UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUILHERME FEIER HUFF

SCRUM: UM ESTUDO DE CASO APLICADO AO DESENVOLVIMENTO DA FERRAMENTA LSTYLE

SANTA HELENA

2023

GUILHERME FEIER HUFF

SCRUM: UM ESTUDO DE CASO APLICADO AO DESENVOLVIMENTO DA FERRAMENTA LSTYLE

SCRUM: A case study applied to the development of the LStyle tool

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Drª Giani Carla Ito

SANTA HELENA

2023

GUILHERME FEIER HUFF

SCRUM: UM ESTUDO DE CASO APLICADO AO DESENVOLVIMENTO DA FERRAMENTA LSTYLE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 19/junho/2023

Giani Carla Ito - Orientadora
Doutora em Computação Aplicada
Universidade Tecnológica Federal do Paraná

Davi M. Rocha
Doutor em Engenharia Agrícola
Universidade Tecnológica Federal do Paraná

Suzan K. B. Piovezan
Mestre em Informática
Universidade Tecnológica Federal do Paraná

SANTA HELENA
2023

Dedico este trabalho a três pilares essenciais
que me acompanharam ao longo dessa
jornada: minha amada família, minha incrível
namorada e meus leais amigos.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Primeiramente, a minha professora orientadora Profª. Drª Giani Carla Ito, cuja orientação e expertise foram essenciais para o desenvolvimento deste projeto. Sua dedicação, paciência e incentivo constante foram fundamentais para o meu crescimento acadêmico e profissional. Agradeço por compartilhar seu conhecimento e por sempre me guiar na direção certa.

À minha equipe de trabalho do LStyle, que se empenhou incansavelmente na execução deste projeto. Cada membro trouxe habilidades e um comprometimento, resultando em uma colaboração de sucesso. Agradeço pela dedicação, trabalho árduo e pelo espírito de equipe exemplar. Foi uma honra trabalhar com vocês.

Agradeço também à minha família e minha namorada, que estiveram ao meu lado durante toda essa jornada acadêmica. Seu apoio incondicional, amor e encorajamento foram a força motriz que me impulsionou a superar os obstáculos e acreditar em mim mesmo. Vocês são meu porto seguro, e sou grato por todo o suporte emocional e por compartilharem comigo essa trajetória.

Por fim, gostaria de expressar minha gratidão à bolsa de fomento à pesquisa, da UTFPR, que possibilitou a realização deste trabalho. O apoio financeiro fornecido foi fundamental para a realização das atividades de pesquisa, aquisição de materiais e participação em eventos acadêmicos. Sou grato pela oportunidade concedida e pelo reconhecimento à importância da pesquisa como um agente transformador.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

A todos vocês, minha professora orientadora, minha equipe de trabalho, minha família, minha namorada e a instituição que concedeu a bolsa de fomento à pesquisa, meu profundo agradecimento. Sem o apoio de cada um de vocês, essa conquista não seria possível. Espero poder retribuir todo o suporte e confiança depositados em mim ao longo dessa jornada.

Podemos somente ver uma pequena
quantidade de um grande número de coisas
possíveis. O computador pode ver todas elas
de uma vez (TURING, 1950)

RESUMO

Este trabalho tem como objetivo desenvolver um *software* para identificação de estilos de aprendizagem, utilizando o *Scrum* como metodologia, a fim de auxiliar docentes a se adaptarem às diferentes formas de aprendizado dos alunos e proporcionar uma experiência educacional mais motivadora e eficiente. A motivação para este projeto surge das dificuldades enfrentadas pelos docentes em se adaptar aos diferentes estilos de aprendizagem dos alunos e dos desafios enfrentados pelas equipes de desenvolvimento de *software* em adotar metodologias eficientes. A metodologia adotada envolveu a utilização do *Scrum*, com a organização do time, *Backlog* e mapa de planejamento, *Sprints* e reuniões, e o desenvolvimento da ferramenta foi realizado buscando entregas rápidas e que agregassem valor, contemplando o *back-end* e *front-end*, com prototipações de telas e modelagens. Ao final do projeto, o *Scrum* foi analisado por meio de questionários aplicados ao time e aos interessados, resultando em uma ferramenta leve e democrática, com acesso à internet e pouca burocracia para uso. A implementação deste *framework* de desenvolvimento ágil se mostrou eficiente para equipes com pouca experiência, contribuindo para a melhoria do processo de desenvolvimento de *software* e, consequentemente, para a ferramenta de identificação dos estilos de aprendizagem dos alunos, que pode melhorar a experiência educacional.

Palavras-chave: ágil; estilos de aprendizagem; framework; software; rápidas.

ABSTRACT

This study aims to develop a software for identifying learning styles, using Scrum as a methodology, in order to assist educators in adapting to the different learning styles of students and providing a more motivating and efficient educational experience. The motivation for this project stems from the challenges faced by educators in adapting to the diverse learning styles of students and the difficulties encountered by software development teams in adopting efficient methodologies. The methodology employed involved the use of Scrum, with team organization, Backlog and roadmap, sprints and meetings, and the development of the tool was carried out seeking rapid deliveries that added value, encompassing back-end and front-end, with screen prototyping and modeling. At the end of the project, Scrum was analyzed through questionnaires applied to the team and stakeholders, resulting in a lightweight and democratic tool, with internet access and minimal bureaucracy for use. The implementation of this agile development framework proved efficient for teams with little experience, contributing to the improvement of the software development process and, consequently, to the tool for identifying students' learning styles, which can enhance the educational experience.

Keywords: agile; learning styles; framework; software; fast.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface*
CA	Conceituação Abstrata
CHAEA	Questionário Honey-Alonso de Estilos de Aprendizagem
DER	Diagrama Entidade Relacionamento
DSDM	*Dynamic Systems Development Method*
EA	Experimentação Ativa
EBLS	*Experience Based Learning Systems*
EC	Experiência Concreta
ECL	Ciclo Experiencial de Kolb
ELT	Teoria da Aprendizagem Experiencial
FDD	*Feature Driven Development*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
LSI	*Learning Style Inventory*
LSQ	*Learning Styles Questionnaire*
MVC	*Model-View-Controller*
MVCS	*Model-View-Controller-Service*
OR	Observação Reflexiva
PO	*Product Owner*
PU	Processo Unificado
REST	*Representational State Transfer*
SSH	*Secure Shell*

TDD	*Test-Driven Development*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
UTFPR	Universidade Tecnológica Federal do Paraná
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

O ambiente educacional, apesar de estar sempre em evolução, ainda se encontra com diversas dificuldades. Muitas vezes o processo de ensino não considera que cada indivíduo tem características diferentes, e acaba unificando a forma de ensinar, afetando o desenvolvimento e desempenho no processo de aprendizagem. Para Franzoni *et al.* (2008) ao personalizar o processo de ensino para atender às necessidades individuais dos alunos, a tecnologia tem o potencial de melhorar significativamente a experiência educacional. Novas estratégias podem ser utilizadas para levar essas individualidades em consideração e assim refletir beneficamente no processo de ensino e aprendizagem. Desta forma, o uso de ferramentas que identificam os estilos de aprendizagem otimizam as práticas pedagógicas e podem promover o melhor aproveitamento educacional.

Na mesma perspectiva, o desenvolvimento de *software* passou por várias transformações ao longo dos anos, evoluindo de um processo linear para a adoção de metodologias de trabalho que visam entregar produtos de qualidade no escopo definido. Como destacam Fowler (2001) e Schwalbe (2014), a adoção de metodologias ágeis revolucionou como as equipes de desenvolvimento entregam *software*, permitindo maior flexibilidade, colaboração e adaptação aos requisitos em constante mudança.

A escolha adequada da metodologia é crucial para garantir que o produto final atenda às expectativas. No entanto, a utilização incorreta ou a ausência de uma metodologia adequada pode levar a lacunas entre o produto desenvolvido e o produto esperado, ou até mesmo à falta de um produto final. A escolha cuidadosa da metodologia apropriada é fundamental para garantir um processo eficaz, resultando em um produto de *software* de alta qualidade. Ao adotar uma metodologia adequada, as equipes de desenvolvimento podem alinhar melhor seus esforços, gerenciar os riscos envolvidos e otimizar a colaboração entre os membros da equipe. Como destacado por Sommerville (2011), essa abordagem leva a uma maior probabilidade de sucesso na entrega do *software* conforme as expectativas dos usuários.

Neste contexto, esse trabalho visa desenvolver uma ferramenta de identificação de estilos de aprendizagem abordando duas metodologias: Honey-Alonso (1994) e David Kolb (1984), visando aplicar e analisar o uso do *Scrum* como *framework* de gerenciamento e desenvolvimento de atividades em uma equipe com pouca experiência no desenvolvimento de *software*.

1.1 Objetivos

Nesta subseção são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Objetivo geral

Desenvolver uma aplicação *Web* para identificação de estilos de aprendizagem utilizando o *Scrum* como metodologia de desenvolvimento.

1.1.2 Objetivos específicos

1. Modelar o sistema com diagramas e documentação;
2. Implementar o *back-end* e *front-end*;
3. Aplicar o *framework Scrum* durante o processo de desenvolvimento;
4. Analisar a utilização do *Scrum*;
5. Identificar os benefícios do uso do *Scrum* no projeto em desenvolvimento.

1.2 Contribuição do trabalho

O trabalho, traz como contribuição efetiva dois pontos principais. O primeiro diz respeito à ferramenta, trazendo a possibilidade do professor gerar subsídios e assim buscar a melhoria dos seus métodos de ensino, adaptando-se às necessidades de sua turma ou de cada discente.

O segundo é a avaliação referente a utilização do *Scrum* no desenvolvimento de *software* em comparação com a não utilização de nenhum método de desenvolvimento, ou um método com pouca interação. Essa análise fornece às equipes novas subsídios para aderir a uma metodologia de trabalho coerente com seu ambiente o mais cedo possível, diminuindo a quantidade de problemas e avaliando se o esforço demandado é justificado diante dos benefícios alcançados.

1.3 Justificativa

Sabendo que cada indivíduo tem seu perfil para aprender, este trabalho visa trazer subsídios ao docente para a identificação dos estilos de aprendizagem, a fim de proporcionar uma formação de conhecimento motivadora para o aluno, ocasionando um melhor desempenho durante o processo de formação. Para isso, uma aplicação *Web* que auxilia na identificação de estilos de aprendizagem será desenvolvido com base em duas metodologias de Honey-Alonso (1994) e David Kolb (1984).

No ambiente de trabalho e desenvolvimento de *software* acontece algo parecido: cada indivíduo ou empresa tem seu método de desenvolver e trabalhar, o que acaba por gerar conflitos, erros e atrasos, fazendo com que o desenvolvimento se torne um desafio para as equipes

principalmente em casos em que os integrantes são pouco experientes. Este trabalho pretende analisar o uso do *Scrum* para auxiliar novas equipes que buscam adotar um método de trabalho efetivo, baseando-se em princípios ágeis.

Deste modo é importante conhecer e compreender os estilos ligados ao desejo de facilitar a aprendizagem do estudante, assim como as metodologias de desenvolvimento de *software*, que visam melhorar e facilitar um processo que pode ser custoso.

1.4 Delimitação do trabalho

Na literatura, existem diversas metodologias que utilizam questionários para definir os estilos de aprendizagem de um indivíduo, como Modelos de Vark (1992), Honey-Alonso (1994), Felder & Silverman (1988), Dunn & Dunn (1978) e David Kolb (1984). Este trabalho limita-se à implementação de duas dessas metodologias: Honey-Alonso e David Kolb. Esses modelos foram escolhidas a partir de pesquisas, onde se averiguaram diversos pontos, como aceitação e licença de uso.

No que se refere ao desenvolvimento de software, existem diversas metodologias e *frameworks* com o intuito de melhorar o desempenho da equipe; alguns tradicionais como o modelo em cascate e o evolutivo, e outros denominados ágeis como XP (1997), *Dynamic Systems Development Method* (DSDM) (1995), *Feature Driven Development* (FDD) (1999), *Crystal* (2000), *Scrum* (1993). Este trabalho será baseado no *Scrum*. Alguns motivos para isso são: sua flexibilidade, pouca divisão na equipe, e por atuar principalmente no gerenciamento, não delimitando como serão executadas as tarefas de programação.

1.5 Estrutura do trabalho

O presente trabalho está dividido em seis seções. A Seção 1 apresenta a introdução do que será abordado ao longo desse documento, contendo os objetivos gerais e específicos do estudo, e a justificativa para a realização da pesquisa.

A Seção 2 consiste em uma revisão bibliográfica abrangendo os principais conceitos das metodologias de identificação de estilos de aprendizagem, assim como um enfoque nos modelos e processos de desenvolvimento de *software* e nos métodos ágeis. O Capítulo 3 apresenta informações sobre trabalhos relacionados desenvolvidos nos últimos anos.

A Seção 4 apresenta de forma detalhada os materiais e métodos de desenvolvimento utilizados para alcançar os resultados finais. São descritas as ferramentas, tecnologias e técnicas empregadas, bem como o fluxo de trabalho adotado e os procedimentos realizados durante o desenvolvimento do projeto.

A Seção 5 apresenta de forma detalhada os resultados obtidos com o trabalho desenvolvido. Inicialmente, são apresentados os dados relacionados a aplicação *Web* desenvolvida,

abordando suas funcionalidades, características e desempenho. Em seguida, é discorrido sobre a aplicação do *Scrum* no projeto, destacando as práticas utilizadas, os benefícios observados e os desafios enfrentados. Também, são apresentados os dados obtidos com a equipe, como a percepção dos membros em relação ao *Scrum*, a satisfação com o processo de desenvolvimento e a efetividade das reuniões e práticas adotadas.

A Seção 6 consiste em demonstrar a visão do autor sobre todo o processo de implantação da metodologia ágil e a avaliação dos resultados obtidos. São apresentadas as conclusões obtidas a partir da aplicação do *Scrum*, enfatizando os pontos positivos e os desafios enfrentados durante o projeto.

2 REFERENCIAL TEÓRICO

Nesta seção, apresentam-se as bases teóricas utilizadas no desenvolvimento deste trabalho. Inicialmente é discorrido sobre a teoria dos estilos de aprendizagem, focando nas duas metodologias já definidas David Kolb (1984) e Honey-Alonso (1994). Posteriormente, desenvolve-se o tema sobre modelos de desenvolvimento de *software*, apontando os processos tradicionais por serem de suma importância na Engenharia de *Software*, focando em desenvolver os conceitos de alguns métodos ágeis: XP (1997) e *Scrum* (1993) escolhidos por sua importância na história e aplicabilidade no processo de desenvolvimento de *software* deste trabalho.

2.1 Estilos de aprendizagem

Os estilos de aprendizagem referem-se às formas únicas pelas quais os indivíduos processam, percebem e absorvem informações. Cada pessoa tem sua maneira de abstrair e compreender informações, e isso pode ser influenciado por fatores genéticos, experiências passadas e ambiente. De acordo com Kalatzis (2008), os indivíduos têm maneiras distintas de receber e processar as informações, enquanto Bello (1990 apud KALATZIS, 2008) define os estilos de aprendizagem de forma geral como os processos pelos quais as pessoas absorvem, processam e retêm as informações.

Cada indivíduo possui um perfil e estilo único de aprendizagem, o que pode influenciar positivamente ou negativamente o processo educacional. Por isso, é fundamental que os professores identifiquem esses estilos e se adaptem às necessidades individuais de cada aluno. Quando não há um alinhamento entre a forma de ensinar do professor e a maneira como o aluno aprende, pode haver um impacto negativo na motivação e no desempenho acadêmico, levando até mesmo ao abandono do curso. Felder e Silverman (1988) prospectam que o desencontro entre a forma que o aluno aprende e a maneira que o professor ensina afeta negativamente o processo pedagógico educacional, fazendo com que o aluno se sinta entediado e consequentemente apresentando avaliações baixas.

Rocha (2021) ressalta que os estilos de aprendizagem conduzem o processo de ensino e aprendizagem para professores e alunos. Os alunos estudam de maneiras diferentes, abrindo a necessidade de se indicar métodos e estratégias de estudos mais adequadas a serem adotadas no processo de aprendizagem.

Kalatzis (2008) e Lopes (2002) apontam que a partir de toda essa diversidade de estilos de aprendizagem, pesquisadores estudam esses estilos e apresentam na literatura vários modelos que buscam identificar os perfis. Esses estudos têm o objetivo de promover benefícios à educação, e fornecer subsídios para melhorar as estratégias pedagógicas mais eficazes para os estudantes.

Já Kolb (1984) define estilos de aprendizagem como um estado duradouro e estável resultante da bagagem hereditária, das experiências passadas e das exigências do meio ambiente.

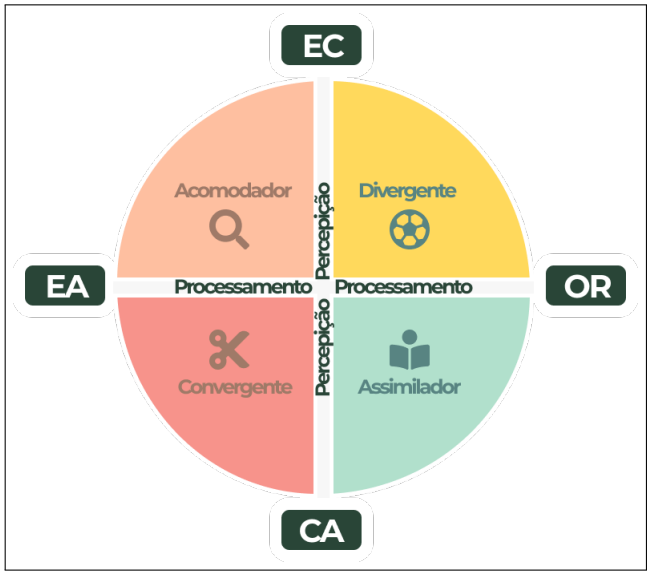
2.2 Metodologia de Kolb

David A. Kolb é um teórico da educação e fundador da *Experience Based Learning Systems* (EBLS). Doutor pela Universidade de Harvard em 1939, e criou LSI em 1971. Para Maiellaro e Pasquali (2021) e Rocha (2021), o modelo de Kolb tem como foco diagnosticar o estilo de aprendizagem de estudantes universitários.

Em 1984, David A. Kolb elaborou um novo modelo intitulado Teoria da Aprendizagem Experiencial (ELT) que, baseado em seus estudos particulares, utiliza a experiência como o centro do aprendizado em um modelo de ciclo, no qual o conhecimento é construído a partir das práticas vividas.

