

DIVERSIFICANDO NA PRÁTICA A UTILIZAÇÃO DOS MÉTODOS ÁGEIS: SCRUM E KANBAN – DESENVOLVIMENTO DE UMA LOCADORA DE VEÍCULOS

Ricardo Alves Ribeiro Filho¹

<https://orcid.org/0009-0008-5393-6781>

Renata Mirella Farina²

<https://orcid.org/0000-0001-9602-5293>

Fabiana Florian³

<https://orcid.org/0000-0002-9341-0417>

Resumo: Cada vez mais o grau de exigência dos clientes no ambiente empresarial quanto ao tempo de entrega dos produtos e melhor qualidade. Quando há uma referência ao desenvolvimento de um software, percebe-se uma grande mudança nos processos hoje estabelecidos, que devem seguir métodos específicos de acordo com a área apropriada. O objetivo deste trabalho é caracterizar os métodos ágeis (*Scrum* e *Kanban*) para gerenciar projetos complexos e identificar qual deles é o melhor aplicado para determinada situação, inclusive com a possibilidade de serem utilizados em conjunto. Foi realizada uma pesquisa bibliográfica e foi desenvolvido uma aplicação como objetivo de se utilizar a comparação de duas Metodologias Ágeis relacionando as boas práticas adotadas em ambos os métodos, através dos conceitos, valores, diferenças de cada uma. Concluiu-se que é possível usar Scrum e Kanban juntos em uma abordagem prática, combinando os melhores elementos de ambas as metodologias. As equipes aproveitaram a flexibilidade e a adaptabilidade do Scrum enquanto usaram a representação visual e os princípios baseados em fluxo do Kanban para otimizar o tempo e a qualidade do trabalho.

Palavras-chave: Metodologia Tradicional, Kanban, Scrum, Métodos Ágeis.

¹ Graduando do Curso de Engenharia da Computação da Universidade de Araraquara- UNIARA. Araraquara-SP. E-mail: rarfilho@uniara.edu.br

² Mestre, Analista de Sistemas, Docente do Curso de Engenharia da Computação e Sistemas de Informação da Universidade de Araraquara- UNIARA. Araraquara-SP. E-mail: rmfarina@uniara.edu.br

³ Doutora, Economista, Bacharel em Direito e Docente do Curso de Engenharia da Computação e Sistemas de Informação do Curso de Engenharia da Computação da Universidade de Araraquara- UNIARA. Araraquara-SP. E-mail: f Florian@uniara.edu.br



DIVERSIFYING THE USE OF AGILE METHODS: SCRUM AND KANBAN – DEVELOPMENT OF A CAR RENTAL COMPANY

Abstract: Increasingly the degree of demand from customers in the business environment regarding product delivery time and better quality. When there is a reference to software development, a great change is perceived in the processes established today, which must follow specific methods according to the appropriate area. The objective of this work is to characterize the agile methods (Scrum and Kanban) to manage complex projects and identify which one is the best applied for a given situation, including the possibility of using them together. A bibliographic research was carried out and an application was developed with the objective of using the comparison of two Agile Methodologies relating the good practices adopted in both methods, through the concepts, values, differences of each one. It was concluded that it is possible to use Scrum and Kanban together in a practical approach, combining the best elements of both methodologies. Teams took advantage of the flexibility and adaptability of Scrum while using the visual representation and flow-based principles of Kanban to optimize time and quality of work.

Keywords: Traditional Methodology, Kanban, Scrum, Agile Methods

Submetido em: 07/06/2023 – **Aprovado em:** 29/06/2023 – **Publicado em:** 30/06/2023

1 INTRODUÇÃO

Scrum e *Kanban* são duas abordagens diferentes para gerenciamento de projetos que podem ser usadas em uma variedade de configurações, incluindo desenvolvimento de sistemas de informação (CRUZ, 2014).

O *Scrum* é uma estrutura interativa e incremental para gerenciar projetos complexos. Ele foi projetado para ajudar as equipes a entregar produtos em ciclos curtos, chamados de "*sprints*", que geralmente duram de duas a quatro semanas. No *Scrum*, os membros da equipe trabalham de forma colaborativa para definir e priorizar uma lista de trabalho, chamada de "*backlog*". No início de cada *sprint*, a equipe seleciona um subconjunto dos itens do *backlog* para trabalhar e se compromete a completá-los até o final do *sprint*. A equipe acompanha seu progresso usando um quadro visual, chamado de "quadro *scrum*", que mostra o estado de cada item do *backlog* à medida que avança no processo de desenvolvimento (MASSARI, 2016).

