principal ideia por trás do sistema disso é fornecer exatamente o que o processo precisa quando é necessário. Em japonês, a palavra *Kan* significa visual e *ban* significa cartão, então *kanban* se refere a cartões visuais.

Muitas equipes do *Scrum* também usam o *kanban* como uma ferramenta visual de gerenciamento de processos e projetos. Esses princípios são úteis para adicionar uma camada extra de visibilidade aos projetos.

O *kanban* tem como objetivo visualizar seu trabalho, limitar o trabalho em andamento e maximizar a eficiência ou fluxo. Assim, nos concentraremos em reduzir o tempo necessário para levar um projeto, *sprint* ou história do usuário do início ao fim. Fazemos isso usando um quadro *kanban* e melhorando continuamente o fluxo de trabalho.

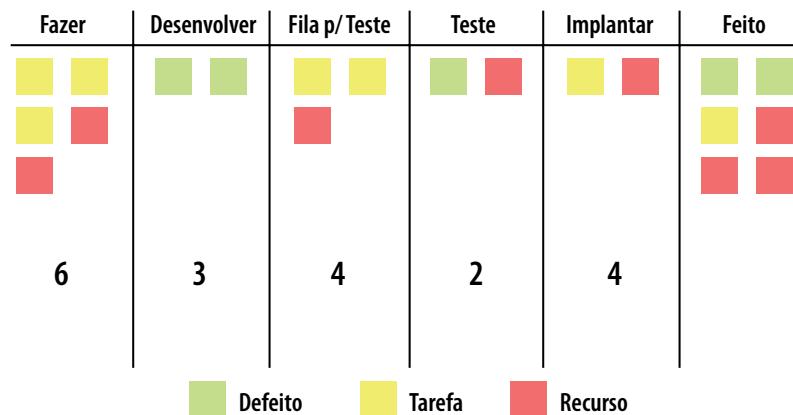


Figura 1 – Exemplo de quadro *Kanban*

- **Buffer:** perceba que este quadro contém 6 colunas (Fazer, Desenvolver, Fila p/ Teste, Teste, Implementar e Feito). A coluna que antecede o Teste (Fila p/ teste) é o que chamamos de *Buffer*. *Buffer* é o conceito de fila antes do gargalo, ou seja, é o que antecede a atividade e está aguardando para ser puxado. O *Buffer* faz com que a equipe mantenha um ritmo e priorize melhor o trabalho, pois desta forma sempre haverá trabalho disponível para ser puxado;
- **Priorização de itens:** veja que os cartões não estão bagunçados no quadro, ou seja, existe uma priorização de itens. Neste caso, os defeitos estão sempre acima, pois encadeiam uma maior prioridade, após eles as tarefas recebem um peso posterior e depois os recursos. A identificação pode ser feita por cores conforme a imagem acima ou por pontos. Essa priorização faz com que a equipe saiba em que trabalhar primeiro, agregando mais valor ao processo;

- **Limite WIP:** em cada coluna existe um número. Este número é o limite WIP, ou seja, o limite de trabalho em andamento que pode estar naquela coluna. Perceba, por exemplo, que a coluna “Desenvolver” ainda poderia receber mais um cartão. (BERNARDO, 2017, p. 3)

Podemos colocar Raias também no quadro *Kanban*.

	Fazer	Desenvolver	Fila p/ Teste	Teste	Implantar	Feito
Tarefas						
Bugs	 	 				
	3	3	2	2	2	
	6	4	4	3	3	

Figura 2 – Exemplo de quadro *Kanban* com raias

Raias: Há uma divisão entre Tarefas e Bugs. Essa divisão é o que conhecemos como raias. O nome remete a uma raia de natação, onde um nadador possui velocidade diferente do nadador na raia ao lado. Essa é uma excelente forma de controlar a demanda, onde em cada raia um fluxo é distinto do outro. No nosso caso perceba que as raias são totalmente diferentes; por exemplo, na coluna “Fazer” a raia de cima possui um limite WIP (trabalho em andamento) de 3 cartões, enquanto a raia abaixo possui um limite WIP de 6 cartões. As raias foram divididas por tipo de item (bugs e tarefas), porém você pode dividir de acordo com a sua necessidade, podendo optar também por classe de serviço, prioridade ou tamanho. (BERNARDO, 2017, p. 4)