De acordo com Kolb (1984), a aprendizagem eficaz ocorre em ciclos, em uma sucessão de quatro estágios, também conhecidos como modos de aprendizagem, sendo: Experiência Concreta (EC) (sentir, vivenciar, experimentar); Observação Reflexiva (OR) (observar, refletir); Conceituação Abstrata (CA) (pensar); e Experimentação Ativa (EA) (fazer, agir). Esses estágios são dispostos em dois eixos: percepção e processamento. A combinação de dois estágios em eixos diferentes geram os quatro estilos de aprendizagem, conforme proposto por Kolb (1984). Na Figura 1 é demonstrado essa combinação.

Figura 1 – Representação do ciclo de Kolb



Fonte: Adaptado de Rocha (2021).

O resultado da combinação dos estágios dois a dois, sendo um do eixo da percepção (EC e CA) e outro do eixo do processamento (EA e OR), possibilitam a identificação do estilo

de aprendizagem predominante no indivíduo. A seguir, no Quadro 1 é apresentado os quatro estilos segundo Kolb (1984).

Quadro 1 – Estilos de aprendizagem segundo Kolb

Estilo	Modos	Descrição
Divergente	OR - EC	Aborda principalmente a criatividade do indivíduo no qual emprega sua imaginação gerando assim novas ideias, dúvidas e discussões, tornando o ato de buscar várias maneiras de aprender prazeroso. São pessoas reflexivas e criativas.
Assimilador	CA - OR	Aborda principalmente modelos teóricos e raciocínio, não focando apenas no uso prático da teoria. São pessoas mais teóricas, conceituais.
Convergente	EA - CA	Aborda uma metodologia prática para resolver problemas usando como ferramenta o raciocínio dedutivo. São pessoas estruturais, sequenciais.
Acomodador	EC - EA	Aborda uma metodologia de assumir riscos para resolver o problema, aplicando a intuição no processo de aprendizagem com tentativas e erros. São pessoas mais práticas, vivenciais.

Fonte: Adaptado de Kolb (1984) e Rocha (2021).

Jesus, Carvalho e Silva (2019) e Rocha (2021) destacam que os estilos de aprendizagem propostos por Kolb não têm como objetivo rotular a maneira como cada pessoa aprende de forma estagnada, mas sim identificar o estilo predominante de aprendizagem em cada indivíduo e, a partir disso, desenvolver os outros estilos. Essa classificação também auxilia o professor a buscar novas estratégias de ensino, possibilitando uma melhora significativa no processo de aprendizagem dos alunos.

Para a identificação dos estilos de aprendizagem, Kolb e Kolb (2005) introduzem o instrumento de identificação de estilos de aprendizagem predominante, o LSI. Kolb e Kolb (2013) destacam seis versões publicadas em 40 anos. Conforme Kolb e Kolb (2005), o LSI foi criado com dois objetivos. O primeiro é o uso como ferramenta educacional para aumentar o entendimento dos indivíduos sobre o processo de aprendizagem com a experiência e sua abordagem individual única para a aprendizagem, aumentando a consciência do indivíduo sobre como ele aprende. Já o segundo objetivo foi comparado a uma ferramenta de pesquisa para investigar o ELT.

A segunda versão do LSI (1985), aplicada neste projeto, conta com as quatro formas de aprender do Ciclo Experiencial de Kolb (ECL) onde cada forma possui com algumas perguntas do questionário, formando a base para o cálculo de estilo de aprendizagem.

Segundo Kolb e Kolb (2005), o questionário apresenta 12 sentenças a serem respondidas, cada uma tendo 4 respostas, em que cada resposta contempla valores de um a quatro. O respondente deve indicar um número diferente para cada alternativa, sendo que o número 4 representa a característica com a qual o indivíduo mais se identifica, decrescendo até 1 para a característica que o indivíduo menos se identifica.

No Quadro 2 é demonstrado as 12 questões do LSI a serem aplicadas no teste e suas respectivas respostas.

Quadro 2 – Questionário de Kolb LSI-2

Pergunta	Opções			
1. Quando aprendo...	Gosto de lidar com sentimentos	Gosto de pensar sobre ideias	Gosto de produzir	Gosto de observar e escutar
2. Aprendo melhor quando...	Escuto e observo com atenção	Confio em pensamentos lógicos	Confio em minha intuição e pensamentos	Trabalho duro para fazer as tarefas
3. Quando estou aprendendo...	Sou racional	Sou responsável no que faço	Sou calado e reservado	Tenho fortes sentimentos e reações
4. Eu aprendo...	Sentindo	Fazendo	Observando	Pensando
5. Quando aprendo...	Me abro a novas experiências	Observo todos os lados do assunto	Gosto de analisar as coisas, iria dividir em partes	Gosto de experimentar e tentar fazer as coisas
6. Quando estou aprendendo...	Sou observador	Sou ativo	Sou intuitivo	Sou lógico
7. Aprendo melhor...	Observando	Se relacionando com outras pessoas	Com teorias racionais	Com oportunidade de provar e praticar
8. Quando aprendo...	Gosto de ver os resultados já trabalhados	Gosto de ideias e teorias	Penso antes de agir	Sinto-me envolvido no assunto
9. Aprendo melhor quando...	Confio em minhas observações	Confio em meus sentimentos	Experimento por minha conta	Confio em minhas ideias
10. Quando estou aprendendo...	Sou reservado	Sou flexível	Sou responsável	Sou raciona
11. Quando aprendo...	Me envolvo no assunto	Gosto de observar	Avalio as coisas	Gosto de ser ativo
12. Aprendo melhor quando...	Analiso as ideias	Sou receptivo e mente aberta	Sou cuidadoso	Sou prático

Fonte: Adaptado de Rocha (2021).

No Quadro 3 é simplificado cada um dos estágios de aprendizagem e as respectivas perguntas que os contemplam.

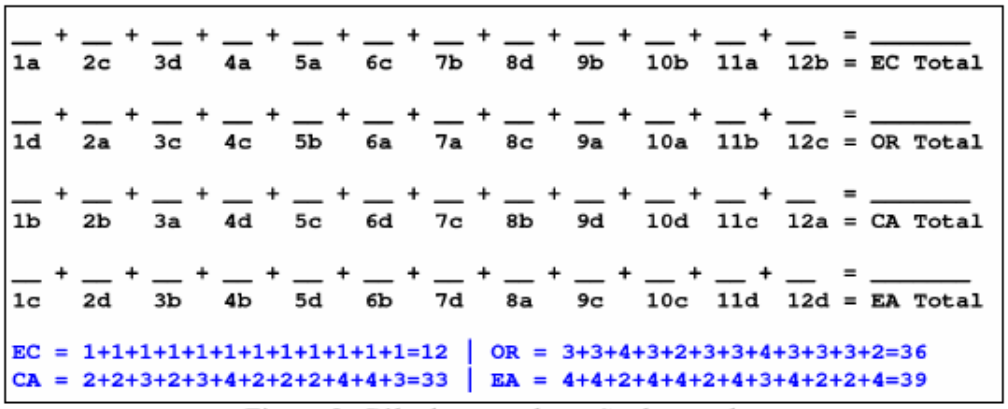
Quadro 3 – Mapeamento das questões LSI

Estágio / Modo	Perguntas
Experiencia Concreta (EC)	1a - 2c – 3d - 4a - 5a - 6c – 7b - 8d - 9b – 10b - 11a - 12b
Observação Reflexiva (OR)	1d - 2a - 3c - 4c - 5b - 6a - 7a - 8c - 9a - 10a – 11b - 12c
Conceituação Abstrata (CA)	1b – 2b - 3a - 4d - 5c - 6d – 7c – 8b – 9d – 10d – 11c - 12a
Experimentação Ativa (EA)	1c – 2d – 3b – 4b – 5d – 6b – 7d – 8a – 9c – 10c – 11d - 12d

Fonte: Adaptado de Rocha (2021).

Com as respostas do questionário, deve-se distribuir a pontuação de cada pergunta, realizando a soma para cada uma das características representadas nos Quadro 2 e 3, encontrando-se os valores para cada modo. A Figura 2 apresenta como funciona esse cálculo: em preto está a fórmula do somatório e em azul um exemplo de resultados.

Figura 2 – Cálculo para obtenção dos modos



Fonte: Adaptado de Rocha (2021).

Conforme Dantas (2011), com o resultado do somatório é possível visualizar o nível de desenvolvimento alcançado pelo sujeito em cada um dos quatro modos de aprendizagem. Para identificar o estilo de aprendizagem predominante, verificam-se os maiores modos de cada eixo (CA – EC) e (EA – OR). O estilo do aluno está no quadrante entre os dois eixos, conforme Kolb e Kolb (2005). Assim, Dantas (2011) faz uma alusão a uma função de duas variáveis, estes valores são dispostos em um gráfico para identificar o seu estilo de aprendizagem predominante através do quadrante no qual a interseção das retas acontece. Esse gráfico é uma adaptação da Figura 1.

Diante dessa abordagem, cada opção de uma pergunta se enquadra em uma habilidade do ciclo. Kolb (1984) apresenta duas formas de perceber e processar o conhecimento, sendo de lados opostos de cada habilidade, combinando uma percepção e um processamento, que originam o estilo predominante do indivíduo.

2.3 Modelo de Honey-Alonso

Segundo Alonso, Gallego e Honey (1994) esse modelo, criado pelo psicólogo Peter Honey e pela doutora em educação Catalina M. Alonso, tem sua origem a partir de ideias do modelo proposto por Kolb em 1976 e Honey e Mumford em 1992. Segundo Alonso, Gallego e Honey (1994), o *Learning Styles Questionnaire* (LSQ) foi organizado para profissionais de empresas do Reino Unido, ao adaptar o questionário Catalina Alonso, trazendo o foco social e educacional.

Segundo Keefe (1982 apud ALONSO; GALLEGO; HONEY, 1994), os estilos de aprendizagem são um composto de características fisiológicas, afetivas, cognitivas e fatores psicológicos que servem como indicadores considerados estáveis, observando o modo como os alunos percebem e interagem no ambiente pedagógico de aprendizagem. Rocha (2021) cita que o método traça a personalidade de cada indivíduo, inferindo um questionário em forma de frases para

identificar e descrever qual a habilidade do indivíduo no meio escolar, analisando os aspectos comportamentais na aquisição do conhecimento.

Rocha (2021) e Gambús, Araújo e Portilho (2023) afirmam que o modelo emprega o questionário catalogado como Questionário Honey-Alonso de Estilos de Aprendizagem (CHAEA). Nesse questionário há 80 questões, das quais 20 são relativas a cada estilo de aprendizagem. As questões contam com duas opções: sim e não. O questionário identifica quatro tipos de estilos de aprendizagem. Para a definição do estilo de aprendizagem, a metodologia aplica um mapeamento das questões, resultantes do estilo identificado a partir do questionário, sendo eles ativo, reflexivo, teórico e pragmático. Lembrando que para cada estilo, a pontuação máxima é 20 pontos, o CHAEA foi projetado para realizar uma soma simples entre as respostas das questões. No Quadro 4 é apresentado uma descrição de cada estilo.

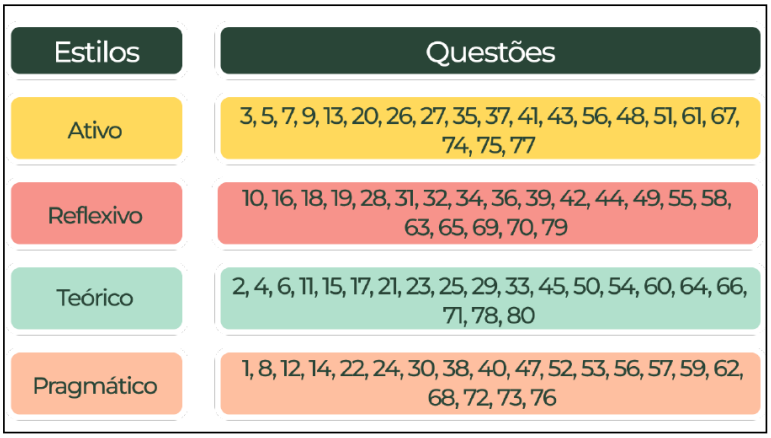
Quadro 4 – Estilos de aprendizagem segundo Honey-Alonso

Estilos	Descrição
Ativo	Aprende melhor fazendo coisas, gosta de experimentar e explorar, é prático e gosta de desafios.
Reflexivo	Aprende melhor observando e refletindo sobre experiências, gosta de pensar antes de agir, é analítico e cuidadoso.
Teórico	Aprende melhor através da lógica e da teoria, gosta de analisar e sintetizar informações, é sistemático e objetivo.
Pragmático	Aprende melhor aplicando as coisas na prática, gosta de experimentar e testar ideias, é orientado para resultados e soluções.

Fonte: Adaptado de Martins (2022).

O questionário, inicialmente em espanhol, foi traduzido e adaptado para a língua portuguesa pela doutora e psicopedagoga Evelise Maria Labatut Portilho em 2003. Rocha (2021) comenta que Portilho (s.d.) trouxe e adaptou o questionário CHAEA para o Brasil e posteriormente disseminou a teoria no país. A Figura 3 contém o mapeamento das questões, presentes no Anexo A, que correspondem a cada estilo de aprendizagem segundo Portilho (s.d.).

Figura 3 – Cálculo para obtenção dos estilos Honey-Alonso



Fonte: Adaptado de Portilho (s.d.).

No entendimento de Alonso, Gallego e Honey (1994), o questionário não possui respostas certas ou erradas, ou seja, o que torna a identificação mais precisa é a honestidade de quem está a preenchê-lo. Seguindo as regras estabelecidas pelo autor do questionário, é possível identificar o estilo principal de aprendizagem de cada pessoa. Como em Kolb (1984) é aconselhado o indivíduo passar por todos os estilos propostos para identificar seu próprio estilo predominante. Além disso, é importante ressaltar que o questionário não deve ser utilizado como uma ferramenta única e definitiva na identificação do estilo de aprendizagem de um indivíduo. Ele deve ser complementado por outras observações e avaliações do aluno em diferentes situações de aprendizagem.

2.3.1 Aplicação *Web LStyle*

De acordo com Huff *et al.* (2022), a ferramenta *LStyle* demonstra a capacidade de identificar os diversos estilos de aprendizagem, proporcionando uma experiência de aprendizagem aprimorada. A aplicação *Web* gratuita, com interface agradável visa trazer para quem a utiliza uma forma simples de responder às perguntas e receber seus resultados.

Para Brasil, Huff e Ito (2022) a aplicação *Web LStyle* tem como proposta conectar alunos e professores, permitindo que os docentes desenvolvam metodologias de ensino personalizadas, adaptadas aos estilos de aprendizagem individuais dos estudantes.

2.4 Processos de desenvolvimento de *software*

Nesta seção serão apresentados os processos de desenvolvimento de *software*, iniciando com uma descrição conceitual do tema. Em seguida, serão contextualizados os modelos tradicionais e ágeis, abordando suas respectivas características. O objetivo é proporcionar ao leitor uma compreensão mais aprofundada dos processos de desenvolvimento de *software* e das diferentes abordagens existentes para sua implementação, permitindo uma análise crítica e uma tomada de decisão embasada na escolha do modelo mais adequado para cada contexto de desenvolvimento.

Segundo Pressman e Maxim (2021), um processo pode ser definido como um conjunto de atividades de trabalho e tarefas realizadas quando algum artefato de *software* deve ser criado. Cada uma dessas atividades se aloca em uma metodologia ou modelo que determina sua relação com o processo e uma com a outra.

Para Hirama (2011), os métodos de desenvolvimento instituem à equipe de projeto uma diretriz do que deve ser feito para atender aos objetivos do *software*. Esses processos definem quais são as atividades que permitem obter o produto final.

2.4.1 Metodologias Tradicionais

De acordo com Bassi Filho (2008) a partir do final da década de 60, surgiram abordagens que buscavam aplicar técnicas de engenharia no desenvolvimento de sistemas de *software*. O objetivo era estabelecer processos que pudessem ser descritos e replicados, visando a criação de sistemas de forma mais estruturada. Essas abordagens adotavam uma abordagem sequencial, dividindo o desenvolvimento de *software* em fases distintas, com o suporte de ferramentas e uma documentação detalhada de todos os artefatos produzidos ao longo do processo. Segundo Soares (2004), as metodologias tradicionais também são conhecidas como orientadas a documentos.

Segundo Pressman e Maxim (2021), há cinco etapas em uma metodologia tradicional genérica: comunicação, planejamento, modelagem, construção e entrega. Como parte do processo, a equipe precisa produzir diversos tipos de itens até chegar ao produto final. Alguns desses processos são diagramas *Unified Modeling Language* (UML), glossários, cenário de uso, especificações, modelagens, solicitações, planos de trabalho, formulários, componentes, testes, entre outros. Esses elementos são chamados de artefatos.

No modelo tradicional, a documentação é de extrema importância. Torres (2014) indica que após a fase de planejamento, a documentação é criada e não é mais alterada, passando então para a fase de desenvolvimento. Bassi Filho (2008) destaca que com uma documentação detalhada é possível que inúmeras pessoas trabalhem sem compartilhar o mesmo espaço físico, ou estarem presentes no mesmo horário.

Apesar disso, Pressman e Maxim (2021) ressaltam que a documentação não substitui a comunicação verbal e as interações entre as pessoas.

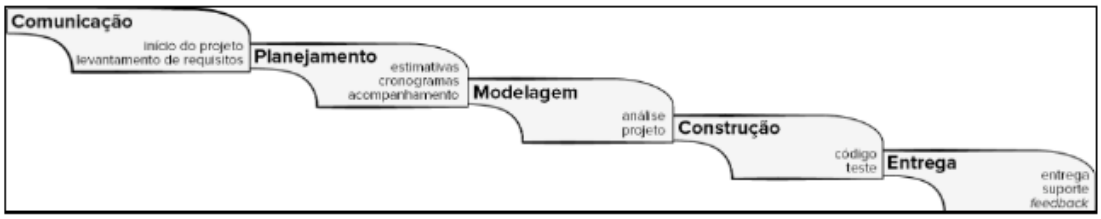
2.4.1.1 Cascata

O processo em cascata, proposto na década de 1970, sugere uma abordagem sequencial para o desenvolvimento de *software*. Hirama (2011) afirma que nesse modelo, para prosseguir para uma próxima fase, a fase anterior deve estar finalizada, verificada e aprovada.

Segundo Hirama (2011), o processo segue linearmente as etapas de: engenharia do sistema, análise de requisitos, projeto, geração de código, testes e manutenção. Já Pressman e Maxim (2021) reduzem esse processo a cinco etapas, que contemplam as etapas citadas anteriormente, sendo elas comunicação, planejamento, modelagem, construção, entrega - Figura 4.