O *Scrum* enfatiza a importância da comunicação e colaboração cara a cara e conta com reuniões regulares, chamadas de "eventos *scrum*", para ajudar a equipe a permanecer no caminho certo e fazer os ajustes necessários (MASSARI, 2016).

Kanban é um método para gerenciar o fluxo de trabalho em uma equipe. Baseia-se na ideia de "visualizar" o trabalho criando uma representação visual do processo de trabalho, normalmente usando um quadro com colunas e cartões. O quadro é usado para acompanhar o andamento do trabalho à medida que avança no processo de desenvolvimento, desde a ideia até a conclusão. As colunas no quadro representam os diferentes estágios do trabalho e os cartões representam os itens de trabalho individuais (PMI, 2017).

O *Kanban* estimula a melhoria contínua ao identificar e eliminar desperdícios no processo de trabalho, podendo ser utilizado em conjunto com outras metodologias, como o *Scrum* (PMI, 2017).

O objetivo deste trabalho é caracterizar os métodos ágeis (*Scrum* e *Kanban*) para gerenciar projetos complexos e identificar qual deles é o melhor aplicado para determinada situação, inclusive com a possibilidade de serem utilizados em conjunto.

Foi realizada pesquisa bibliográfica com foco nos temas *Kanban* e *Scrum* e foi desenvolvido uma aplicação comparando as duas Metodologias Ágeis e relacionando as boas práticas adotadas em ambos os métodos, através dos conceitos, valores, diferenças de cada uma.

2 REVISÃO BIBLIOGRÁFICA

2.1 METODOLOGIAS ÁGEIS

De acordo com Cruz (2016) a metodologia ágil é um processo pelo qual uma equipe pode gerenciar um projeto dividindo-o em várias etapas e envolvendo colaboração constante com as partes interessadas e melhoria e iteração contínuas em todas as etapas. A metodologia ágil começa com os clientes descrevendo como o produto final será usado e que problema será resolvido. Isso esclarece as expectativas do cliente para a equipe do projeto. Quando o trabalho começa, as equipes passam por um processo de planejamento, execução e avaliação, o que pode mudar a entrega final para atender melhor às necessidades do cliente. A colaboração contínua é essencial, tanto entre os membros da equipe quanto com as partes interessadas no projeto, para tomar decisões totalmente informadas.

Nessas premissas o desenvolvimento ágil de software refere-se a um grupo de metodologias de desenvolvimento de software baseadas no desenvolvimento iterativo, em que requisitos e soluções evoluem através da colaboração entre equipes multifuncionais organizadas. Métodos ágeis ou processos ágeis geralmente promovem um processo disciplinado de gerenciamento de projetos que incentiva a inspeção e adaptações frequentes, uma filosofia de liderança que incentiva o trabalho em equipe, a auto-organização e a responsabilidade, um conjunto de práticas recomendadas de engenharia destinadas a permitir a entrega rápida de software de alta qualidade, e uma abordagem de negócios que alinha o desenvolvimento às necessidades do cliente e aos objetivos da empresa.

A metodologia ágil não é um conjunto de regras e nem é um conjunto de diretrizes. Em vez disso, o processo pode se caracterizar por um conjunto de princípios que incentivam flexibilidade, adaptabilidade, comunicação e software de trabalho sobre planos e processos. Ainda segundo o autor, o desenvolvimento ágil de software permite que a equipe trabalhe em conjunto de maneira mais eficiente e eficaz no desenvolvimento de projetos complexos. Consiste em práticas que exercitam técnicas iterativas e incrementais que são facilmente adotadas e apresentam ótimos resultados. Esses métodos e metodologias atendem a todas as necessidades de uma indústria de desenvolvimento de software, desde o design e a arquitetura do software, o desenvolvimento e o teste até o gerenciamento e as entregas do projeto. Além disso, os métodos e metodologias ágeis também abrem espaço para a melhoria de processos como parte integrante de cada entrega (VAZQUEZ, 2016).