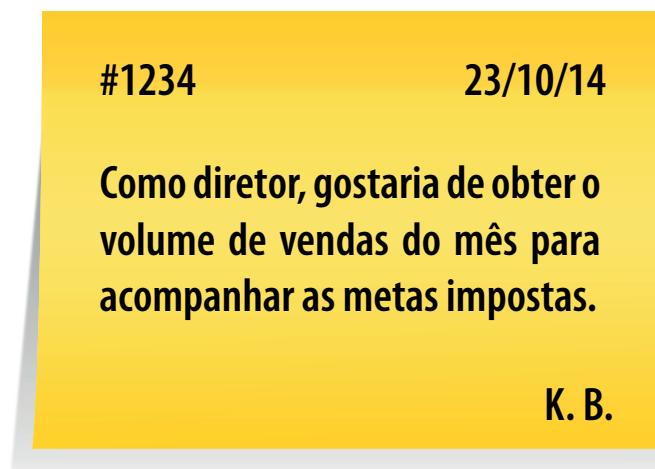


Figura 3 – Cartões de Parede *Kanban*

Fonte: Reprodução

É importante que os cartões possuam informações simples e descritivas. Na imagem acima é possível perceber que a informação foi atribuída de maneira organizada, observe que no topo do cartão, da esquerda para direita, temos um código (#1234) que simboliza o número do item. Esse número poderia ser a representação de um código retirado de algum sistema de controle, onde lá mais informações poderiam ser encontradas. Ao lado, existe uma data (23/10/2014) que representa a entrada desse item, ela é considerável porque evidencia o que está sendo mantido há mais tempo no sistema *Kanban*. Ela poderia também representar o limite, às vezes existem requisitos que precisam de uma data limite; sendo assim, é interessante analisar se para você é mais viável manter essa data como limite ou entrada, porém caso decida por uma, faça os outros cartões seguirem o mesmo padrão, evitando assim confusões. Logo abaixo temos uma descrição. Observe que ela é curta e simples, é sempre contada a partir da perspectiva da pessoa que deseja a nova funcionalidade. Normalmente, seguem um modelo simples, como, por exemplo, “Como/ Sendo <quem>, eu quero/gostaria/devo/posso <o que>, para que/de/ para <porque/resultado>“. Por fim, duas iniciais (K. B.) são vistas ao final do cartão, elas representam o nome do responsável naquele momento pelo item, no caso Kleber Bernardo, ou seja, eu. :) Você poderia também inserir o nome completo ou colar um avatar que representasse o responsável. (BERNARDO, 2017, p. 4)

Isso vai ser o suficiente para irmos estabelecer o que faremos no projeto daqui por diante.

Iniciação do Projeto

Agora que nivelamos o essencial para podermos ir para a fase de iniciação, você precisa entender desde já que a **comunicação** é tipicamente o que levanta ou quebra uma equipe de desenvolvimento em *software*.

Mesmo que todos os outros aspectos da equipe sejam ideais, quando falta comunicação, você terá desempenho abaixo do esperado.

A comunicação eficaz também pode permitir que as equipes superem muitas circunstâncias que as colocam niveladas abaixo do ideal.

A boa comunicação começa com habilidades de escuta; se não se pode ouvir, avaliar e entender a mensagem, não se pode responder efetivamente a ela.

Mas não esgotamos somente nesse assunto. Para que as equipes de desenvolvimento de *software* funcionem efetivamente, elas deverão possuir **metas, objetivos, estratégias e táticas** claramente definidas.

Todo projeto de *software* ágil começa com uma visão e um conjunto de requisitos iniciais.

A diferença é que, em vez de passar meses detalhando especificações exaustivas, um projeto ágil começará a ser desenvolvido assim que os requisitos forem aprimorados

apenas o suficiente para que os codificadores comecem a construir as partes principais do sistema de informação.