Segundo Pressman e Maxim (2021), na fase de comunicação o cliente é contatado pela equipe de desenvolvimento, a fim de entender as necessidades e requisitos do *software* a ser desenvolvido. No planejamento a equipe de desenvolvimento define os objetivos, as metas e as restrições do projeto, além de elaborar o plano de desenvolvimento. A modelagem é a etapa onde são elaborados os modelos conceituais, funcionais e de dados do sistema, a fim de se obter uma visão clara e completa dos requisitos e do escopo do projeto. A construção aborda o

Figura 4 – Modelo em cascata



Fonte: Pressman e Maxim (2021).

desenvolvimento, onde é realizada a codificação do *software* com base nos modelos, e também abrange os testes no *software* a fim de verificar se ele atende aos requisitos e expectativas do cliente. Na entrega, o *software* é entregue ao cliente para uso e manutenção.

2.4.1.2 Espiral

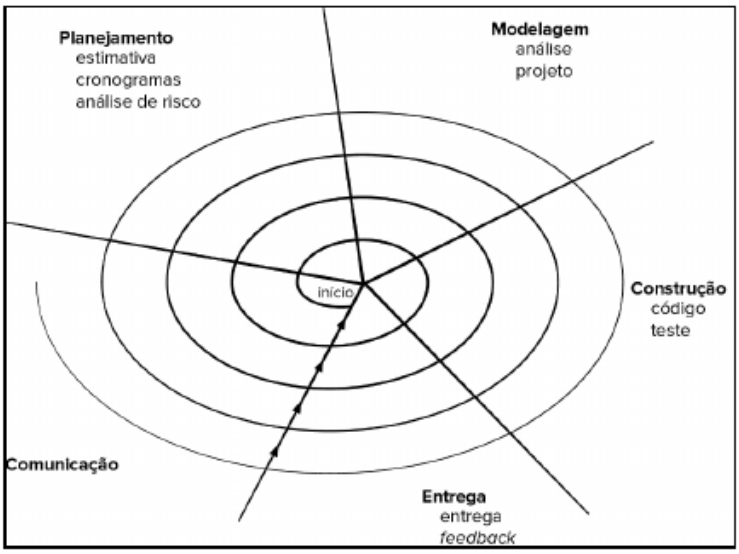
O processo espiral, também conhecido como evolutivo, foi pensado para resolver alguns problemas do modelo em cascata. Este processo determina que o *software* evolui ao longo do tempo. Pressman e Maxim (2021, p. 29) complementam que “conforme o desenvolvimento do projeto avança, os requisitos do negócio e do produto frequentemente mudam, tornando inadequado seguir um planejamento em linha reta de um produto final”.

Segundo Hirama (2011), os processos evolutivos iniciam com um subconjunto de requisitos do produto desenvolvido e incrementado gradualmente. Cada novo incremento é entregue ao cliente. Pressman e Maxim (2021) afirmam que o modelo espiral tem potencial para o desenvolvimento rápido de versões cada vez mais completas do produto. Para Torres (2014), o modelo espiral tem maior comunicação com o cliente, possibilitando minimizar riscos a cada ciclo de vida.

O processo espiral é dividido em um conjunto de atividades definidas pela equipe de engenharia de *software*, porém, normalmente se utiliza as etapas do processo em cascata. Uma nova etapa é adicionada na fase de avaliação do cliente, o cliente avalia o *software* e fornece *feedback*. Com base nesse *feedback*, o modelo retorna à fase de planejamento e inicia uma nova iteração, segundo Pressman e Maxim (2021). A Figura 5 é uma representação do processo espiral onde as atividades evoluem do centro para fora, no sentido horário.

Hirama (2011) e Pressman e Maxim (2021) concordam que, diferente de outros modelos de processo, o modelo espiral pode ser adaptado para ser aplicado durante a vida do *software*. O modelo espiral é uma abordagem realista para o desenvolvimento de *software* em larga escala. Esse processo demanda uma atenção direta sobre os riscos técnicos e gerenciais em todas as etapas do projeto, reduzindo os riscos e minimizando futuros problemas maiores.

Figura 5 – Modelo espiral típico

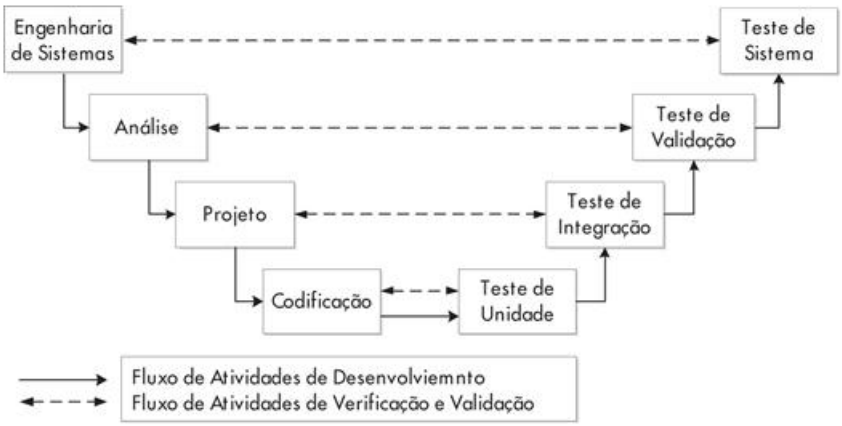


Fonte: Pressman e Maxim (2021).

2.4.1.3 Modelo V

Para Hirama (2011), o modelo em V é semelhante ao modelo cascata, porém se diferencia ao enfatizar o planejamento dos testes durante todas as fases de desenvolvimento. Cada tipo de teste é executado de acordo com um plano de testes, com o objetivo de verificar e validar se as especificações definidas estão sendo atendidas. Na Figura 6 está representado o modelo V, o lado esquerdo dá ênfase à decomposição de requisitos e o lado direito à integração do sistema.

Figura 6 – Modelo V



Fonte: Hirama (2011).

Conforme afirmado por Maschietto *et al.* (2020), o modelo V enfatiza a importância dos testes em diversos níveis ao longo do processo de desenvolvimento. De acordo com Hirama

(2011), as fases representadas no lado esquerdo da Figura 6 possuem objetivos similares aos do modelo em cascata, enquanto as fases do lado direito enfatizam os diferentes tipos de testes necessários para verificar e validar o cumprimento das especificações.

Segundo Hirama (2011) esse modelo é reconhecido por suas características distintivas que promovem a eficácia dos testes no processo de desenvolvimento. Essas características incluem a divisão clara entre as atividades de desenvolvimento e as de verificação e validação, permitindo uma abordagem mais estruturada. Além disso, o modelo enfatiza a clareza dos objetivos de verificação e validação, garantindo que o *software* esteja sendo construído corretamente e conforme as especificações acordadas com o cliente. Por fim, o modelo V também facilita o planejamento dos testes, permitindo uma melhor alocação de recursos e uma especificação mais precisa dos tipos de testes necessários.

2.4.1.4 Processo unificado

Conforme destacam Pressman e Maxim (2021), esse processo tenta busca aproveitar os recursos e características mais relevantes dos modelos tradicionais, ao mesmo tempo, em que visa incorporar princípios do desenvolvimento ágil de *software*. Reconhecendo a relevância da comunicação efetiva com o cliente e a necessidade de métodos simplificados para expressar a visão do cliente sobre o sistema.

Hirama (2011) destaca que o Processo Unificado (PU) é um modelo híbrido de processo que integra as perspectivas dinâmicas (fases) e estáticas (fluxos de trabalho) em um único processo. Ele é orientado pelos Casos de Uso, que são utilizados para especificar as funcionalidades do *software* sob a perspectiva dos usuários. Além disso, o PU tem como foco central a arquitetura do sistema, fornecendo uma visão detalhada dos elementos estruturais e comportamentais desde o início do desenvolvimento. O produto é desenvolvido de forma iterativa e incremental, passando por fases de concepção, elaboração, construção e transição, onde são definidos o escopo, a arquitetura, a implementação e os testes do *software*, respectivamente.

Para (PRESSMAN; MAXIM, 2021) o PU consiste em quatro fases distintas. Na fase de concepção, ocorre a comunicação com o cliente e o planejamento, onde são descritos os requisitos de negócio fundamentais em casos de uso preliminares. Já a fase de elaboração envolve atividades de planejamento e modelagem, refinando os casos de uso e estabelecendo uma base de arquitetura com diferentes visões do *software*. A fase de construção consiste na implementação dos recursos e funções necessários, realizando testes de unidade e integração. Na fase de transição, o *software* é entregue aos usuários para testes beta, coletando *feedback* e realizando ajustes. A fase de produção monitora o uso contínuo do *software*, fornece suporte ao ambiente operacional e lida com relatórios de defeitos e solicitações de mudanças.

2.4.2 Metodologias Ágeis

De acordo com Beck *et al.* (2001), em 2001, 17 líderes de projetos de *software* se se reuniram para discutir formas de trabalho com objetivo de chegar a uma nova metodologia de produção de *software* que pudesse ser usada, substituindo os modelos tradicionais de desenvolvimento. Esse grupo publicou o Manifesto Ágil.

Bassi Filho (2008) sintetiza que esses métodos concentram-se nos fatores humanos e em entregar o produto ao cliente, e não nas burocracias dos processos.

Beck *et al.* (2001) relatam que as ideias fundamentais que guiam o desenvolvimento ágil são:

- **Indivíduos e iterações** são mais importantes do que processos e ferramentas.
- **Software funcionando** é mais importante do que documentação completa.
- **Colaboração com o cliente** é mais importante do que negociação de contratos.
- **Adaptação a mudanças** é mais importante do que seguir o plano inicial.

Bassi Filho (2008) destaca que os métodos ágeis surgiram com a proposta de obter resultados práticos em um período menor do que era comum na indústria de *software*. Ao contrário dos modelos tradicionais, os métodos ágeis priorizam o foco no produto em vez do processo. Para atingir esse objetivo, os métodos ágeis tiveram que adotar uma abordagem que dispensasse ou modificasse as etapas do processo e a forma como os profissionais envolvidos no desenvolvimento de *software* realizavam suas atividades. Contudo, as abordagens ágeis eram frequentemente vistas como polêmicas devido à sua natureza disruptiva em relação aos métodos tradicionais de desenvolvimento de *software*, que consideram certas características como essenciais. No entanto, a ênfase das metodologias ágeis no produto final e nos interesses de negócio tem despertado interesse em uma parcela crescente da comunidade de desenvolvimento.

Para Pressman e Maxim (2021), as metodologias ágeis não são indicadas para todos os projetos, produtos, pessoas e situações. Essas metodologias seguem as práticas de engenharia de *software* confiáveis e podem ser aplicadas como uma filosofia geral para todos os trabalhos de *software*.

2.4.2.1 Extreme Programing (XP)

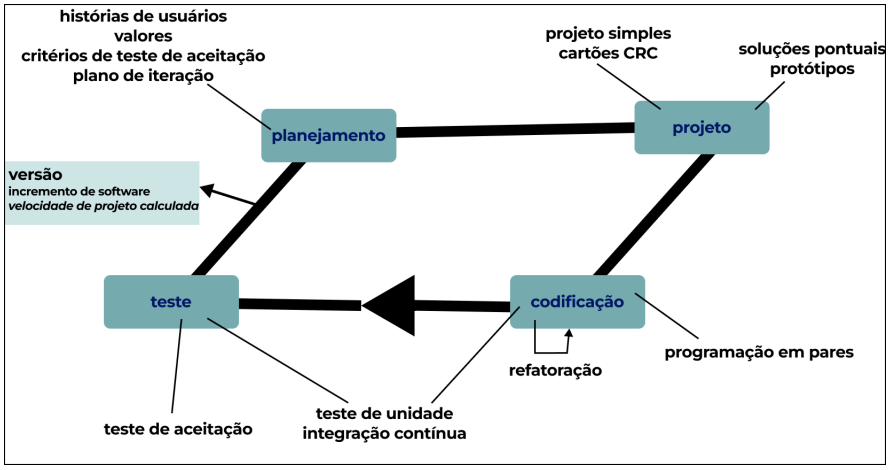
Conforme Sommerville (2011) e Fowler (2013), o método XP foi desenvolvido no final dos anos 90 para impulsionar práticas como o desenvolvimento iterativo a níveis “extremos”. Segundo Bassi Filho (2008), com Martin Fowler e Ron Jeffries em sua equipe, Beck a tornou altamente produtiva e entregou um sistema de qualidade em menos tempo que as demais tentativas.

Diferente do *Scrum* que não delimita como deve ser o desenvolvimento, deixando a equipe mais livre nesse quesito, Pressman e Maxim (2021, p. 46) afirmam que a XP envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento, projeto, codificação e testes". Já Sbrocco e Macedo (2012) cita que o XP se trata de uma metodologia flexível que pode ser utilizada em diversos projetos.

Sbrocco e Macedo (2012) afirmam que essa metodologia visa atender as necessidades do cliente com qualidade e de forma mais simples. Utiliza um modelo incremental, ou seja, cronologicamente novas melhorias são implementadas.

Para Torres (2014), a abordagem XP é de maior risco e exige maior habilidade, conhecimento e experiência para se obter sucesso em sua execução. Para o autor, o processo XP segue a seguinte estrutura: requisitos do projeto, histórias, caso de testes, atividades e finalização das entregas (realizado, completo, pronto). Na Figura 7 o processo XP é ilustrado, destacando alguns conceitos e tarefas-chave associadas a cada uma das atividades metodológicas.

Figura 7 – O fluxo geral do processo XP



Fonte: Adaptado de Pressman e Maxim (2016).

Bassi Filho (2008) destaca que a primeira versão do XP foi baseada em quatro valores, mas, na sua segunda versão foi adicionado o do “Respeito” para ressaltar o lado humano presente no desenvolvimento de *software*. Os cinco valores são:

- ***Feedback*:** os retornos constantes em ciclos curtos dão ao processo capacidade para garantir ajustes rápidos e precisos no desenvolvimento.
- **Comunicação:** o XP propõe reuniões diárias entre equipe, assim, pontos críticos podem ser tratados antes que se tornem problemas mais graves, e boa comunicação com o cliente para que os desenvolvedores possam compreender precisamente o que o cliente necessita, fazendo com que ele fique ciente das limitações.
- **Simplicidade:** com intuito de evitar desperdício, reduzir custo e tempo, manter as funcionalidades fáceis de se utilizar, o XP prioriza o que é absolutamente necessário para o projeto.

- **Coragem:** estar aberto a mudanças, encarar falhas, aceitar os feedbacks, propor melhorias e saber dizer não quando necessário.
- **Respeito:** peça fundamental para que os demais valores corroborem. Além disso, o respeito para com a equipe é assumir somente responsabilidades que conseguirá cumprir.

Teles (2017) destaca que a equipe deve ser relativamente pequena, com no máximo doze integrantes, dividida em cinco papéis: gerente de projeto, *coach*, analista de teste, redator técnico e desenvolvedores.

- **Gerente de projeto:** o Responsável pelo assunto administrativo. Lidera a equipe em questões que não são ligadas diretamente às atividades diárias. Administra o relacionamento com o cliente.
- ***Coach*:** responsável técnico pelo projeto. Normalmente é o desenvolvedor mais experiente. Assegura o bom funcionamento do processo e busca formas de melhorá-lo continuamente.
- **Analista de teste:** ajuda o cliente a escrever os testes de aceitação. Identificar eventuais defeitos do sistema.
- **Redator técnico:** ajuda a equipe a documentar o sistema.
- **Desenvolvedor:** analisa, projeta e codifica o sistema.

Sbrocco e Macedo (2012) e Pressman e Maxim (2016) destacam que o XP não conta somente com os seus valores, mas também com doze práticas que traduzem os valores em ações para o dia a dia:

- **Cliente presente:** o cliente deve conduzir o desenvolvimento a partir do *feedback*. Para que isso aconteça a comunicação é essencial.
- **Jogo do Planejamento:** um encontro no início da semana, com a equipe e o cliente, para elencar as funcionalidades e tornar clara as expectativas.
- ***Stand Up Meeting*:** a equipe de desenvolvimento se reúne a cada dia para avaliar o trabalho que foi executado e priorizar o que será implementado no próximo dia.
- **Programação em pares:** desenvolvedores trabalham em pares, permitindo que o código seja revisado enquanto é construído.
- **Desenvolvimento guiado pelos testes:** testes são desenvolvidos antes de codificar, guiando o desenvolvimento para satisfazer esses testes.

- **Refatoração:** possibilita a melhoria contínua do projeto, ou seja, é a alteração de código sem afetar o funcionamento para tornar o *software* mais simples de ser manipulado.
- **Código Coletivo:** qualquer desenvolvedor tem acesso a todas as partes do código para fazer alterações importantes, o que fornece agilidade ao processo e cria mais mecanismos de revisão.
- **Código padronizado:** a equipe estabelece padrões de codificação.
- **Metáfora:** tem em vista facilitar a comunicação com o cliente, entendendo qual a realidade dele.
- **Ritmo sustentável:** manter um ritmo de trabalho com qualidade, onde os desenvolvedores devem permanecer atentos e dispostos.
- ***Design* simples:** a equipe deve optar por um código que seja suficiente para atender a necessidade da funcionalidade que está sendo implementada, obtendo *feedback* rapidamente.
- **Integração contínua:** a integração contínua permite saber o *status* real da programação. Ela é feita diversas vezes ao dia pelos pares, reduzindo a quantidade de erros.
- ***Releases* curtos:** visa produzir um conjunto reduzido de funcionalidades para iniciar a produção, assim, o cliente pode usar o *software* no seu dia a dia.

Bassi Filho (2008) deixa claro que aplicar todas as práticas, sem considerar seus princípios e valores, não é uma abordagem efetiva. Segui-las sem pensar vai à contramão da própria filosofia que o XP segue. Esse ponto levou à criação de uma nova prática chamada “quando não funcionar ajuste”, essa que sugere a adaptação das práticas consoantes ao ambiente.

2.4.2.2 Scrum

O termo “*scrum*” se originou no jogo de rugby, e diz respeito à forma do time recolocar a bola em jogo. Na metodologia, a equipe de desenvolvimento trabalha unida visando entregar o *software* funcional de alta qualidade. Neste modelo, desenvolvido por Ken Schwaber e Jeff Sutherland na década de 1990, a equipe se compromete com um objetivo e autonomia para definir a tática para alcançá-lo, segundo Bassi Filho (2008).

Para o scrumstudy (2022), um projeto *Scrum* é caracterizado por um esforço de colaboração para criar um novo produto, serviço ou resultado, conforme definido na declaração da visão do projeto.