O mesmo autor reforça que o *Scrum* permite focar na entrega do valor do negócio no menor tempo possível. Inspecciona rápida e repetidamente o software de trabalho real. Ele enfatiza a responsabilidade, o trabalho em equipe e o progresso iterativo em direção a uma meta bem definida. O *Scrum Framework* geralmente lida com o fato de que os requisitos provavelmente mudarão ou na maioria das vezes não são conhecidos no início do projeto.

2.2 CARACTERIZAÇÃO DO SCRUM COM O KANBAN

O *Scrum* é um *framework* de Gerenciamento Ágil de Projetos que pode ser utilizado para a construção de qualquer tipo de produto, principalmente por sua característica de ser iterativo e incremental. Devido o curto tamanho dos seus ciclos de desenvolvimento, as modificações e adaptações ocorrem para a correção dos desvios encontrados.

Como definição, estudos indicam que o *Scrum* é um processo empírico, onde o conhecimento provém da experiência e da tomada de decisão baseada no que é conhecido, e sua utilização está sustentada em 3(três) pilares: transparência, inspeção e adaptação (Figura 1).

Figura 1 - Pilares do Scrum



Fonte: <http://tiexames.com.br>

Além dos 3(três) pilares o *Scrum* também apresenta 5(cinco) valores: Coragem, foco, comprometimento, respeito e abertura.

Ao ser absorvido os 5 (cinco) valores pelo *Time Scrum*, os 3(três) pilares se fortalecem cada vez mais, aumentando sensivelmente a confiança do grupo.

O *Scrum* é um *framework* leve, simples de entender, e extremamente difícil de dominar, pois exige o equilíbrio entre flexibilidade e disciplina.

O *framework Scrum* consiste de *times Scrum* associados a papéis, eventos, artefatos e regras. As regras fazem a integração entre os papéis e os artefatos.

Cada componente dentro do *framework* serve a um propósito específico e é essencial para o uso e sucesso do *Scrum* (SCHWABER, 2023). A partir desses conceitos é possível deduzir que o resultado será transformacional para a empresa fictícia, as histórias agora serão alocadas internamente pela equipe do *Scrum*, liberando essa responsabilidade do líder do projeto.

A equipe terá o poder de dividir o trabalho como achar melhor e se afastar dos especialistas internos ao longo do tempo. Os desenvolvedores serão mais interessados no trabalho, e as habilidades são mais bem distribuídas entre os membros da equipe. Agora há menos histórias em desenvolvimento, o que significará entregas mais rápidas de novos recursos.

O *Scrum* também adicionará maior visibilidade para todas as partes interessadas no que está acontecendo. O comprometimento em produzir algo até o final de cada *sprint* dará não apenas aos desenvolvedores, mas também a todas as partes interessadas, uma visão do progresso.

Isso será completamente novo para a empresa, uma vez que anteriormente meses se passavam antes que qualquer trabalho fosse mostrado a alguém. Será benéfico ver o progresso do produto, em vez de apenas testemunhar uma grande revelação final.

Obviamente, o cliente também se beneficia enormemente com essa abordagem. O cliente está obtendo um produto que deseja e não algo que a equipe apenas pensava que ele queria. Esse produto também tende a ser de maior qualidade, com defeitos identificados no início do processo através das revisões no final de cada *sprint* de duas semanas.

Os resultados não poderão acontecer com a abordagem tradicional de execução de projetos da organização. O cliente não teria visto o produto até o final do processo de desenvolvimento, quaisquer alterações envolveriam reescritas caras. Também teria havido um impacto em outros projetos, porque os membros da equipe teriam sido vinculados a essas reescritas.

2.2.1 O Time e os papéis do Scrum

No *Scrum* a equipe é chamada de Time, e possui os seguintes papéis: *Product Owner* (PO), *Scrum Master* (SM) e Equipe de Desenvolvimento. O *Product Owner* é quem colhe o *backlog* do produto (funcionalidades para atender ao cliente/mercado, ordena o *backlog* do produto, fazendo a priorização por ROI, Valor, Riscos, Necessidades, planeja as releases, detalha e explica o *backlog* do produto de maneira clara para o Time.