Atenção também deve ser dada ao fato de que, durante todo o projeto, os *Product Owner*, frequentemente chamados de donos do produto, trabalham um passo à frente dos desenvolvedores, cultivando uma lista de pendências priorizada que inclui tarefas granulares a serem abordadas no curto prazo e itens definidos grosseiramente para mais adiante.

Você perceberá também que os testadores trabalham em paralelo com os desenvolvedores, concentrando-se na automação para que as alterações possam ser totalmente testadas poucos minutos após o *check-in*.

Talvez você esteja buscando alguma analogia, Ágil é muito rico nisso. Imagine que você está querendo trabalhar em seu jardim. A maioria das pessoas faz a paisagem nos jardins em etapas, colocando um gramado em primeiro lugar, depois plantam uma ou duas árvores, depois adicionam alguns arbustos e, finalmente, colocam aquelas plantinhas perenes com flores sazonais para padronizar e harmonizar o jardim em acordo com a estação.

Bem, um sistema de informação ou um *software* é parecido com um jardim nesse aspecto, é algo vivo, dinâmico e que crescerá e amadurecerá com o tempo.

Se existe um objetivo em uma equipe ágil é construir isso pedaço a pedaço e fornecer entregas desse “jardim” totalmente funcionais ao final de cada mexida, que chamamos de *Sprint*.

Voltemos a um exemplo, que utilizei anteriormente nesse nosso material de balizamento, referente ao **acesso controlado**. Talvez na funcionalidade básica de login de um site, desde a interface do usuário até o banco de dados, tenha sido criada durante essa iteração uma *Sprint*. Mais adiante, na próxima iteração, um recurso do tipo “esqueci a senha”, “recupere sua senha” ou “fale com o administrador” pode ser adicionado e assim por diante.

Quando escrevo isso, quero lhe dizer que ágil é uma mudança na cultura organizacional do lugar em que você trabalha, não é simplesmente aceitar a chegada de um consultor, contratar pessoas e agora todo mundo é ágil, rapaziada, vamos voar!

Nem que você tenha rios de dinheiro e todas as ferramentas ágeis do mundo, isso não tornará nenhuma equipe de desenvolvimento de sistemas Ágil, assim como comprar um monte de máquinas novas não transformará uma fábrica em uma indústria 4.0.

O papel das ferramentas é simplesmente refletir os processos de suas equipes e torná-los menos onerosos.

Essa mudança de cultura envolve olhar para os riscos. De certa forma, Processo Ágil é apenas outro nome para o bom e velho gerenciamento de riscos. Trata-se de um guia, de um processo para avançar sob incerteza severa provocada pela *internet*, pela globalização, pela concorrência sem fronteira e pelo mundo como ele é hoje.

Em vez de tentar prever cem passos à frente, Ágil força você a desistir da previsão de longo prazo de um caminho exato para uma direção.

Por isso planejamos o sistema ou *software* apenas alguns passos à frente, executamos e avaliamos se ainda estamos na direção certa e planejamos os próximos passos de acordo. Escrevi isso para que fique claro que a importância do movimento Ágil não é vender ferramentas ou métodos, mas conscientizar sobre os conceitos de gerenciamento de riscos principalmente.

Situação-problema

Descrição geral: a ACME Cafeteria Gourmet é uma empresa de café “gourmetizado”, é nova no mercado, entretanto já oferece serviços diferenciados e inovadores para aquele momento que é só seu ou para dividir com as pessoas que você quiser. Trabalharemos somente na via digital, sem lojas físicas, e em nosso site oferecemos a opção de o cliente pedir seus *kits* e receber mensalmente em casa. O nosso objetivo é satisfazer todas as expectativas de nossos clientes.

A visão macro:

E-commerce

Em linhas gerais, o negócio é venda através da *internet* de *kits* de café gourmetizado para que o próprio cliente possa preparar em casa, no melhor estilo barista, um café de alta qualidade feito pelas suas próprias mãos.