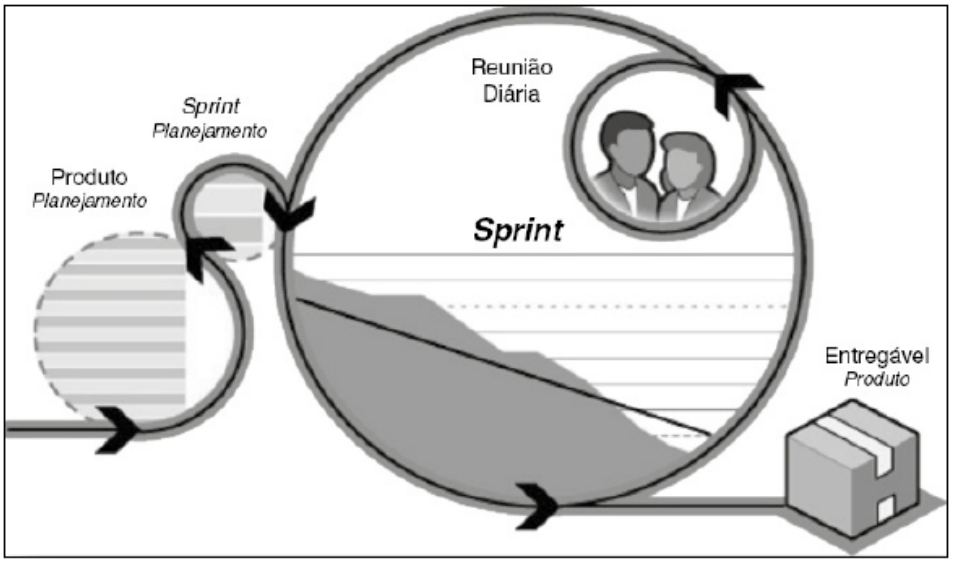
De acordo com Kniberg (2007, p. 6) “*Scrum* não é uma metodologia, é um *framework*. O que significa que *Scrum* não vai te dizer exatamente o que fazer”. Sabendo disso, o *Scrum*

sugere alguns passos gerais, mas sem delimitar nenhuma das etapas, deixando a equipe livre para trabalhar como preferir, permitindo ser adaptado para uma situação específica.

O scrumstudy (2022) define que *Scrum* é um *framework* ágil com estrutura adaptável, iterativa, rápida, flexível e eficaz projetada para fornecer valor significativo rapidamente em todo o projeto. O *Scrum* garante transparência na comunicação e cria um ambiente de responsabilidade coletiva e progresso contínuo.

Para Pressman e Maxim (2021) e scrumstudy (2022), uma das características que tornam o *Scrum* uma metodologia de desenvolvimento de *software* eficaz é a sua abordagem de equipe multifuncional e auto-organizada. Essa metodologia permite que o trabalho seja dividido em ciclos curtos e concentrados, conhecidos como *Sprints*, que permitem que o *Time Scrum* foque em um conjunto específico de tarefas em um período determinado de tempo - Figura 8.

Figura 8 – O fluxo geral do processo *Scrum*



Fonte: Torres (2014).

Conforme scrumstudy (2022), o ciclo *Scrum* representado pela Figura 8 se dá em alguns passos chamados de eventos, os quais acontecem durante o fluxo do *Scrum*. Esse fluxo que se inicia com uma reunião de *Stakeholders*, na qual a Visão do Projeto é criada. O *Product Owner* (PO) então desenvolve um *Backlog* Priorizado do Produto que contém uma lista priorizada de requisitos de negócios e projetos escritos na forma de Histórias de Usuário. Pressman e Maxim (2021) destacam que antes da reunião de planejamento da *Sprint*, a equipe juntamente com o PO projetam o *Backlog* do Produto, onde os itens são ordenados por sua importância para o negócio e complexidade das tarefas de engenharia de *software*.

Cada *Sprint* começa com uma Reunião de Planejamento da *Sprint*. Nessa reunião as Histórias de Usuários com maior prioridade são consideradas para inclusão no *Backlog da Sprint* conforme scrumstudy (2022). Torres (2014) completa que na Reunião de Planejamento da *Sprint* o PO junto a equipe avalia o *Backlog* e seleciona os itens com maior prioridade ou valor agregado para a *Sprint*. Esses itens são adicionados ao *Sprint Backlog*, que define a meta

para o *Sprint* e o trabalho a ser realizado. É importante ressaltar que a meta não deve ser alterada durante seu decorrer, garantindo que o time possa se concentrar no desenvolvimento dos itens selecionados e atingir os objetivos definidos para a *Sprint*.

Schwaber e Sutherland (2020) citam que as *Sprints* são eventos de duração fixa, geralmente de uma a quatro semanas, e ao terminar uma *Sprint* outra deve começar imediatamente sem interromper o fluxo de trabalho. Na *Sprint*, o Time *Scrum* trabalha para criar pequenas funcionalidades ou incrementos de produtos passíveis de entrega.

scrumstudy (2022) afirma que durante a *Sprint* são realizadas reuniões diárias curtas e focadas, nas quais os membros discutem o progresso do dia. Torres (2014) destaca que na *standup meeting* cada membro da equipe responde três perguntas:

- O que fez para o projeto desde a última reunião?
- O que fará para o projeto até a próxima reunião?
- Há algum obstáculo para conseguir seu objetivo e/ou precisa de ajuda?

Autores como Pressman e Maxim (2021), Schwaber e Sutherland (2020) e Maschietto *et al.* (2021) concordam que as reuniões diárias melhoram a comunicação entre os membros da equipe, ajudam encontrar potenciais problemas e promover tomadas de decisões rápidas, muitas vezes diminuindo a necessidade de reuniões mais demoradas.

A reunião de revisão de *Sprint* ocorre no final da janela de tempo, sendo geralmente limitada a uma reunião de 4 (quatro) horas para uma janela de tempo de 4 (quatro) semanas. Os participantes são o *Scrum Master*, o PO, os desenvolvedores e outros envolvidos e interessados no produto. Essa reunião pretende principalmente realizar uma demonstração do que foi implementado. Pressman e Maxim (2021) salientam que a demonstração pode não ter toda a funcionalidade planejada, mas sim algumas partes que seriam entregues na janela de tempo estipulada para a *Sprint*.

Para Maschietto *et al.* (2021), a equipe e o PO definem os itens do *Backlog* considerados prontos na *Sprint* e quais não foram finalizados. Para os itens não prontos, são tomadas algumas decisões. Caso não tenham sido iniciados, é definido se ainda são necessários e, para os que não foram concluídos, o time deve determinar a quantidade de trabalho para concluí-los. No fim dessa reunião, o produto deve estar revisado contendo itens para a próxima janela de trabalho.

Para o scrumstudy (2022), o ciclo da *Sprint* termina com uma Reunião Retrospectiva da *Sprint*. Segundo Maschietto *et al.* (2021) a retrospectiva é o momento em que o Time *Scrum* terá a oportunidade de inspecionar a si mesmo e traçar um plano de melhorias para as próximas *Sprints*. Já para Schwaber e Sutherland (2020), o propósito é planejar formas de melhorar a qualidade e a efetividade. Bassi Filho (2008) destaca que nessa reunião a equipe avalia seu trabalho e identifica possíveis oportunidades de melhorias, onde os membros da equipe repen-

sam o último *Sprint* e refinam o processo de desenvolvimento através de uma reflexão sobre o passado.

Pressman e Maxim (2021) definem que durante a reunião são debatidos os seguintes tópicos: segundo:

- O que deu certo na *Sprint*.
- O que poderia melhorar.
- Com o que a equipe se compromete em melhorar na próximo *Sprint*.

O responsável por liderar a Reunião de Retrospectiva é o *Scrum Master*. Essa reunião deve ter um tempo de no máximo três horas para *Sprints* de até quatro semanas. Para Pressman e Maxim (2021), a reunião encoraja a equipe a melhorar as suas práticas, e assim tornar o processo mais eficaz.

2.4.2.2.1 Papéis essenciais (Time Scrum)

Schwaber e Sutherland (2020) definem que a unidade fundamental do *Scrum* é um pequeno time de pessoas que consiste em: um *Scrum Master*, um *Product Owner* (PO) e desenvolvedores. No Time *Scrum* não há sub-times ou hierarquias. O time é multifuncional, e os membros possuem todas as habilidades necessárias para gerar valor a cada *Sprint*, que são autogerenciáveis. A equipe normalmente é composta por dez pessoas, ou menos.

Para Pressman e Maxim (2021) o PO deve possuir visão do produto em diversos níveis, mantendo uma visão de longo prazo com o gerenciamento do *Backlog* e de curto prazo por meio da definição do objetivo da *Sprint*. Torres (2014) que, quando há diversos envolvidos e interessados no produto, o dono deve ser representado por uma pessoa que entenda as necessidades e que possa priorizá-las.

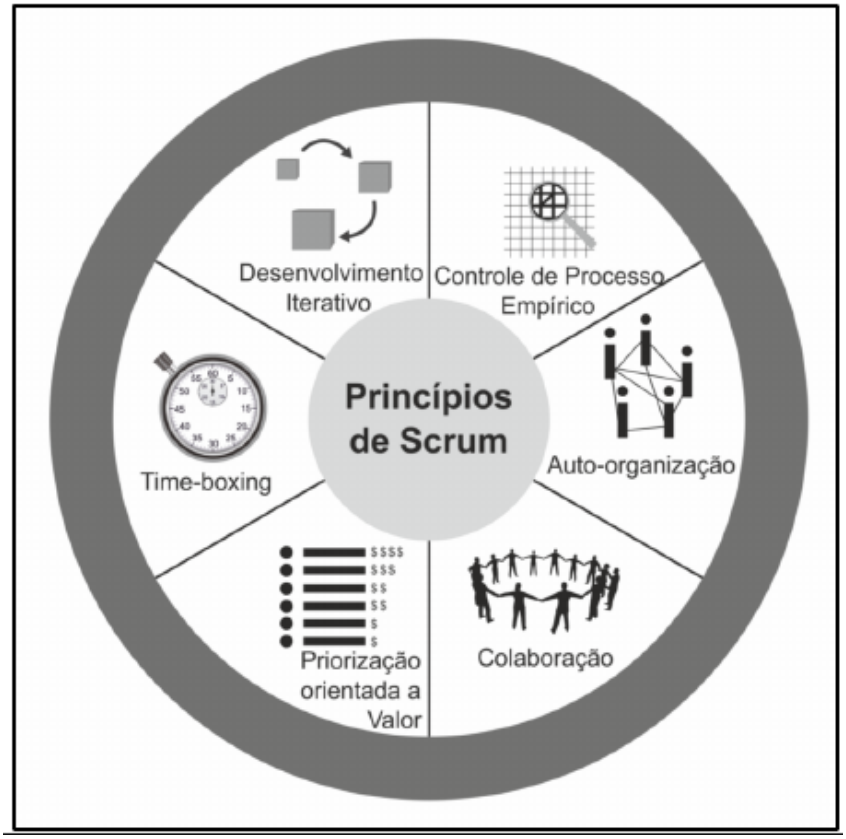
Cohn (2011) cita que o *Scrum Master* deve garantir uma visão ampla do projeto e que o processo seja seguido, podendo ajudar o PO com as decisões que conduzam o projeto a obter o máximo retorno, aconselhando a equipe com suas escolhas. Também é sua responsabilidade eliminar os itens do *Backlog* de impedimentos, fazer gráficos de acompanhamento da evolução do projeto e proteger a equipe contra instabilidades e fatores externos que prejudiquem seu rendimento.

Para Schwaber e Sutherland (2020) e Torres (2014), a equipe de desenvolvimento é vista como uma unidade no *framework*. Ela deve ser multifuncional e autossuficiente, isso significa que seus membros devem reunir as habilidades necessárias para atingir o objetivo. Uma equipe de desenvolvimento *Scrum* deve ser auto-organizada: cada membro deve saber que seu trabalho é igual e essencialmente importante para o sucesso de todos.

2.4.2.2.2 Princípios do Scrum

De acordo com scrumstudy (2022), o *Scrum* conta com seis princípios, os quais são diretrizes fundamentais para a aplicação do *framework* e devem ser usados em todos os projetos. Na Figura 9 são ilustrados os seis princípios do *Scrum* descritos por scrumstudy (2022):

Figura 9 – Seis princípios do *Scrum*



Fonte: scrumstudy (2022).

1. **Controle de Processo Empírico:** enfatiza a filosofia geral do *framework* baseado nas ideias de transparência, inspeção e adaptação. Essa princípio auxilia o aprendizado por meio da experimentação quando o problema não tem soluções claras.

2. **Auto-organização:** uma equipe de desenvolvimento *Scrum* deve ser autogerenciada, onde cada membro reconhece que seu trabalho é igualmente importante e essencial para o sucesso coletivo.

3. **Colaboração:** se concentra nas três dimensões centrais relacionadas ao trabalho colaborativo: conscientização, articulação e apropriação. Defende a entrega do projeto como um processo de criação de valor compartilhado.

4. **Priorização Baseada em Valor:** destaca o foco em entregar o máximo de valor de negócio possível durante todo o projeto.

5. ***Time-boxing* (Gerenciamento de Tempo):** descreve como o tempo é considerado, e ajuda a gerenciar efetivamente o planejamento e a execução do projeto.

6. **Desenvolvimento Iterativo:** define o desenvolvimento iterativo e enfatiza como administrar melhor as mudanças e criar produtos que atendem às necessidades do cliente.

2.4.2.2.3 Elementos Scrum

O *Scrum* define elementos de apoio para seguir suas práticas: *Backlogs* e gráficos de acompanhamento. Como o *Scrum* é um *framework* de grande modularidade por parte da equipe, nem sempre se utiliza de todos esses elementos. Contudo, é aconselhado o uso para uma maior chance de sucesso. O Quadro 5 exemplifica brevemente cada um desses elementos.

Quadro 5 – Elementos do Scrum

Elemento	Descrição
Backlog do Produto	Lista de todos os cartões de funcionalidades que o produto deve possuir e que ainda não foram implementadas.
Backlog Selecionado	Um subconjunto de funcionalidades escolhidas a partir do *Backlog* do produto para ser implementados na *Sprint* atual e que não pode ser modificado durante o *Sprint*.
Backlog do *Sprint*	Lista priorizada obtida a partir da quebra dos cartões do *Backlog* selecionado em tarefas menores.
Backlog de Impedimentos	Lista dos obstáculos identificados pela equipe que não pertencem ao contexto do desenvolvimento.
Gráficos de Acompanhamento	Gráficos que medem a quantidade de trabalho restante (*burndown charts*) são os preferidos em *Scrum*. É recomendado fazê-los para várias esferas do projeto: para o produto, para a *release* e para a *Sprint*.

Fonte: Adaptado de Bassi Filho (2008).

2.4.2.3 Práticas comuns no Ágil

Conforme Bassi Filho (2008) diversas metodologias e *frameworks* ágeis apresentam práticas semelhantes, e podem ser utilizadas independente da metodologia. As principais características comuns entre os métodos são testes, desenvolvimento iterativo, desenvolvimento incremental, colaboração, estimativas, negociação e priorização. Essas práticas serão abordadas nesta seção.

2.4.2.3.1 Testes

Bassi Filho (2008) aborda que nos métodos tradicionais os testes e a implementação são tratados como fases completamente distintas. Isso não ocorre nos modelos ágeis, onde

esta segmentação tende a desaparecer. A implementação e testes acontecem muitas vezes juntos: o mesmo programador cria o código e também o testa.

Souza (2016) contribui expondo que, no diagrama da abordagem tradicional, os esforços de teste acontecem no final da codificação e antes da publicação, porém esse diagrama é idealista demonstrando que há um tempo para os testes como existe para a codificação, o que em muitos projetos não acontece isso não acontece. Por outro lado, o ágil é iterativo e incremental, possibilitando testes a cada incremento de codificação assim que a funcionalidade é publicada. Deste modo, a principal diferença entre as duas abordagens está na definição de "item pronto". Na abordagem tradicional, o produto fica "pronto" para então ser testado, já nas abordagens ágeis, a funcionalidade só está "pronta" após os testes.

Para Madeyski (2009), testes no início do projeto e no seu decorrer facilitam a identificação de problemas e reduz o custo. Algumas metodologias levam isso ao extremo criando os testes antes mesmo do desenvolvimento. Essa técnica é chamada de *Test-Driven Development* (TDD). Além disso, a automação de testes é uma prática muito utilizada em metodologias ágeis, permitindo que os testes sejam executados a cada nova implementação.

2.4.2.3.2 Desenvolvimento iterativo

Os métodos ágeis possuem como característica o desenvolvimento em ciclos, podendo produzir e integrar partes do *software* com sucessivas entregas ao cliente.

Tomás (2009) afirma que o desenvolvimento iterativo acontece em ciclos, com sucessivas entregas ao cliente, o qual pode dar seu *feedback* para que o restante da implementação seja direcionada. Desta forma, o processo torna-se flexível para acomodar mudanças funcionais e de prioridade durante a construção do sistema. Sommerville (2011) aponta que essas mudanças não devem impactar integralmente o sistema, nem modificar sua premissa fundamental.

2.4.2.3.3 Desenvolvimento incremental

Conforme Bassi Filho (2008), o *software* pode receber incrementos funcionais de duas formas: através da adição de novas funcionalidades à medida que o *software* se desenvolve, implementando completamente as funcionalidades e entregando-as; ou evoluindo as funcionalidades juntamente com o sistema, onde as aplicações são criadas de forma simplificada, para serem entregues rapidamente, e caso necessário são melhoradas nas próximas iterações. Sommerville (2011) discute diversas vantagens e desvantagens em utilizar esse tipo de método de desenvolvimento. Uma das vantagens é que se torna mais barato e fácil efetuar mudanças no *software* durante o desenvolvimento.

2.4.2.3.4 Colaboração

Torres (2014) ressalta que a colaboração busca aproximar os *stakeholder* ao desenvolvimento do projeto para poderem acompanhar a evolução do produto. Isso permite uma boa comunicação entre equipe e cliente, evitando surpresas na entrega final, e a colaboração entre os membros da equipe durante o processo de desenvolvimento, que traz eficiência durante processo.

Bassi Filho (2008) completa dizendo que a identificação e resolução de problemas são mais rápidas e as prioridades na implementação podem ser negociadas com mais facilidade.

2.4.2.3.5 Estimativas

Para Bassi Filho (2008) as estimativas podem ser representadas por dois valores um é o palpite sobre determinado evento e o outro é a probabilidade desse evento acontecer.

Pressman e Maxim (2021) afirmam que, como as equipes ágeis se baseiam na comunicação e transparência, elas têm ciência que essas estimativas podem estar incorretas. Uma prática utilizada é realizar estimativas no início do projeto, porém são revisadas ao longo do processo conforme o decorrer do desenvolvimento.

Essas estimativas são feitas em uma escala de valores preestabelecidos pela equipe, e podem ser divididas em funcionalidades ou grupo de funcionalidades. Cohn (2006) sugere a utilização da escala de Fibonacci ou potências de 2, pois são valores exponenciais que vão absorver uma parcela do erro ou incerteza.