O *Product Owner* faz o que for necessário para o Time entregar o máximo de valor por *Sprint*. O Time de Desenvolvimento é o responsável por transformar os itens do *backlog* em incremento de produto pronto. São auto-organizados e multidisciplinares, comprometidos com a meta estabelecida, comunicativos e resolvem seus conflitos.

O Time é composto de no mínimo 3 (três) e no máximo 9 (nove) integrantes. O *Scrum Master* é o responsável por garantir que o processo *Scrum* seja praticado na empresa da forma correta, remover, caso existam, os impedimentos, treinar as equipes, ser uma Liderança Servidora.

Aplicando este conceito para definir os papéis dos envolvidos, a empresa fictícia deverá mudar todas as suas equipes de desenvolvimento de produtos para uma abordagem *Scrum*. Embora a empresa consiga fazer a mudança rapidamente, em apenas seis meses, ainda não será rápida o suficiente para as equipes de desenvolvimento que procuram adotá-la.

As equipes finais para fazer a transição deverão cair positivamente sobre si mesmas para se mudarem para o *Scrum*. Claramente, *Scrum* é a solução certa no momento certo para a empresa de exemplo. A empresa fictícia conseguirá obter benefícios rapidamente, em grande parte porque todos os envolvidos reconhecerão a oportunidade e se comprometerão com ela.

A qualidade da codificação e o envolvimento da equipe no processo aumentarão enormemente, o processo de desenvolvimento possuirá mais estrutura. As rodadas quinzenais proporcionam um ritmo regular às equipes, o que será muito reconfortante para as equipes e para os gerentes.

Os gerentes gastarão menos tempo gerenciando a equipe, que por sua vez os libera para contribuições mais valiosas do que o gerenciamento de tarefas. A moral e o ambiente de trabalho para os desenvolvedores serão muito melhores. Alguns desenvolvedores serão totalmente transformados, capacitar e dar voz a eles os tirará de seu papel vedado e realmente aumentará sua confiança.

A comunicação será aprimorada dentro da equipe de desenvolvimento e também entre desenvolvedores e proprietário do produto, aumentando o nível de envolvimento em todos os níveis. Os principais desafios que a organização enfrentará com seu antigo modelo de execução de projeto não serão mais um problema.

2.2.2 Eventos do Scrum

O *Scrum* estabelece eventos, cuja finalidade é a de reduzir as reuniões não previstas e, manter a frequência dos eventos. Todos eventos são *time-boxed*, ou seja, possuem duração máxima que não pode ser modificada (SUTHERLAND, 2014).

São 5 (cinco) os eventos: *Sprint*, Planejamento da *Sprint*, Reunião Diária, Revisão e Retrospectiva. Ainda existe o evento denominado *Release*, não referenciado no *Scrum Guide* (SCHWABER, 2023).

É dentro das *Sprints* que o time transforma os itens de *backlog* em partes de produto pronto, ou seja, um potencial incremento do produto utilizável.

A *Sprint* possui uma duração máxima de 4 (quatro) semanas, e durante a sua execução não é permitido mudanças que coloquem em risco o objetivo da *Sprint*. As metas de qualidade não diminuem. Por ser o coração do *Scrum*, a *Sprint* possui outros eventos encapsulados.

O Planejamento da *Sprint* possui uma duração máxima de 8 (oito) horas, e o time deve ter as respostas para os seguintes questionamentos:

- Qual o objetivo da *Sprint*?
- O que pode ser entregue como resultado do incremento na próxima *Sprint*?
- Como o trabalho necessário para entregar o incremento será realizado?

A reunião diária tem como objetivo, sincronizar as tarefas e montar um plano para as próximas 24 (vinte e quatro) horas. A reunião ocorre com todo o Time em pé, sempre no mesmo local e horário, e tem duração máxima de 15(quinze) minutos. Cada integrante do Time deve responder as 3 (três) perguntas: O que foi realizado desde a última reunião diária?

- O que será feito até a próxima reunião?
- Existem impedimentos?

A revisão tem como objetivo, a apresentação e inspeção do incremento gerado. A reunião tem duração máxima de 4 (quatro) horas para uma *Sprint* de 4 (quatro) semanas, e 2(duas) horas para uma *Sprint* de 2 (duas) semanas.

Durante a Revisão, o Time apresenta o que foi feito, responde a perguntas sobre o incremento e quais problemas ocorreram e como foram solucionados. O *Product Owner* avalia e identifica o que está realmente pronto e analisa se a meta da *Sprint* foi alcançada.