Para tanto, haverá um catálogo digital onde os produtos são divididos em *kits* de cafés, chás e outros produtos para barista contendo coadores diferentes, xícaras especiais, bules, cafeteiras italianas, prensa francesa e, por fim, tipos de cafés diferentes vendidos em sacos já moídos para vários tipos de cafeteiras e métodos.

Os pedidos são feitos a partir do cadastro do cliente e a escolha do seu tipo de assinatura.

Os *kits* com o café *gourmet* do mês ou chá são montados e enviados mensalmente aos seus destinatários em qualquer canto do Brasil através dos Correios, via Sedex.

Haverá uma área onde o cliente deverá se identificar e na empresa teremos dois perfis, um para o *marketing* e outro para pedidos/estoques com controle de acesso.

Canecas, xícaras, pires, colheres, filtros e cafeteiras personalizadas: escolha a cor e a estampa (mande a sua ou escolha uma das nossas).

Descontos em livrarias parceiras e em cursos de barista: cursos com os melhores baristas – aprenda do básico ao avançado na arte dos cafés e chás. As melhores livrarias com os melhores preços para os melhores clientes.

Receitas *gourmet* com passo a passo para fazer você mesmo: compre a receita e receba todo o *kit* necessário para fazê-la na sua casa.

Newsletter com dicas sobre acompanhamentos, modos de preparo e novidades sobre o mundo dos amantes de cafés e chás: dicas, novidades, informações exclusivas.

Formas de pagamento: cartões de crédito, débito, *paypal*, mercado pago, boleto bancário. Garantimos a segurança de seus dados; caso não receba o produto, receba seu dinheiro de volta.

Esse é um exemplo de visão inicial. Às vezes isso vem direto, logo após o time de negócios fazer o *Business Model Canvas*, ou o Quadro de Modelo de Negócios, que é uma ferramenta ágil.

Visão Geral e Fases

Vamos trabalhar com os artefatos relacionados à preparação para o pessoal de codificação trabalhar. Em um projeto scrum, isso é bem rápido ou, se preferir, Ágil.



O que usaremos de ferramental:

- **Visão de Negócio:** *Canvas*
 - *Canvas*: Como estruturar seu modelo de negócios, disponível em: <https://bit.ly/3bbPls>
 - *Business Model Canvas*, disponível em: <https://bit.ly/38XmIb2>
- **Cronograma e controles:**
 - *Trello*, disponível em: <https://bit.ly/3bbVv1l>
 - *ProjectLibre – Project Management*, disponível em: <https://bit.ly/3rVBQJf>
 - Diagramas UML, disponível em: <https://bit.ly/3ohk29e>
 - Mapeamentos de negócios com *Bizagi*, disponível em: <https://bit.ly/3hlpDTs>
 - Estudo I.H.C. com *Mockups*, disponível em: <https://bit.ly/3hKrJSH>
 - **Modelagem Banco:** *Workbench*, disponível em: <https://bit.ly/3ofJEDu>
 - **Protótipo navegacional:** HTML 5 e CSS3.

As fases de nosso projeto tratarão do seguinte:

- **Sprint 1:** Entender o negócio, estudar concorrentes, fazer o *benchmarking*, macro-fluxo do negócio, desenvolver as histórias do negócio (criar os *story cards*);
Qual é o modelo da receita (quais os controles financeiros)?
- **Sprint 2:** *Business Model Canvas*, lapidar a visão do negócio, revisar as histórias de usuário, *backbone* das histórias de negócio (Tema, Épico e Cartões);
- **Sprint 3:** Priorização do *backbone*, *Planning Poker*, *Story Map*;
- **Sprint 4:** Aprofundamento com Requisitos funcionais, não funcionais e regras de negócio;
- **Sprint 5:** Diagrama Geral de Caso de Uso, Diagramas de Caso de Uso, Diagrama de Classes;
- **Sprint 6:** Mapa Navegacional – todas as telas, todos os campos, todos os Casos de Uso e todas as mensagens de erro;
- **Sprint 7:** Modelagem do banco de dados;
- **Sprint 8:** Casos de Uso Expandido e Diagramas de Sequência;
- **Sprint 9:** Protótipo de Telas e Mapa Navegacional;
- **Sprint 10:** Desenvolvimento do Cronograma.