As equipes também podem definir como vão utilizar essas escalas, classificadas por complexidade, horas de trabalho, dias ideais. Bassi Filho (2008) explica que os pontos de complexidade e os dias ideais fornecem estimativas de tamanho, medidas sobre o volume de trabalho. Também é possível obter medidas de duração, como horas de trabalho, que preveem a quantidade de tempo necessária para realizar a tarefa.

2.4.2.3.6 Negociação

A negociação é uma habilidade fundamental no contexto ágil, pois valoriza a colaboração e o envolvimento de todas as partes interessadas. Segundo Lindvall *et al.* (2004), a negociação em métodos ágeis envolve a definição de prioridades e o estabelecimento de expectativas realistas em relação ao produto final, permitindo que o time ágil se adapte a mudanças de forma mais eficaz.

De acordo com Cohn (2006), a negociação em metodologias ágeis é baseada em um diálogo colaborativo em que as partes interessadas trabalham juntas para encontrar soluções que atendam aos objetivos do projeto e aos interesses de todos os envolvidos.

Cohn (2006) destaca que a negociação em métodos ágeis é influenciada visto que os requisitos do projeto estão em constante mudança e evolução. Por isso, as partes interessadas devem estar dispostas a adaptar suas expectativas e prioridades ao longo do tempo. Para Bassi Filho (2008), as negociações normalmente se baseiam em ajustes no tempo e no escopo do projeto, dois pontos diretamente relacionados e de interesse do cliente. Custo e qualidade também são pontos que devem ser definidos na negociação.

2.4.2.3.7 Priorização

Para (HIGHSMITH, 2004), o ágil é fortemente baseado em adaptações a mudanças. Com isso suas estratégias de planejamento são focadas a curto prazo, onde há planos mais detalhados, e menos focadas em planos a longo prazo, que se tornam mais superficiais.

De acordo com Bassi Filho (2008), as equipes ágeis reveem seus planos constantemente, e a cada *review*, a equipe tem oportunidade de avaliar as condições do projeto, se baseando para traçar o melhor caminho para chegar aos objetivos.

Para Karlsson e Ryan (1996), um dos maiores riscos do desenvolvimento de *software* está associado ao não atendimento das necessidades e expectativas.

Bassi Filho (2008) destaca que uma má priorização pode acabar por gerar trabalho desnecessário no momento, e até mesmo o atraso na entrega de algumas funcionalidades mais importantes. A priorização tenta evitar o desperdício de recursos. Existem diversas técnicas de priorização que podem ser usadas para guiar essa etapa do processo, destacam-se três: a Ponderação por Pesos Relativos, a Priorização por Valor e Risco e o Modelo de Satisfação do Cliente.

3 TRABALHOS RELACIONADOS

Esta seção contém trabalhos encontrados na literatura, publicados e com alguma relação com o tema desse projeto. Os trabalhos correlacionados com o tema podem ser encontrados, primeiramente, sobre estilos de aprendizagem, os quais utilizam ou aplicam as metodologias de Kolb, ou Honey-Alonso. Em seguida, trabalhos que estudam ou utilizam as Metodologias Ágeis no processo de *software*.

3.1 Identificação de estilos de aprendizagem

Assunção e Nascimento (2019) desenvolvem uma análise quanto ao método de ensino de professores de Ciências e Matemática usando o LSI de David Kolb, trazendo uma relação entre a forma de o professor ministrar os conteúdos das ciências e da matemática e a forma de aprender dos alunos. Esta amostragem foi realizada por meio de entrevista por formulários escritos. No geral, foi constatada uma forma particular de ensinar, além da falta de conhecimento dos professores em relação aos estilos de aprendizagem, bem como do estilo predominante dos alunos analisados.

Araújo *et al.* (2019) buscam identificar os estilos de aprendizagem dos alunos dos cursos de Engenharia de Produção e Pedagogia, na Universidade Federal Rural do Semi-Árido, campus Angicos. Nessa pesquisa os autores utilizaram questionários como técnica de coleta de dados, adaptados do inventário de Honey-Alonso. A adaptação conta com menos questões que o questionário original, contudo não é descrito como os testes foram aplicados. O estudo concluiu que os alunos de Engenharia de Produção têm o estilo de aprendizagem predominantemente ativo e em Pedagogia predomina o estilo reflexivo, e fatores como idade, e gênero também podem afetar o estilo dos alunos.

No trabalho de Rocha (2021) é apresentado o desenvolvimento de uma ferramenta com o intuito de dinamizar a aplicação do questionário de Kolb. O professor e o aluno têm uma interação dinâmica, onde o docente tem a oportunidade de criar os testes para sua turma, e analisar os resultados com gráficos, e os discentes podem conhecer seu estilo de aprendizagem. Rocha (2021) conclui que a ferramenta desenvolvida com Django, denominada L-Style, teve um resultado satisfatório e forneceu a capacidade de analisar e diagnosticar estilos de aprendizagem segundo Kolb.

Com base nos trabalhos citados, nota-se que os questionários são aplicados manualmente, ou em formulários online simples. A opção de ferramenta desenvolvida por Rocha (2021) que antecede este trabalho, traz um dinamismo e interação maior, contudo apresenta falhas, como um processo muito burocrático para o professor, e o desempenho muitas vezes não é satisfatório.

3.2 Processo de *software* com metodologias ágeis

Fontoura (2019) utiliza métodos de desenvolvimento de *software* ágeis com engenharia de requisitos. Na sua pesquisa, o autor desenvolveu um questionário respondido por profissionais de TI, tendo respostas de 30 profissionais da área em diferentes empresas. Nesta pesquisa, tornou-se evidente que a maioria dos profissionais entrevistados emprega o *Scrum* ou alguma adaptação dessa metodologia de trabalho em suas equipes.

Nunes (2016) apresenta um caso de uso da implantação do *Scrum* e XP em pequenas empresas de Tecnologia da Informação (TI). O autor tem o objetivo de viabilizar a solução para a falta de um modelo de desenvolvimento para empresas deste perfil. Nesse estudo o autor conclui que o *Scrum* e XP podem favorecer o processo de *software* nesse tipo de empresa.

Rocha (2022) buscou analisar a aplicação da metodologia *Scrum* para gerenciamento de projetos de *software* de forma remota, em uma instituição de Ciência e Tecnologia. Após realizar a coleta de dados através de entrevistas com membros do time, os dados foram analisados e concluiu que o estudo foi satisfatório, comprovando a aplicabilidade da metodologia remotamente sem perdas significativas para o ambiente analisado. Com isso o autor destaca que o *Scrum* não perde a aplicabilidade de forma remota.

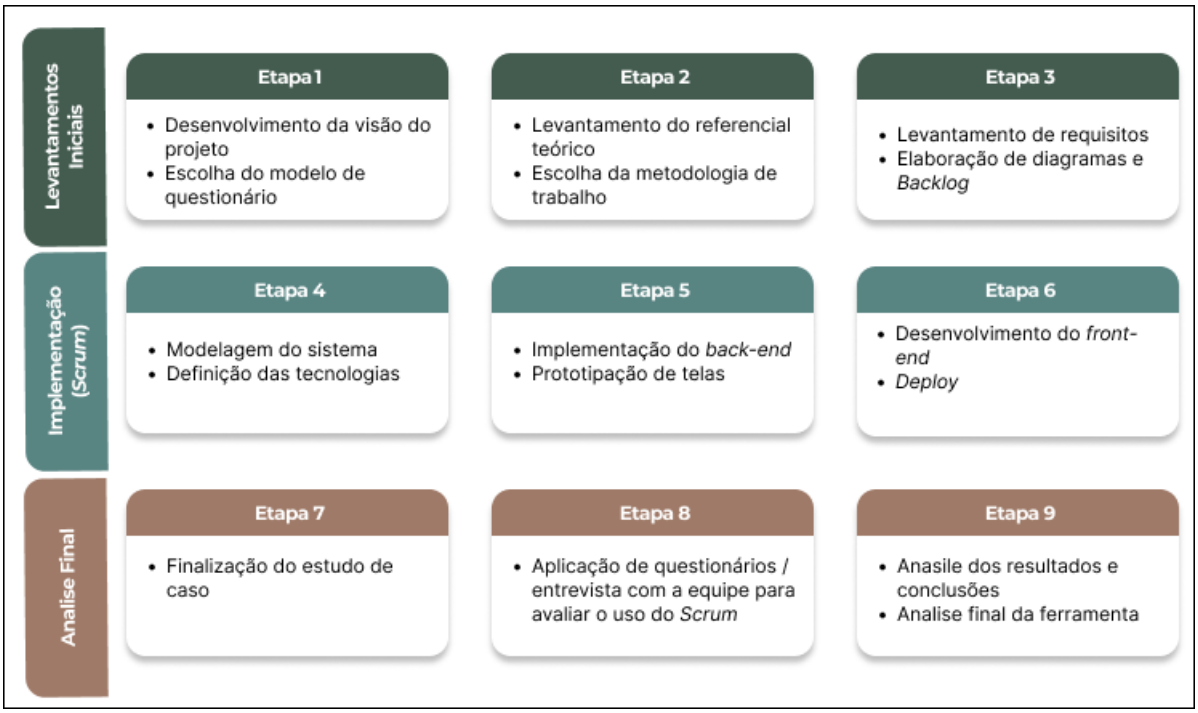
Com base nos trabalhos citados, nota-se uma grande adesão do *Scrum* tanto no ambiente empresarial quanto no ambiente institucional e acadêmico, de forma que esse *framework* vem se fortalecendo no mercado. Entretanto, são escassos os estudos que apresentam a experiência do uso do *Scrum* com equipes compostas por membros voluntários.

4 MATERIAIS E MÉTODOS

Esta seção visa descrever os procedimentos metodológicos adotados, buscando proporcionar clareza e transparência no processo de investigação. Inicialmente será apresentado a descrição dos materiais empregados no processo de desenvolvimento, com ênfase em ferramentas e processos. Em seguida, são apresentados os métodos empregados para alcançar o resultado, considerando aspectos relevantes para o alcance dos objetivos estabelecidos.

A metodologia de pesquisa adotada na condução deste trabalho foi dividida em nove etapas dispostas em três fases: (1) Levantamentos Iniciais, (2) Implementação e (3) Analise Final. O *Scrum* foi empregado primordialmente durante a fase (2), entretanto, sua implementação se iniciou na etapa três da fase (1). Na Figura 10 consta a sequência de atividades do cronograma.

Figura 10 – Sequência de atividades



Fonte: Autoria própria (2023).

4.1 Materiais

Nesta seção serão apresentados os materiais utilizados durante o desenvolvimento desse projeto, focando em *softwares* e ferramentas utilizadas.

4.1.1 Ferramentas de construção

Nesta subseção será apresentado um resumo das ferramentas utilizadas para construção de diagramas, como Diagrama de Caso de Uso da UML e Diagrama Entidade Relacionamento (DER), e de prototipação de interface conforme ilustrado no Quadro 6.

Quadro 6 – Ferramentas de construção

Ferramenta	Descrição
Draw.io	Também conhecido como diagrams.net, é um editor gráfico *online* no qual é possível realizar desenhos gráficos e diagramas de *software* (DRAW, 2023).
DBeaver	Ferramenta de gerenciamento de banco de dados universal de código aberto que permite aos usuários se conectarem a vários bancos de dados. Também dispõe de recursos para gerar diagramas de bancos de dados como o DER (DBEAVER, 2023).
Figma	É uma plataforma *online* para construção de *design* de interfaces e protótipos, com o propósito de trazer colaboração entre pessoas. Atualmente é mantida pela Adobe e conta com funções limitadas para seu plano gratuito (FIGMA, 2023).

Fonte: Autoria própria (2023).

4.1.2 Linguagens de programação, *frameworks* e *drives*

As principais ferramentas e linguagens utilizadas para o desenvolvimento do *software*, incluindo *back-end* e *front-end* conforme ilustrado no Quadro 7:

Quadro 7 – Linguagens de programação, *frameworks* e *drives*

Ferramenta	Descrição
TypeScript	Uma linguagem de programação fortemente tipada baseada em JavaScript, de código aberto e desenvolvida pela Microsoft. Ela adiciona funcionalidades que facilitam a escrita e manutenção do código (TYPESCRIPT, 2023).
JavaScript	É uma linguagem de programação popular usada principalmente para criar e melhorar a experiência do usuário em sites e aplicativos da *web*. É uma linguagem interpretada com tipagem dinâmica fraca (MDN, 2023).
Node.Js	O Node é um *drive* de execução de eventos assíncronos para a linguagem JavaScript. Por ser assíncrono, o Node é uma excelente ferramenta para aplicações cliente-servidor onde o servidor pode ter diversos clientes de uma vez (NODE, 2023).
React.Js	Uma biblioteca JavaScript que permite a criação de interfaces de usuários de alto desempenho. Suas principais vantagens são a componentização e a capacidade de lidar com atualizações em tempo real (META, 2023).
Express.Js	Biblioteca que implementa recursos necessários para desenvolver uma API, recursos completos de rotas, tratamento de erros, recursos de *middlewares*, entre outros recursos necessários para a criação de uma aplicação *back-end* completa (EXPRESS, 2023).
Prisma.orm	É uma ferramenta de mapeamento objeto-relacional moderna e de código aberto para bancos de dados. Ele permite que desenvolvedores trabalhem com bancos de dados de forma mais fácil e intuitiva, escrevendo código em vez de consultas SQL (PRISMA, 2023).

Fonte: Autoria própria (2023).

4.1.3 Ferramentas diversas

Nesta subseção estão presentes os resumos das demais ferramentas utilizadas para chegar ao objetivo final do projeto, sendo essas: editor de código, ferramentas de versionamento, banco de dados, conteinerização, gerenciamento, *proxy* reverso, servidor, entre outras. No Quadro 8 estão descritas as demais ferramentas.

Quadro 8 – Ferramentas diversas

Ferramenta	Descrição
VSCode	É um editor de código-fonte, gratuito, com bom desempenho e diversos recursos facilitadores (STUDIO, 2023).
Google Sheets	Em português Google Planilhas, conta com diversos recursos para gestão e organização de planilhas (GOOGLE, 2023).
Git	O Git é um sistema de controle de versão de código, permite rastrear e controlar as alterações feitas aos arquivos de um projeto, bem como colaborar em tempo real com outros desenvolvedores (GITHUB, 2023).
GitHub	É uma plataforma de hospedagem de projetos de código que permite a colaboração em tempo real em projetos de *software*. O GitHub utiliza Git para rastrear as alterações em um projeto, além de oferecer diversos outros recursos (GITHUB, 2023).
PostgreSQL	É um sistema de gerenciamento de banco de dados relacional de código aberto altamente avançado e escalável (POSTGRESQL, 2023).
Adminer	É uma ferramenta para gerenciamento de conteúdo em bancos de dados, desenvolvida em PHP (ADMINER, 2023).
Insomnia	É uma ferramenta de teste e depuração de *Application Programming Interface* (API) que permite enviar solicitações *Hypertext Transfer Protocol* (HTTP) e visualizar as respostas (INSOMNIA, 2023).
Docker	É uma plataforma de conteinerização de código aberto que permite a criação, implantação e execução de aplicativos em contêineres isolados e portáteis. Essa ferramenta foi utilizada para desenvolvimento e para hospedagem, juntamente com os recursos do *docker-compose* (DOCKER, 2023).
ClickUp	Ferramenta de gerenciamento de projetos e tarefas que oferece diversos recursos para ajudar as equipes a alcançar seus objetivos (CLICKUP, 2023).
ScrumPoker-Online.org	Disponibiliza recursos para a realização de estimativas de tarefas, empregando a escala de Fibonacci. Essa escala é uma abordagem comumente adotada no contexto do desenvolvimento ágil (SCRUMPOKER-ONLINE.ORG, 2023).
Traefik	É um servidor *proxy* reverso de código aberto e balanceador de carga construído para aplicativos de contêineres e microsserviços. Ele oferece facilidade de uso e integração com o Docker (LABS, 2023).
Discord	Aplicação que possibilita criar servidores para comunicação (DISCORD, 2023).
Servidor	O servidor é uma Máquina Virtual disponibilizada pela Universidade Tecnológica Federal do Paraná (UTFPR), com sistema operacional Ubuntu Server, acesso via *Secure Shell* (SSH), 2 GB de memória RAM, e 10 GB de memória ROM.

Fonte: Autoria própria (2023).

É importante ressaltar que todas as ferramentas empregadas neste projeto são de uso gratuito ou possuem planos gratuitos disponíveis. Apenas o servidor foi fornecido pela UTFPR.

4.2 Métodos

Nesta seção serão apresentados os métodos utilizados durante o desenvolvimento desse projeto, focando em estruturar o descrito na Figura 10 e suas etapas, assim como demonstrar a utilização do *Scrum* neste trabalho.

4.2.1 Questionário e visão do projeto

Levando em consideração que o trabalho desenvolvido tem como referências os estudos de Rocha (2021) e Martins (2022), o modelo de Kolb - Quadro 2, foi empregado na primeira versão da aplicação, devido à sua ampla bibliografia e disponibilidade para uso livre. Por outro lado, Martins (2022) optou pelo questionário Honey-Alonso - Anexo A, em virtude de suas perguntas de resposta simples e também por ser de uso livre.

Após delimitadas as metodologias de identificação de estilos de aprendizagem, foi realizada a visão do projeto, documento utilizado em equipes de desenvolvimento para delimitar o escopo do projeto, seus objetivos, premissas, restrições, entre outros itens. Vale destacar que esse documento é utilizado em equipes *Scrum* com objetivo de deixar as partes interessadas cientes do que será desenvolvido e o que fará parte do projeto.

A fundamentação da visão se baseou nas características da ferramenta L-Style em sua primeira versão, desenvolvida em Rocha (2021) e Huff *et al.* (2022). Ademais, as características e limitações relevantes foram estabelecidas em reuniões com as partes interessadas e descritas na visão do projeto. É importante ressaltar que outros projetos podem abordar essas limitações encontradas.

4.2.2 Metodologia de trabalho

Pesquisou-se sobre diversos métodos, dentre eles os tradicionais como cascata e espiral, e ágeis, como XP, *Crystal*, *Scrum*, DSDM, com objetivo de compreender alguns processos de trabalho e desenvolvimento de *software*. Com o levantamento inicial, observou-se que existem diversas metodologias e *frameworks*. Para analisar e aplicar um estudo de caso, o *Scrum* foi escolhido como *framework* de trabalho por ser flexível e adaptativo, além de ter uma composição simples, deixando equipes com pouca experiência a vontade durante o processo.