A Retrospectiva tem como objetivo, inspecionar como a última *Sprint* foi em relação às pessoas, aos relacionamentos, aos processos e as ferramentas. A reunião tem duração máxima de 3 (três) horas.

Para a melhoria do próximo ciclo de desenvolvimento o Time questiona o que foi feito de bom, o que foi feito de ruim, o que será mantido e, o que será mudado e como proceder de posse das respostas às 4 (quatro) perguntas acima, são elaboradas as Lições Aprendidas para serem aplicadas preferencialmente na próxima *Sprint*.

2.2.3 Artefatos

Os artefatos do *Scrum* representam a transparência e oportunidades para inspeção e adaptação. São 3 (três) os artefatos: *Backlog* do Produto, *Backlog* da *Sprint* e Incremento. O *Backlog* do Produto é um inventário priorizado e ordenado das necessidades em relação ao valor do produto.

Seu conteúdo pode ter novas funcionalidades, casos de uso, defeitos, *bugs*, requisitos não funcionais, riscos, histórias de usuário, enfim, tudo o que for necessário para o desenvolvimento do produto (SCHWABER, 2023).

O trabalho a ser feito é estimado dentro do *Backlog* do Produto que é gerenciado pelo *Product Owner* e desenvolvimento pelo Time. O *Backlog* da *Sprint* é o conjunto de itens do

Backlog do produto que foram selecionados para a *Sprint*, juntamente com um plano para entregar o incremento do produto e atingir o objetivo da *Sprint*.

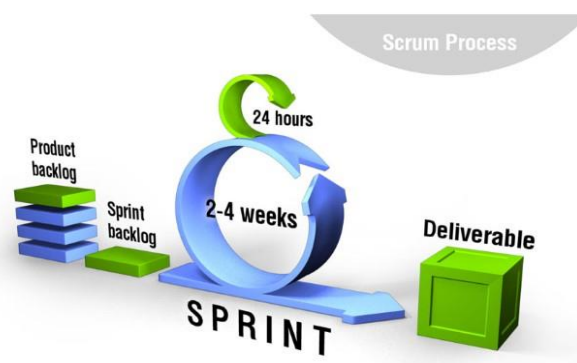
Este artefato pertence exclusivamente ao Time de Desenvolvimento, e é uma imagem em tempo real do trabalho que o Time de Desenvolvimento planeja completar durante a *Sprint*.

O Incremento é a soma de todos os itens do *Backlog* da *Sprint* completados durante a *Sprint* e o valor dos incrementos de todas as *Sprints* anteriores. Ao final de cada *Sprint*, um novo incremento deve estar pronto, significando que estará na condição de utilizável e atender a definição de pronto do Time *Scrum*.

2.2.4 Ciclo de vida do Scrum

O Ciclo de Vida do *framework Scrum* é apresentado na figura 2.

Figura 2 - Ciclo de Vida do Scrum



Fonte: CAMARGO, 2023

- **Product Backlog:** Refere-se a uma acumulação de trabalho num determinado intervalo de tempo. É uma espécie de estoque de produtos ainda não produzidos.
- **Sprint Backlog:** É uma lista de atividades que precisar ser feita durante uma Sprint.
- **Sprint:** É um período entre 2 a 4 semanas que uma equipe trabalha para concluir uma quantidade definida de trabalho.
- **Deliverable:** Produto pronto para a entrega para o cliente após passar pelos passos anteriores de desenvolvimento.

3 RESULTADOS DA UTILIZAÇÃO DO SCRUM E KANBAN

É possível usar *Scrum* e *Kanban* juntos em uma abordagem prática, combinando os melhores elementos de ambas as metodologias. Isso pode ser particularmente útil em testes, pois permite que as equipes aproveitem a flexibilidade e a adaptabilidade do *Scrum* enquanto usam a representação visual e os princípios baseados em fluxo do *Kanban* para otimizar seu trabalho.

Nesta abordagem híbrida, as equipes podem usar *Scrum* para planejamento e gerenciamento de iteração, enquanto usam quadros *Kanban* para visualizar o fluxo de trabalho e acompanhar o progresso. Isso pode ajudar as equipes a identificar gargalos e otimizar seu fluxo de trabalho, mantendo a entrega iterativa e incremental do *Scrum*.