Colocaremos na forma de *sprint* dentro das outras unidades para podermos acompanhar adequadamente.

Decerto que não desenvolverei um sistema de exemplo, pois isso seria um livro e não um curso. Mas, farei vários artefatos de exemplo baseados no case proposto. Você deverá entender os exemplos e treinar para ganhar fluência, dessa forma conhecerá mais profundamente o projeto Ágil.

Preparação

Proponho que para que você acompanhe essa disciplina sejam baixados e instalados os aplicativos acima designados com os *links*.

Também peço que crie um diretório no disco rígido de seu computador ou algo em nuvem como o *One Drive* da *Microsoft* ou o *Google Drive* da *Google*, porque já vêm com integração de editor de texto, planilha, apresentação etc.

Documentos e Protótipos para o Desenvolvimento Ágil

Em linhas gerais, nosso propósito aqui é desenvolver alguns artefatos, no melhor estilo; eu começo e nos exercícios práticos, vocês terminam.

Aprender na prática significa que você deve treinar à exaustão, assim será um(a) profissional mais assertivo(a), direto(a) e pronto(a) para enfrentar qualquer exame de seleção em empresas ou até mesmo trabalhar por conta, coisa cada vez mais comum na área de TI.

Como documentos, pretendemos gerar:

- Documento geral do negócio;
- Diagrama BPMN do negócio;
- Cartões de História do Cliente;
- *Backlog* do produto (Temas, Épicos e Histórias);
- Escolha das histórias com maior valor;
- Priorização das *Sprints*;
- Definição de pronto;
- Definição da *Sprint ZERO*;
- Quebra do *Backlog* do produto em *Sprints* 1 a *n*;
- Aprofundando-se nas *Sprints* próximas (duas *sprints* sequenciadas);
- Regras de negócio, requisitos funcionais e não funcionais vinculados às funcionalidades;
- Diagrama de classes;

- Diagrama geral de casos de uso;
- Diagrama individual de caso de uso;
- Documentos de caso de uso expandidos;
- Diagramas de sequência;
- Diagramas de classe;
- Modelagem conceitual do banco de dados;
- Modelagem lógica do banco de dados;
- Mapa conceitual;
- Mapa navegacional;
- Mockups da IHC, ao menos os principais;
- Protótipo Conceitual para averiguar disposições e um pouco de *design* projetado em pptx (rapidez e ajustes) antes de codificar, agilidade;
- Cronograma;
- Jogo do planejamento (ao menos uma rodada para aprender a estimar);
- Cerimônias scrum;
- Base de lições aprendidas e riscos.

Sim, é um caminho longo e desafiador, mas tudo o que você precisa para resolver seu projeto estará aqui e nas próximas unidades.

Vamos em frente!

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

▶ Vídeos

Scrum – A arte de fazer o dobro do trabalho na metade do tempo | Jeff Sutherland | Resumo do livro
https://youtu.be/0z15GwRK3_I

Visão de produto no *Scrum*

<https://youtu.be/vg1S1WYZa6o>

O papel do *Product Owner* em 15 minutos

<https://youtu.be/7IhnYbmovb4>

Aprenda *Kanban* em 9 minutos (Desenvolvimento ágil de *software*)

<https://youtu.be/WjZBnYa58B4>

Scrum master – Produtividade e performance

<https://youtu.be/mYx8JPQplEg>

📄 Leitura

Manifesto para Desenvolvimento Ágil de *Software*

<https://bit.ly/3rPl9hv>

Referências

BERNARDO, K. **Kanban do início ao fim.** 2017. Disponível em: <<https://www.culturaagil.com.br/kanban-do-inicio-ao-fim/>>. Acesso em: 26/07/2020.



Cruzeiro do Sul
Educacional