4.2.3 Uso do *Scrum*

O *Scrum* foi utilizado durante o processo de desenvolvimento da aplicação *Web*, especificado na fase de Implementação - Figura 10, mas seus conceitos também foram utilizados na etapa (3) da fase (1) Levantamentos Inicias. Nesta seção serão descritos os processos, a formação do time, como foram realizadas as reuniões e as adaptações que ocorreram durante o desenvolvimento para obter o melhor rendimento.

4.2.3.1 Time *Scrum*

O Time *Scrum* iniciou com 4 (quatro) desenvolvedores, 1 (um) *design*, e 1 (um) integrante responsável por realizar pesquisas, o sétimo integrante é a professora orientadora, que participa na orientação e como um interessado no produto, totalizando 7 (sete) integrantes no processo. Após quatro meses de projeto, dois integrantes saíram por motivos pessoais, restando 3 (três) desenvolvedores e 1 (um) *design*, totalizando 5 (cinco) integrantes no processo.

Como existem dois papéis principais e indispensáveis no *Scrum*, o PO e o *Scrum Master*, um dos integrantes ficou responsável por essas tarefas. Essa abordagem foi escolhida pois a equipe era pequena e apenas um integrante tinha experiência com o *Scrum*, o qual ficou responsável por gerenciar as atividades a serem efetuadas, *roadmap*, *Product Backlog*, além de definir com a equipe o que será realizado durante a *Sprint*, comandar as reuniões de planejamento, e ser um facilitador para equipe.

4.2.3.2 *Backlog* e *roadmap*

O Levantamento de Requisitos aconteceu juntamente com os primeiros passos do *Scrum*. Os requisitos foram levantados conforme a primeira versão da aplicação e algumas modificações necessárias identificadas durante os testes da primeira versão e conversas com as partes interessadas.

Após a conclusão do Levantamento de Requisitos, o PO elaborou a primeira versão do *Product Backlog* com tópicos que incluíram as funções necessárias para alcançar o objetivo final descrito na visão do projeto. A partir desse ponto, foram criadas as histórias de usuário ao nível macro, também chamadas de épicos, para cada tópico, representando cartões mais abstratos de cada item. Para cada um desses épicos foi atribuído um valor de prioridade.

A equipe, em conjunto com a orientadora, estabeleceram um plano de reavaliação do *Backlog* e *roadmap* a cada quatro ou cinco meses. É importante destacar que o primeiro *roadmap* foi definido exclusivamente pelo PO em uma fase inicial de construção da equipe, que ainda estava sendo formada. Posteriormente, o segundo *roadmap* aconteceu após quatro meses e foi com participação de toda equipe.

Para a elaboração do *Product Backlog* e *roadmap*, a ferramenta Google Planilhas (em inglês, *Google Sheets*) foi utilizada.

4.2.3.3 *Sprints* e reuniões

O tempo definido das *Sprints* foi de duas semanas, contudo em alguns poucos casos, esse tempo variou.

As entregas do produto, para receber os *feedbacks*, aconteceu no intervalo entre uma ou duas *Sprints*, contudo a primeira entrega aconteceu somente depois de quatro meses por não ter servidor disponível para hospedar a aplicação. Isso não significa que o processo foi totalmente longe do cliente, que sempre recebeu novas atualizações a cada *Sprint*. Após a primeira entrega, respeitaram-se as entregas constantes.

As tarefas a serem realizadas na *Sprint* são definidas pelo PO e *Scrum Master* com base no *Backlog* e *roadmap*, as histórias são divididas em tarefas e então passadas para o time, que discute sobre essas tarefas durante a Reunião de Planejamento.

As reuniões realizadas foram Reuniões de Planejamento, Reuniões Diárias (*dayles*), Reuniões de Retrospectiva e *Review*.

As Reuniões de *Review* aconteceram a cada quatro meses ou oito *Sprints*. Nessa reunião a equipe discutiu o que foi feito até o momento, o que ainda precisa ser realizado, e conduziu um novo *roadmap* priorizando itens importantes para o projeto.

A cada início de *Sprint* uma Reunião de Planejamento foi realizada, conduzida pelo *Scrum Master*, na qual a equipe planejou as tarefas a serem realizadas durante a *Sprint*. Durante a reunião, o *Scrum Master* especificou cada uma das tarefas e a equipe decidiu se elas são viáveis para aquela *Sprint*. As tarefas foram então distribuídas para cada integrante, com prioridade dada às habilidades individuais de cada membro.

Reuniões de Retrospectiva aconteceram a cada final de *Sprint*. Nessa reunião cada integrante relatou o que fez no tempo determinado, e relatou três pontos principais sobre a *Sprint*: o que aconteceu de bom, o que aconteceu de ruim, e o que pode melhorar para as próximas *Sprints*.

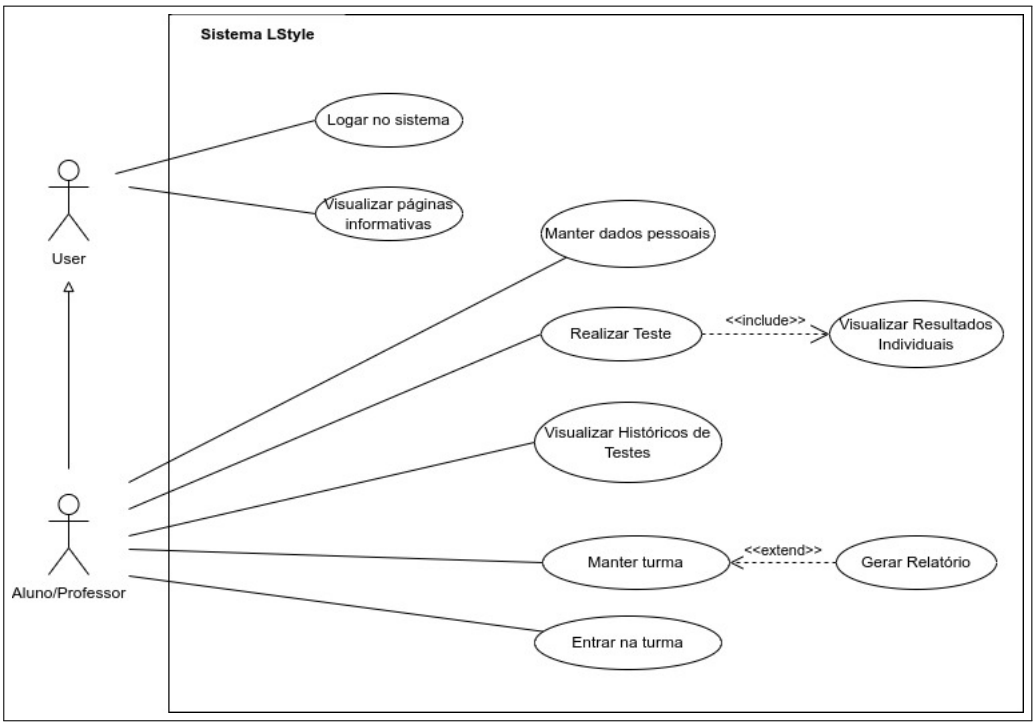
Durante o desenvolvimento do projeto, alguns testes sobre Reuniões Diárias aconteceram baseadas na quantidade de reuniões realizadas durante uma semana. Iniciou-se com duas reuniões por semana, posteriormente passou para cinco reuniões por semana, então foram realizadas algumas *Sprints* com três reuniões por semana. Após esses testes, o time concordou em manter cinco reuniões por semana. Essas reuniões são de no máximo 15 (quinze) minutos e aconteceram majoritariamente pelo aplicativo Discord. Nelas a equipe relata o que fez, o que irá fazer posteriormente e se teve algum empecilho.

4.2.4 Modelagem do sistema

A modelagem do sistema, iniciou-se com o levantamento de requisitos. Para a elaboração dos diagramas foi utilizado a ferramenta *diagrams.net* descrita na subseção 4.1.1.

Após a definição dos requisitos do sistema, foram elaborados os diagramas de Caso de Uso para representar as funcionalidades do sistema. O objetivo principal do diagrama de Caso de Uso é facilitar a compreensão do cliente sobre as funcionalidades do sistema que está sendo desenvolvido. A Figura 11 apresenta o diagrama de Caso de Uso geral do sistema.

Figura 11 – Diagrama de caso de uso



Fonte: Autoria própria (2022).

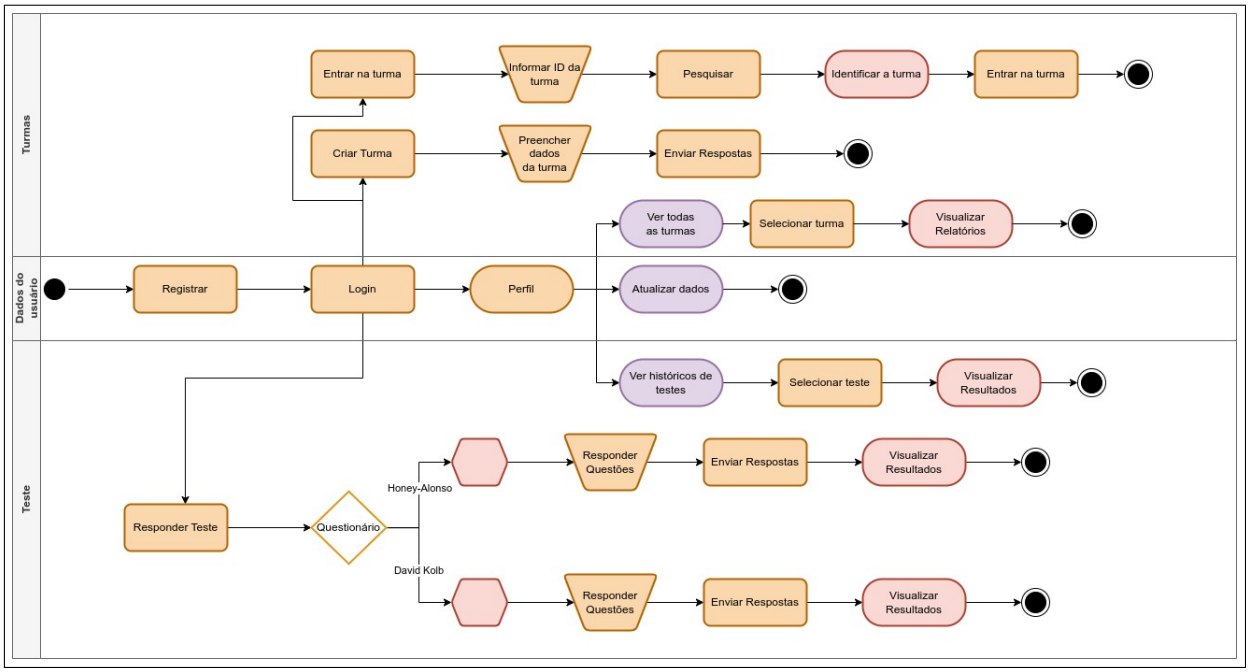
Para a elaboração do diagrama presente na Figura 11 os seguintes requisitos funcionais e não funcionais foram considerados:

- Deve ser possível realizar o login no sistema.
- Deve ser possível realizar o cadastro e editar as informações do perfil.
- Deve ser possível realizar os testes de Honey-Alonso e David Kolb.
- Os testes só podem ser realizados por usuários logados.
- Os usuários que realizaram os testes devem ter a possibilidade de ver seus resultados logo após terminarem o teste.
- É necessário visualizar os resultados dos últimos testes (histórico).

- Todos os usuários logados devem poder criar turmas e gerar relatórios.
- Todos os usuários logados devem poder entrar em uma turma.
- O sistema deve ser rápido e leve.
- O sistema deve permitir diversas aplicações clientes.
- O usuário sem conta deve poder visualizar as páginas informativas.

Para um melhor entendimento do sistema, foi elaborado um diagrama de atividades com o objetivo de auxiliar o cliente a adquirir uma compreensão mais aprofundada do sistema. O diagrama pode ser visualizado na Figura 12.

Figura 12 – Diagrama de atividades



Fonte: Autoria própria (2023).

Posteriormente foi projetado o banco de dados. Nesse processo, a equipe juntou os requisitos do sistema e elaborou um DER para representar visualmente o sistema de banco de dados, com tabelas, campos e relacionamentos. Esse processo foi realizado na ferramenta *DBeaver*.

4.2.4.1 Definição das tecnologias

As tecnologias utilizadas foram escolhidas a partir de algumas tendências e experiências dos desenvolvedores. O principal ponto para a escolha das tecnologias foi a facilidade de acesso a conteúdos e curva de aprendizado baixa para novos membros. As tecnologias escolhidas descritas na subseção 4.1.2 também contam com um bom desempenho e estão em constante aprimoramento.

4.2.4.2 Prototipação de telas

O início do *design* das telas foi desenvolvido na etapa 5 do processo. Nessa etapa também ocorreu o desenvolvimento inicial do *back-end*. A prototipação de telas foi desenvolvida no *Figma*, descrita na subseção 4.1.1. Nessa etapa, o membro da equipe responsável pelo *design* trabalhou com alguns outros membros para construir a interface da aplicação *Web* para uma primeira versão do projeto.

4.2.5 Implementação

As etapas da implementação do *back-end*, também conhecida como lógica do negócio ou processo interno, e da interface do sistema, comumente conhecida de *front-end*, assim como o *deploy*, que pode ser chamado de hospedagem, serão abordados nesta seção. A modelagem e a prototipagem são etapas da fase de (2) implementação, embora esses procedimentos sejam abordados na subseção 4.2.4.

4.2.5.1 Processo interno (*back-end*)

O modelo de desenvolvimento adotado pela equipe para a implementação do *back-end* foi o de uma API *Representational State Transfer* (REST). Esse modelo foi adotado por ser fracamente acoplado, permitindo que vários clientes consumam os dados disponibilizados pela API em um formato padrão, facilitando a extensão do projeto no futuro.

A implementação foi feita inteiramente em TypeScript utilizando o drive de execução Node.Js, como descrito na subseção 4.1.2. Outras ferramentas também foram utilizadas, dando destaque à biblioteca Express.Js utilizada para tratar recursos de requisição e rotas, e o *Prisma.orm* para realizar o mapeamento objeto relacional e consultar o banco de dados.

O padrão de desenvolvimento adotado foi uma extensão do *Model-View-Controller* (MVC) o *Model-View-Controller-Service* (MVCS) que adiciona uma camada extra chamada de serviços, além das camadas modelo, visualização e controle. Ressalta-se que por ser uma API REST, a camada de *View* não é implementada nessa etapa, onde cada requisição retorna ao cliente somente um *JavaScript Object Notation* (JSON) com os dados da requisição. A camada *Model* representa os dados e a lógica de negócio da aplicação. Essa camada mapeia todos os objetos retornados em uma consulta no banco de dados. O *Controller* coordena as interações entre o modelo e a visão, ou seja, a requisição e a resposta ao cliente. A camada *Service* é responsável por gerenciar a lógica de negócio que não pertence diretamente ao modelo, que também não faz parte da visão ou do controlador.

4.2.5.2 Interface da aplicação (*front-end*)

A interface da aplicação é o serviço que consome os dados da API e os apresenta para o usuário, fazendo o papel da camada de visualização. Os dados são requisitados via métodos HTTP, o *back-end* trata essa requisição, retorna os dados no formato JSON, e por fim a aplicação serve esses dados ao usuário.

A implementação do sistema foi realizada no *framework* React.Js utilizando a linguagem TypeScript, descritos na subseção 4.1.2. Nesta fase do projeto, o objetivo da equipe foi reproduzir fielmente o *design* elaborado no Figma. Para facilitar o desenvolvimento, diversos recursos e bibliotecas disponíveis para o React.JS foram utilizados.

O padrão de desenvolvimento adotado para o *front-end* foi o de componentização, onde o time visa criar componentes genéricos e reutilizá-los o máximo possível. A equipe adotou outra prática para aprimorar a organização da estrutura do *web-app*, que consistiu na separação de cada etapa do processo em diretórios distintos. Essa abordagem foi fundamental para facilitar a navegação e a organização durante o desenvolvimento do projeto, promovendo uma maior clareza e precisão na gestão dos arquivos.

4.2.5.3 *Deploy*

A etapa de *deploy* de uma aplicação é fundamental para que ela possa ser acessada pelo público. É nesta fase que o desenvolvedor realiza o processo de hospedagem e disponibilização da aplicação em um ambiente de produção. Nesta seção será descrito o processo para hospedar a aplicação completa, abordando desde a configuração do servidor e a disponibilização da aplicação em produção. Serão apresentados os principais desafios encontrados durante essa etapa e como eles foram solucionados.

O servidor utilizado foi cedido pela UTFPR, e suas configurações estão descritas na subseção 4.1.3. O acesso ao servidor é realizado via SSH, e só é possível realizar o acesso conectado à rede da UTFPR.

No servidor criou-se um diretório onde foram baixados os arquivos do *back-end* e *front-end*. Para colocar uma aplicação Node ou React em produção é necessário fazer o *build*, então antes de iniciar o servidor foi realizado o *build* de ambas aplicações.

Para rodar o servidor foi utilizado a ferramenta Docker e o *docker-compose*, facilitando caso for necessário mudar de servidor. No arquivo *docker-compose-production.yml* iniciou-se alguns contêineres: o banco de dados PostgreSQL, um contêiner para o *back-end*, um contêiner para o *front-end*, e um contêiner do *proxy* reverso utilizado a ferramenta Traefik.

4.2.6 Questionário de avaliação do processo

Segundo Cohn (2011), a avaliação do *Scrum* não é necessário encontrar resultados e respostas perfeitas, mas sim respostas que ajudem a responder algumas questões. Por exemplo: a adoção valeu a pena? O que pode ser melhorado? Estamos fazendo produtos melhores? O produto tem menos defeito? Estamos sendo mais rápidos que costumávamos?

Foi realizado um processo de avaliação interna do *Scrum* e métodos utilizados durante a execução do projeto. Aplicou-se um questionário de avaliação baseado em Souza (2014). Esse questionário foi aplicado em forma de entrevista membro a membro, com um total de quatro membros que participaram durante todo o processo. O questionário conta com cinco perguntas com respostas de 1 a 5 sendo 1 totalmente negativa e 5 totalmente positiva, e conta com três perguntas dissertativas para os membros discorrerem sobre o modelo adotado, totalizando oito perguntas. As perguntas realizadas podem ser observadas no Apêndice A.

Também foi solicitado que os membros presentes na implementação da ferramenta L-Style versão 1 (um) realizassem um comparativo sobre o processo antes e depois da adoção do *Scrum*.

Para o cliente interessado no produto (professora orientadora) que acompanhou o processo de desenvolvimento, foram realizadas cinco perguntas para avaliar sua percepção sobre a equipe e o processo *Scrum*. As perguntas realizadas podem ser observadas no Apêndice B, sendo duas no modelo de assinalar entre 1 e 5 como apresentado anteriormente, e três no modelo dissertativo.