No geral, usar *Scrum* e *Kanban* juntos pode ser uma maneira poderosa de gerenciar e concluir o trabalho com eficácia, principalmente em testes onde flexibilidade e adaptabilidade são importantes.

3.1 Dos Resultados das Aplicações Práticas

Dos resultados que foram obtidos para a *Sprint* em questão, foi necessário elencar um problema simples como, por exemplo, um sistema de locação, considerando inicialmente que a estrutura técnica da entidade Cliente já está pronta, as divisões empregadas para montar a *sprint* que foram abordados no Quadro 1 são:

Quadro 1 – Divisão de tarefas por novas demandas de trabalho

User Story	Tasks
Eu como usuário do sistema, desejo registrar a locação de um veículo para um cliente.	Criar as estruturas de banco de dados da locação.
	Configurar mapeamento do banco de dados no ORM.
	Criar <i>design</i> da tela que cadastra a locação
	Criar funcionalidade no <i>Back-end</i>
	Criar validações de regras de negócio
	Criar validações de tela

Fonte: O Autor

A partir do princípio que essas são as funcionalidades que é necessário, foi desenvolvido um produto para melhor demonstrar essas especificações (Figura 3).

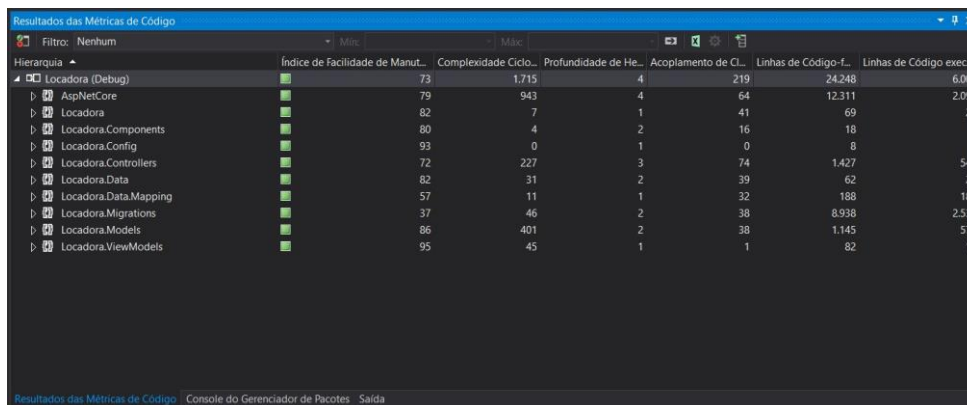
Figura 3 – Tela para retiradas de veículos em nova locação

The screenshot shows a web application interface for vehicle return. The top navigation bar is blue and contains the text 'Locadora', a date 'terça-feira, 21 de março de 2023', and the user name 'Ricardo'. A sidebar on the left lists various menu items under categories like 'Operacional', 'Garagem', 'Manutenção', and 'Financeiro'. The main content area is titled 'Retirada de veículo' and has three tabs: 'Dados Pessoais', 'Locação', and 'Documentação'. The 'Locatário' section contains a form with the following fields: CPF *, Nome *, Nome da mãe *, Estado civil *, Naturalidade *, UF de nascimento *, Profissão *, Nacionalidade *, Data de nascimento *, Cliente estrangeiro? *, Documento do estrangeiro * (with a dropdown for 'Registro Nacional de Estrangeiros (RNE)'), RG *, Órgão expedidor do RG *, and Estado emissor do RG *.

Fonte: O Autor

Outro fator importante sobre os resultados obtidos são as métricas de código (Figura 4), foi possível abstrair melhor os números envolvidos na elaboração do projeto.

Figura 4 – Métricas de código



Hierarquia	Índice de Facilidade de Manut...	Complexidade Ciclo...	Profundidade de He...	Acoplamento de CL...	Linhas de Código-f...	Linhas de Código exec...
Locadora (Debug)	73	1.715	4	219	24.248	6.003
↳ ASPNetCore	79	943	4	64	12.311	2.099
↳ Locadora	82	7	1	41	69	22
↳ Locadora.Components	80	4	2	16	18	4
↳ Locadora.Config	93	0	1	0	8	2
↳ Locadora.Controllers	72	227	3	74	1.427	549
↳ Locadora.Data	62	31	2	39	62	20
↳ Locadora.Data.Mapping	57	11	1	32	188	186
↳ Locadora.Migrations	37	46	2	38	8.938	2.530
↳ Locadora.Models	86	401	2	38	1.145	577
↳ Locadora.ViewModels	95	45	1	1	82	14

Fonte: O Autor

De forma introdutória, as métricas de software são medidas quantitativas utilizadas para avaliar o desempenho e a qualidade de um software.