4.2.7 Análise final

A análise final da adoção da abordagem *Scrum* foi conduzida por meio de uma avaliação que incluiu aspectos quantitativos e qualitativos. Para obter uma visão completa, foram realizadas entrevistas com os membros da equipe utilizando o questionário descrito na subseção 4.2.6 e uma análise das tarefas concluídas dentro do prazo estabelecido. Essa abordagem permitiu uma avaliação detalhada do impacto do *Scrum*, em termos de eficiência na entrega de resultados e na percepção e satisfação da equipe em relação ao processo.

A avaliação final da aplicação foi realizada com base em critérios específicos, visando determinar a sua eficácia, adequação aos objetivos estabelecidos no início do projeto e desempenho satisfatório. Foram considerados aspectos como a funcionalidade da ferramenta, sua capacidade de cumprir o papel definido e a obtenção de um desempenho satisfatório. Esses critérios foram essenciais para verificar se a aplicação atendeu às expectativas e necessidades estabelecidas.

5 RESULTADOS

Nesta seção serão expostos os resultados alcançados mediante a condução deste trabalho. O primeiro tópico descreverá os resultados obtidos a partir do desenvolvimento da aplicação *Web LStyle*[1], destacando as funcionalidades implementadas e as contribuições dessa ferramenta. O segundo tópico abordará os resultados obtidos com o experimento de uso do *Scrum*, incluindo as etapas realizadas, as métricas coletadas, as dificuldades encontradas e as conclusões obtidas a partir da análise dos dados.

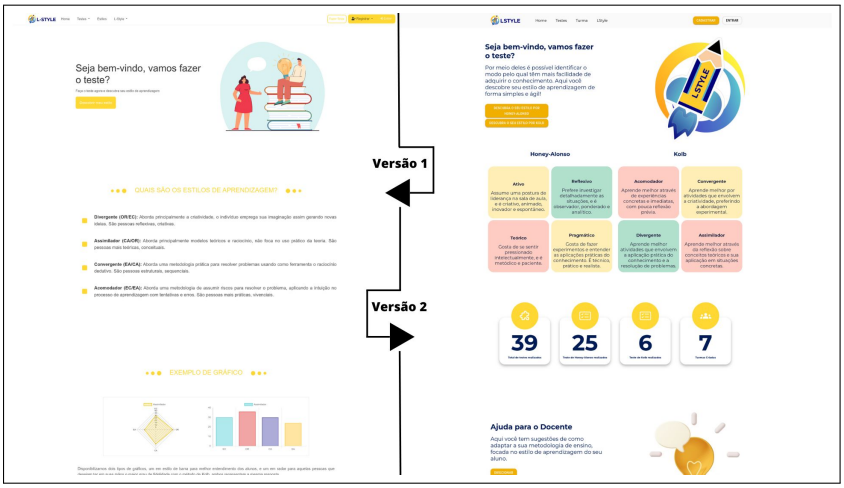
5.1 Resultados da ferramenta

Uma das primeiras modificações foi a alteração do nome passando de *L-Style* para *LStyle*, sem o hífen. *LStyle* é uma plataforma com uma interface que possibilita acessar os testes de forma livre, pública e acessível e democrática, visando proporcionar praticidade aos professores que buscam adotar metodologias para adequar sua abordagem de ensino às necessidades de seus alunos. A aplicação tem como objetivo principal oferecer soluções que promovam maior eficácia e eficiência no processo educacional.

A implementação da versão 2 da aplicação *Web*, trouxe melhorias significativas. Houve aprimoramento no desempenho, reduzindo travamentos, melhorias na interface, tornando-a mais limpa e padronizada. Também foi otimizado o processo de criação de turmas, eliminando a necessidade de refazer o teste a cada nova turma. Ademais, foi adicionada a metodologia Honey-Alonso (1994) ao sistema.

A aplicação *Web* permite que o usuário possa responder um questionário desejado de forma simples, com o menor número possível de cliques, e assim, obter e visualizar seus resultados. Na Figura 13 é apresentado um comparativo entre a versão 1 e a versão 2 da aplicação.

Figura 13 – Comparação entre tela principal das duas versões



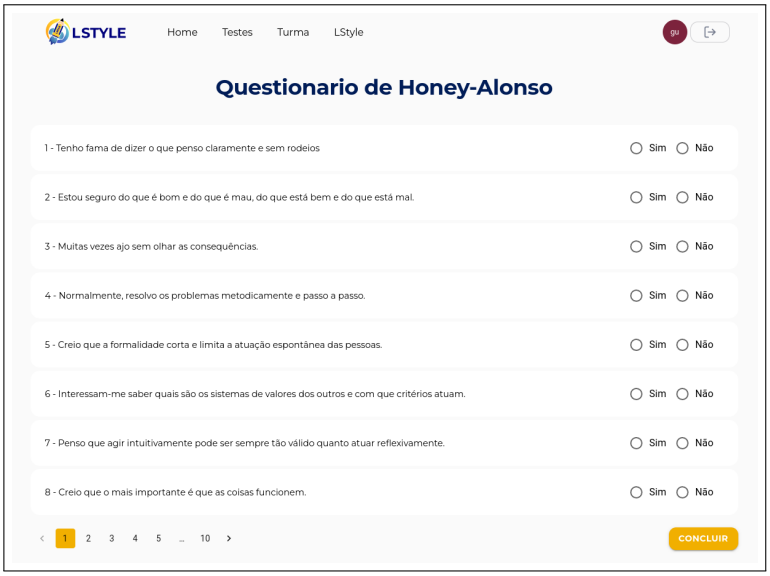
Fonte: Autoria própria (2023).

[1] Disponível em: http://applstyle.sh.utfpr.edu.br

A plataforma *LStyle* possibilita o cadastro de novos usuários, o *login* no sistema, a criação de novas turmas e simplifica o processo de ingresso do usuário em uma turma, com o mínimo de burocracia comparado com a primeira versão. Além disso, a plataforma suporta a participação do usuário em várias turmas sem a necessidade de refazer o teste, tendo em vista que os questionários podem ser extensos. Essa funcionalidade não estava disponível na versão 1 (um) da aplicação *Web*.

A principal funcionalidade da aplicação é a realização do teste tanto do modelo de Honey-Alonso (1994) quanto do modelo de David Kolb (1984). A aplicação oferece um teste otimizado, com uma interface visual simples, permitindo que o usuário se concentre na resposta. A Figura 14 apresenta o questionário de Honey-Alonso.

Figura 14 – Teste de Honey-Alonso



Fonte: Autoria própria (2023).

A Figura 15 apresenta o questionário de Kolb.

Figura 15 – Teste de David kolb

Fonte: Autoria própria (2023).

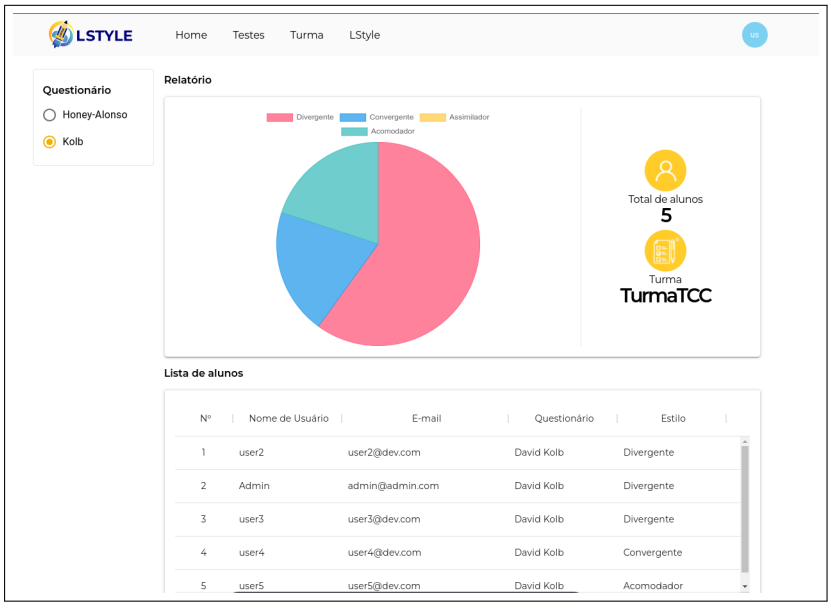
Os resultados de ambos os testes são apresentados de forma clara e direta, por meio de gráficos. Para o teste de Honey-Alonso, é exibido um gráfico de barras, enquanto para o teste de Kolb é utilizado um gráfico de radar. Além disso, são fornecidas breves descrições sobre o estilo de aprendizagem resultante de cada teste, juntamente com algumas dicas sobre como esse estilo de aprendizagem pode ser mais efetivamente abordado. A Figura 16 apresenta um resultado para exemplificar.

Figura 16 – Resultado para um teste de Honey-Alonso

Fonte: Autoria própria (2023).

Além dos resultados individuais, a aplicação *Web* oferece recursos de turmas, permitindo que os professores criem turmas específicas e que os alunos se cadastrem nessas turmas. Essa funcionalidade proporciona ao professor a facilidade de acessar os dados dos seus alunos, bem como visualizar a quantidade de alunos com cada estilo de aprendizagem e identificar o estilo predominante em suas turmas. Na Figura 17 é possível ver a página de turma.

Figura 17 – Página de relatório da turma



Fonte: Autoria própria (2023).

Essa análise abrange tanto o método de Honey-Alonso (1994) quanto o método de Kolb (1984), fornecendo ao professor uma visão completa e detalhada do perfil de aprendizagem dos seus alunos. Dessa forma, o professor pode tomar decisões mais embasadas em relação ao planejamento de aulas e atividades, visando atender às necessidades individuais de cada aluno e promover um ambiente de ensino mais efetivo.

A versão 2 da aplicação foi projetada com uma interface aprimorada, visando proporcionar aos usuários uma experiência mais agradável, simplificada e ágil em comparação com sua versão anterior.

5.1.1 Documentos e modelos

A documentação desenvolvida foi a mais simples possível, buscando o foco no produto que necessitava ser entregue. Foi feito o documento de visão, onde consta escopo do projeto, equipe e papeis. Também foram desenvolvidos diagramas para exemplificar a ferramenta visualmente aos interessados. É possível visualizar os diagrama: caso de uso e atividades na subseção 4.2.4.

Foram desenvolvidas outras documentações relacionadas ao projeto, como a documentação do *back-end*, *front-end* e *deploy*. Outro diagrama desenvolvido foi o DER que pode ser visualizado na Figura 18.

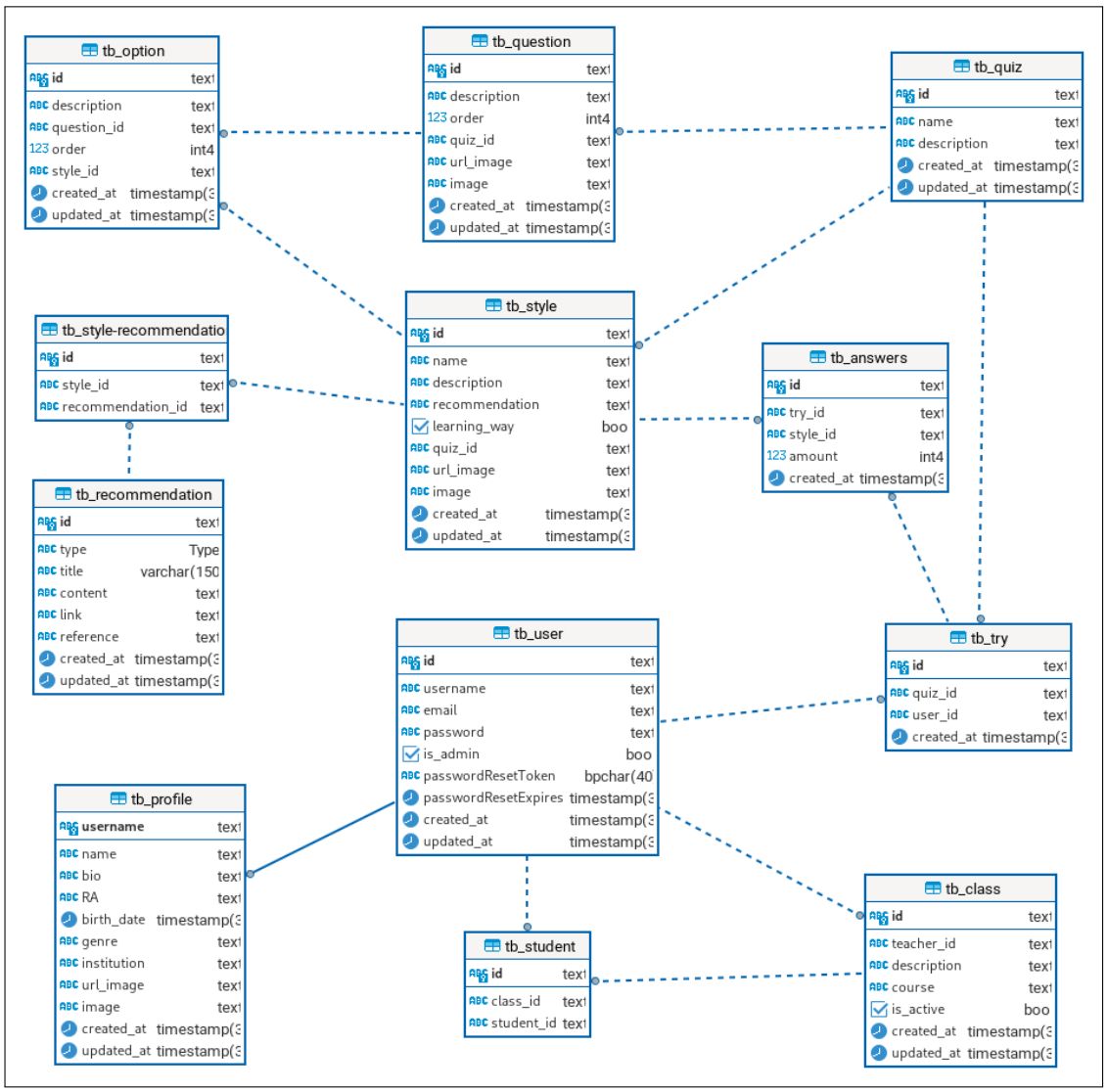
A documentação do *back-end* inclui o diagrama do banco de dados, o modelo de desenvolvimento adotado, bem como o padrão de *commit* utilizado para registrar modificações no código-fonte e as instruções para instalação da aplicação. Consta também as rotas implementadas e os modelos mapeados na aplicação.

Na documentação do *front-end* consta o manual de instalação, como realizar o *commit*, informações sobre os principais serviços como as consultas na API, e informações sobre a palheta de cores e padrões de código. Já a documentação do *deploy* tem informações de como realizar a implementação da aplicação em produção em um servidor.

5.1.2 Resultados *Scrum*

Antes de avaliar a adoção do *Scrum* foram obtidos outros resultados relevantes relacionados ao *framework*. É importante ressaltar, primeiramente, a elaboração da documentação de visão e dos diagramas de caso de uso e atividade, os quais estão disponíveis na subseção 4.2.4. Vale ressaltar que o documento de visão desempenha um papel de extrema importância, uma vez que representa o ponto de partida do projeto. Destaca-se a elaboração do *Backlog* do produto, contemplando diferentes níveis de granularidade, desde épicos até histórias de usuários e tarefas. Essa prática substitui a abordagem tradicional de engenharia de requisitos, proporcionando maior agilidade e flexibilidade no processo de desenvolvimento de *software*. Essa

Figura 18 – Diagrama Entidade Relacionamento



Fonte: Autoria própria (2023).

abordagem ágil, fundamentada no *Scrum*, permite uma melhor adaptação às mudanças e um maior alinhamento com as necessidades dos usuários.

Como descrito na subseção 4.2.3.2, o *Backlog* foi gerado em planilhas do Google. É possível observá-lo na Figura 19, que apresenta elementos como a descrição dos épicos, o mês em que o levantamento foi realizado e a prioridade atribuída a cada um dos épicos.

Uma estratégia adotada pela equipe consistiu em realizar as implementações centrais do *back-end* antes de avançar para a implementação do *front-end*. Essa abordagem permitiu um foco inicial nos aspectos funcionais e lógicos do sistema, enquanto simultaneamente uma parte da equipe se dedicava à pesquisa e ao *design* das interfaces do usuário. Essa divisão de tarefas proporcionou um aproveitamento mais eficiente dos recursos e uma maior agilidade no desenvolvimento, garantindo a qualidade tanto na parte de funcionalidades quanto na experiência do usuário.

Figura 19 – *Backlog* primeira versão

Versão 1 - Agosto de 2022		
Product Backlog Inicial		
Nível		Épicos
ID	História	Prioridade
#1	Cadastro de Usuário	Muito Alta
#2	Perfil de Usuário	Alta
#3	Adicionar Metodologia	Muito Alta
#4	Criar Questionário	Muito Alta
#5	Teste	Muito Alta
#6	Turma	Alta
#7	Sugestões	Baixa

Fonte: Autoria própria (2023).

Na segunda versão do *Backlog*, adotou-se o mesmo método, porém foi observado que a equipe estava mais experiente e conseguiu realizar uma definição de épicos mais precisa e com descrições aprimoradas. A Figura 20 ilustra essa versão do *Backlog*, na qual alguns épicos da primeira versão foram refinados e novos épicos foram adicionados. Vale destacar que os épicos marcados em verde, com a prioridade de “completo", são aqueles que já foram implementados antes da criação do *Backlog* versão 2.

Figura 20 – *Backlog* segunda versão

Versão 2 - Novembro de 2022		
Product Backlog		
Nível		Épicos
ID	História	Prioridade
#1	Cadastro de Usuário	Completo
#2	Perfil de Usuário	Completo
#3	Refatoração do Perfil	Media
#4	Questionário (Honey-Alonso)	Completo
#5	Questionário (Kolb)	Alta
#6	Teste (Honey-Alonso)	Completo
#7	Teste (Kolb)	Alta
#8	Funcionalidades de Turma	Completo
#9	Sugestões	Baixa
#10	Informativo	Muito Alta

Fonte: Autoria própria (2023).

A fim de reduzir o nível de granularidade dos épicos, foram elaboradas histórias de usuário, sendo que cada história está associada a um épico específico. Posteriormente, essas histórias foram divididas em tarefas para serem incluídas na *Sprint*. A Figura 21 apresenta algumas das histórias concebidas pelo PO para os itens listados na Figura 19. Em seguida, essas histórias foram desmembradas e adicionadas ao sistema *ClickUp*.