Elas podem ser usadas para ajudar a entender como o software está funcionando atualmente, identificar problemas e avaliar a eficácia de soluções implementadas. Algumas métricas comuns incluem:

- Linhas de código: mede o tamanho do software em termos de número de linhas de código escritas.
- Defeitos: mede o número de erros ou problemas identificados no software.
- Tempo de resposta: mede o tempo que leva para o software executar uma tarefa específica.
- Tempo de disponibilidade: mede a porcentagem de tempo em que o software está disponível para uso.
- Usabilidade: mede a facilidade de uso do software para os usuários.
- Qualidade do código: mede a qualidade do código escrito, incluindo coisas como legibilidade e manutenibilidade.

Essas são apenas algumas das muitas métricas que podem ser usadas para avaliar o software.

É importante escolher as métricas adequadas para suas necessidades específicas e usá-las de forma consistente para ter uma compreensão clara do desempenho e da qualidade do software.

4 CONCLUSÃO

A partir dos objetivos propostos e do desenvolvimento da aplicação o *Scrum* e o *Kanban* podem ser ferramentas eficazes para gerenciar projetos, mas possuem pontos fortes diferentes e podem ser mais adequados para diferentes tipos de projetos.

O *Scrum* é mais estruturado e bem definido, tornando-o mais fácil de usar para equipes que são novas nas práticas ágeis.

O *Kanban* é mais flexível e pode ser facilmente adaptado para atender às necessidades de um projeto específico, tornando-o uma boa escolha para equipes com experiência em práticas ágeis.

No geral, tanto o *Scrum* quanto o *Kanban* podem ser ferramentas eficazes para gerenciar projetos em um ambiente ágil. A escolha de qual *framework* usar dependerá das necessidades e objetivos específicos da equipe e do projeto.

Entretanto, a utilização do *Scrum* com o *Kanban* em conjunto auxilia um desenvolvimento flexível e de adaptabilidade com representação visual e fluxo de tempo mais otimizados para projetos com umas demandas complexas, porém com uma qualidade alta e um tempo de desenvolvimento acelerado em comparação a utilizar apenas um ou outro método separadamente.

REFERENCIAS BIBLIOGRÁFICAS

CAMARGO, Robson. “Metodologia Agile: quais as mais utilizadas?”, 2018. Disponível em: <https://robsoncamargo.com.br/blog/Metodologia-Agile-quais-as-mais-utilizadas>. Acesso em 01 de janeiro de 2023.

CRUZ, Fábio. Scrum e PMBOK® Unidos no Gerenciamento de Projetos. Rio de Janeiro/RJ. Brasport, 2014.

CRUZ, Fábio. PMO Ágil – Escritório Ágil de Gerenciamento de Projetos. Rio de Janeiro/RJ. Brasport, 2016.

MASSARI, Vitor L. Agile Scrum Master no Gerenciamento Avançado de Projetos. Rio de Janeiro/RJ. Brasport, 2016.

MASSARI, Vitor L. Gerenciamento Ágil de Projetos. Rio de Janeiro/RJ. Brasport, 2014.

PMI - Project Management Institute. Um Guia do Conhecimento em Gerenciamento de Projetos - Guia PMBOK® - 6ª Edição. Pensilvânia - Project Management Institute, Inc, 2017.

SCHWABER, Ken; SUTHERLAND, Jeff. The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game. Disponível em: <https://www.scrum.org/resources/scrum-guide>, 2017. Acesso em 07 de janeiro de 2023.

SUTHERLAND, Jeff. Scrum – A Arte de Fazer o Dobro do Trabalho na Metade do Tempo. Rio de Janeiro/RJ. Leya, 2014.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. Engenharia de Requisitos: Software Orientado ao Negócio. Rio de Janeiro/RJ. Brasport, 2016.