O *roadmap*, ilustrado na Figura 22, representa uma abordagem para estimar e prever os prazos de entrega de determinadas histórias. É importante destacar que o *roadmap* não é obrigatório, mas sim uma ferramenta que auxilia a equipe no planejamento e estabelecimento

Figura 21 – Histórias de usuário versão 1

ID	#1 - #2	História	Cadastro de Usuário // Perfil de usuário
ID		**Descrição da História**	
#1 - #2	1	Registrar na ferramenta	
#1 - #2	2	Alterar dados de perfil	
#1 - #2	3	Alterar senha (caso tenha esquecido)	
#1 - #2	4	Visualizar perfil próprio	
#1 - #2	5	Visualizar perfil de outros usuários	
ID	#3	**História**	Adicionar Metodologia
ID		**Descrição da História**	
#3	1	Manter estilos de aprendizagem	
#3	2	Manter maneira de aprender	

Fonte: Autoria própria (2023).

de prazos para as histórias. No entanto, é crucial compreender que esse plano pode sofrer alterações ao longo do projeto. A equipe deve constantemente avaliar as tarefas e adaptar o planejamento para entregar o máximo valor possível ao cliente.

Figura 22 – *Roadmap* versão 1

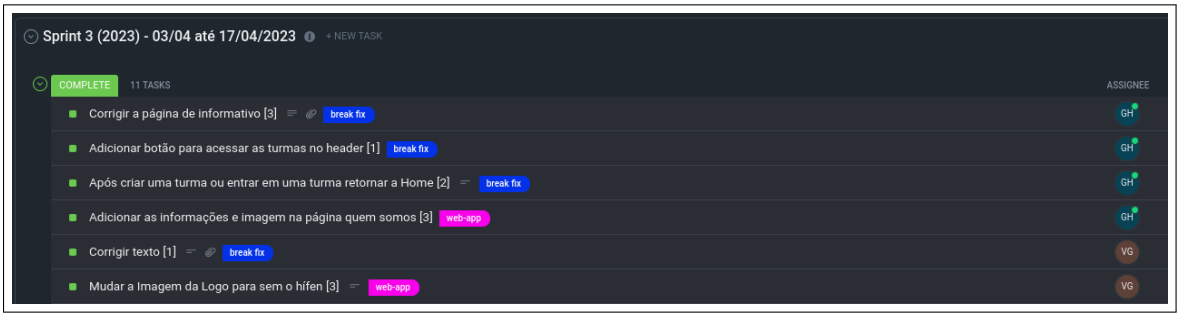
		2022.2			
		Sprint 1 (17/10 -	Sprint 2 (31/10 -	Sprint 3 (14/11 -	Sprint 4 (28/11 -
Tela Inicial					
	Componente bem-vindo				
	Informativo Honey-Alonso				
	Informativo Kolb				
	Card informativo de testes				
	Card direcionamento				
Telas de Formulário					
Cadastro					
	Card Apresentação				
	Formulário				
Login					
	Card Apresentação				
	Formulário				
Cadastro de Turma (iniciar nova turma)					

Fonte: Autoria própria (2023).

Na Figura 22, as linhas representam as histórias de usuário, enquanto as colunas representam as *Sprints*. Para indicar em qual *Sprint* uma história deve ser incluída, utilizou-se a marcação da intersecção entre a história e a respectiva *Sprint* com uma cor específica.

A ferramenta de gerenciamento *ClickUp* foi utilizada para o acompanhamento das *Sprints* e tarefas durante o processo de desenvolvimento. As histórias planejadas para cada *Sprint* foram desdobradas em tarefas antes das Reuniões de Planejamento. Durante essas reuniões, a equipe discutiu a viabilidade e qualidade das tarefas, decidindo se elas seriam incluídas na *Sprint*. Além disso, a complexidade das tarefas foi avaliada por meio de votação, e os membros da equipe, com o auxílio do *Scrum Master*, distribuíram as tarefas entre si. A Figura 23 exemplifica uma *Sprint* definida no *ClickUp*. Para a votação de complexidade utilizou-se a ferramenta *Scrum Poker Online* descrita na subseção 4.1.3.

Figura 23 – Exemplo de *Sprint* no *ClickUp*



Fonte: Autoria própria (2023).

Na Figura 23 é possível observar a interface da ferramenta *ClickUp* utilizada para gerenciar as *Sprints* e as tarefas correspondentes. No canto superior da interface, é possível identificar o título da *Sprint* em questão, juntamente com a data de sua realização. Logo abaixo do título da *Sprint*, encontram-se as tarefas atribuídas a ela.

Ao analisar cada tarefa da esquerda para a direita, é possível identificar diferentes elementos. O primeiro elemento é o *status* da tarefa, que indica o estado atual de execução (por exemplo, “Em andamento", “Concluída"ou “Pendente"). Em seguida, encontra-se o título da tarefa, que fornece uma descrição breve e concisa do objetivo a ser alcançado.

Na sequência, entre colchetes, está indicada a complexidade atribuída à tarefa, utilizando a escala de Fibonacci mencionada anteriormente. Essa complexidade é utilizada para estimar o esforço necessário para a conclusão da tarefa. Outro elemento presente é a *tag*, que consiste em uma etiqueta ou marcador que pode ser atribuído à tarefa para categorizá-la ou identificá-la de acordo com determinado critério. Por fim, à direita da interface, é especificado o membro da equipe a quem a tarefa está destinada, indicando a pessoa responsável pela sua execução.

Essa visualização detalhada das informações das tarefas, proporcionada pela ferramenta *ClickUp*, permite uma melhor organização e acompanhamento do trabalho durante a *Sprint*, contribuindo para a eficiência e o êxito das atividades desenvolvidas.

Durante a realização do trabalho, constatou-se que a prática de realizar reuniões diárias trouxe benefícios significativos para a equipe. Ao estabelecer o hábito de conduzir reuniões diárias, foi possível observar uma melhoria na comunicação e na participação dos membros da equipe, inclusive durante as reuniões de planejamento (*Plannings*).

A frequência das reuniões diárias teve um impacto positivo na colaboração e na troca de informações entre os membros da equipe. Essas reuniões diárias proporcionaram um espaço para cada membro relatar o seu progresso, compartilhar desafios encontrados e receber *feedback* dos demais integrantes. Essa comunicação contínua permitiu um alinhamento mais efetivo entre os membros, facilitando a identificação de possíveis problemas e a busca por soluções conjuntas. Durante o desenvolvimento do trabalho, foi observado que a equipe enfrentou certas dificuldades ao utilizar o sistema de *status* das tarefas. Em algumas ocasiões, ocorreu o esquecimento de atualizar o andamento das tarefas no sistema. No entanto, essa situação não

se tornou um problema significativo, uma vez que as reuniões diárias de atualização permitiam o acompanhamento constante do progresso das atividades.

Por outro lado, quando a equipe realizava menos de quatro reuniões durante a semana, foi observado um distanciamento entre os membros e uma menor interação. Nessas situações, havia uma tendência de perda de sincronia e de informações importantes, especialmente nos dias em que as reuniões diárias não eram realizadas. Esse cenário evidenciou a importância das reuniões diárias como um mecanismo de atualização constante e de alinhamento de esforços.

Portanto, a prática de realizar reuniões diárias mostrou-se eficaz na melhoria da comunicação e no engajamento da equipe, contribuindo para um melhor acompanhamento do progresso do trabalho, a identificação de possíveis obstáculos e a promoção de uma maior sinergia entre os membros durante todo o processo de desenvolvimento do *software*.

A prática da Reunião de retrospectiva da *Sprint* desempenhou um papel importante no aprimoramento contínuo da equipe e no progresso do projeto. Nessa reunião, os membros tiveram a oportunidade de avaliar seu desempenho durante a *Sprint*, identificar pontos fortes e áreas que precisavam de melhorias, e compartilhar sugestões para aprimorar o andamento do trabalho individual e coletivo. Essa reunião é uma prática-chave do método ágil, na qual a equipe reflete sobre o trabalho realizado durante a *Sprint* e busca oportunidades de aprendizado e aprimoramento. Essa reunião promove a transparência, a colaboração e a responsabilidade individual e coletiva dos membros da equipe.

Durante a condução do projeto, foram adotadas outras técnicas ágeis, como a programação em pares, para promover a colaboração e o compartilhamento de conhecimento entre os membros da equipe. No entanto, foi identificado que a implementação dessa prática enfrentou desafios significativos devido a conflitos de horários entre os membros.

Inicialmente, a prática de realizar *feedbacks* constantes durante o desenvolvimento do projeto enfrentou desafios, principalmente durante a etapa de desenvolvimento do *back-end*. No entanto, quando a interface do usuário foi adicionada pela primeira vez, os *feedbacks* começaram a se tornar mais frequentes e trouxeram diversos benefícios para o processo de desenvolvimento. Os *feedbacks* permitiram à equipe realizar ajustes e melhorias contínuas no produto. Eles forneceram percepções valiosos sobre a usabilidade, a experiência do usuário e a qualidade geral do *software*.

5.1.3 Análise do *framework Scrum*

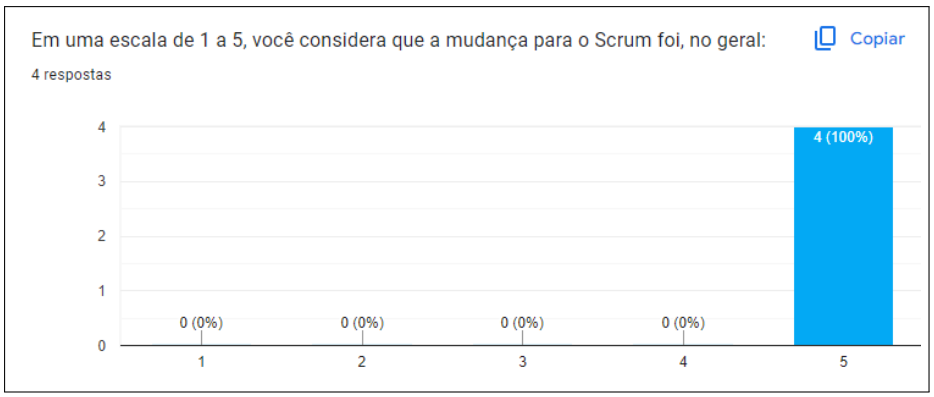
A partir da entrevista com o orientador e com a equipe é possível verificar alguns dados sobre o uso do *Scrum*.

5.1.3.1 Entrevista com a equipe

Para coletar dados com a equipe, optou-se por uma abordagem híbrida com perguntas quantitativas e qualitativas. A fonte de dados veio dos quatro membros remanescentes na equipe, e que estiveram na maior parte do projeto. O conjunto de perguntas utilizadas na entrevista pode ser encontrado no Apêndice A.

De maneira geral, os entrevistados demonstraram uma resposta positiva em relação ao método utilizado no desenvolvimento do projeto. No Gráfico 1 é demonstrada a resposta para a questão 1 do Apêndice A.

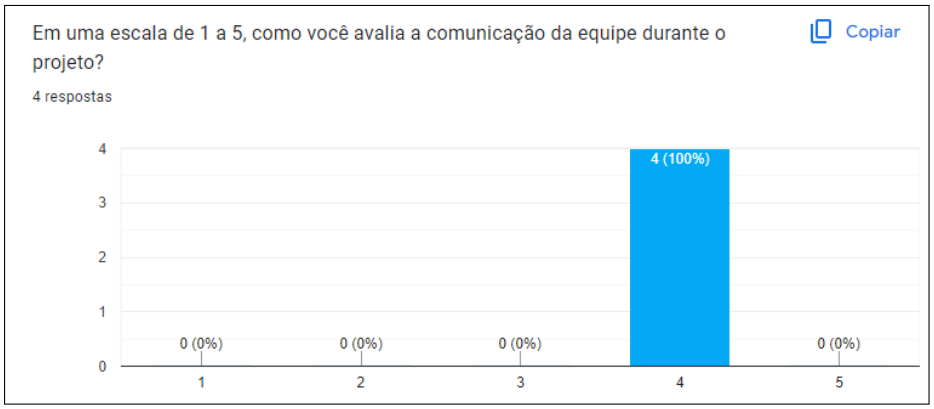
Gráfico 1 – Você considera que a mudança para o *Scrum* foi, no geral



Fonte: Autoria própria (2023).

A equipe avaliou sua comunicação no geral como positiva, isso se dá ao fato da realização das reuniões e conversas de atualização, além de utilizar os canais como *Discord* e *WhatsApp* para a troca de mensagem, contudo consideram que a comunicação ainda pode ser melhor - Gráfico 2.

Gráfico 2 – Como você avalia a comunicação da equipe durante o projeto?

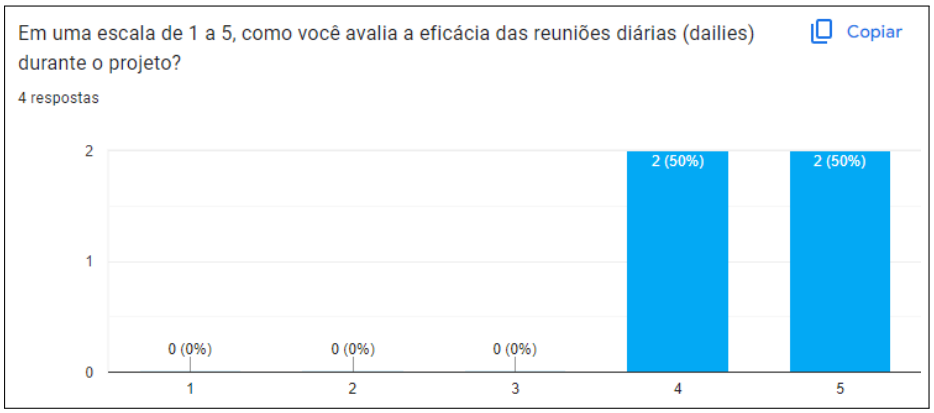


Fonte: Autoria própria (2023).

No Gráfico 3 é possível observar que houve discordância na equipe em relação as reuniões diárias, variando entre totalmente positivas e apenas positivas. Essa divergência ocorre devido ao fato de alguns membros considerarem que, apesar das reuniões diárias serem muito

eficazes quando todos comparecem, frequentemente acontecia de membros não comparecerem à reunião.

Gráfico 3 – Como você avalia a eficácia das reuniões diárias (*dailies*) durante o projeto?



Fonte: Autoria própria (2023).

Referente às perguntas 4 e 5 do Apêndice A, a equipe também permaneceu dividida entre positiva e totalmente positiva. Em relação à produtividade, a equipe ressaltou que houve um ganho significativo. No entanto, alguns membros reconhecem que ainda há espaço para melhorias, embora não atribuam a responsabilidade ao *framework*. Sobre transparência e autogerenciamento, a equipe manteve uma visão positiva em relação à transparência, concordando que a equipe sempre foi clara em relação ao que estava indo bem ou mal. No entanto, é perceptível que a equipe ainda precisa melhorar na questão do autogerenciamento, pois muitas vezes dependeu demasiadamente do *Scrum Master*.

A equipe ao ser questionada sobre “qual é a sua opinião geral sobre o processo do *Scrum* e como ele foi aplicado no projeto?"respondeu que o processo do *Scrum* e sua aplicação no projeto foi positiva. Houve um consenso de que a adoção do *Scrum* trouxe benefícios significativos em comparação ao modelo anterior utilizado. Os entrevistados destacaram aspectos como organização, priorização de tarefas e transparência como pontos positivos da metodologia. Foi mencionado que o *Scrum* agregou valor ao projeto, tanto na qualidade das entregas quanto no desenvolvimento geral. Também foi observado que houve desafios na implementação, contudo à medida que a equipe se adaptou, o processo melhorou gradualmente.

A equipe também concordou que o *framework Scrum* é adequado para a equipe e para o tipo de projeto. Outras metodologias poderiam se encaixar também, contudo o *Scrum* funcionou muito bem. Houve um consenso unânime de que, sem a utilização do *framework Scrum*, os mesmos resultados alcançados não teriam sido possíveis. Isso ressalta a importância de elementos-chave, tais como organização, priorização e comunicação, que desempenharam um papel fundamental no decorrer do processo de desenvolvimento.

Os membros que estavam envolvidos no projeto antes da adoção do *framework Scrum* relataram que, embora tenham enfrentado um aumento na carga de trabalho relacionada à organização, eles observaram um significativo avanço em várias áreas e uma melhoria geral na agilidade do processo.

5.1.3.2 Entrevista com o interessado

Para coletar dados sobre o interesse no produto, optou-se por uma abordagem predominantemente qualitativa em vez de quantitativa. Devido ao fato de haver apenas um entrevistado, as respostas fornecidas pelo mesmo durante a entrevista foram consideradas como principal fonte de dados. O conjunto de perguntas utilizadas na entrevista pode ser encontrado no Apêndice B.

De maneira geral, o entrevistado demonstrou uma resposta positiva em relação ao método utilizado no desenvolvimento do projeto. Na pergunta cinco, ao ser questionado se recomendaria essa ou outra metodologia para equipes que ele orienta ou orientará, o entrevistado destacou que a utilização de uma metodologia é essencial, pois realizar o trabalho sem uma estrutura definida é extremamente difícil.

Ao avaliar a qualidade do produto final entregue em uma escala de 1 a 5, em que 1 representa totalmente negativo e 5 totalmente positivo, o entrevistado atribuiu 4 pontos. Isso indica que, apesar do produto atender às expectativas em grande parte, ainda existem áreas que requerem melhorias. Quanto à comunicação com a equipe, o entrevistado atribuiu 5 pontos, indicando que a comunicação foi eficiente e que houve diversas reuniões ao longo do projeto para manter uma boa interação entre os membros da equipe.

Ao ser questionado se sentiu que suas necessidades e expectativas foram atendidas durante o projeto, o entrevistado expressou que algumas foram atendidas e outras não. Isso indica que, apesar de haver aspectos positivos no atendimento às suas necessidades, ainda existem áreas que podem ser aprimoradas para melhor satisfazê-las. Grande parte das questões não atendidas está relacionada a uma seção específica do sistema, que consiste em uma aba contendo sugestões para auxiliar os docentes na elaboração de aulas de acordo com os estilos de aprendizagem dos alunos, no entanto, é importante ressaltar que essa demanda não é trivial e requer uma extensa pesquisa relacionada ao assunto.

No que diz respeito à percepção sobre o processo do *Scrum* como um todo e se ele ajudou a equipe a entregar um produto de alta qualidade, a resposta do entrevistado foi positiva. Segundo ele, o *Scrum* foi fundamental para garantir uma coordenação eficiente e evitar que o projeto ficasse sem direcionamento. Ele enfatizou que o *Scrum* trouxe melhorias significativas em relação à forma como o trabalho estava sendo realizado anteriormente. Portanto, o entrevistado reconhece o valor do *Scrum* como uma metodologia que contribui para a entrega de um produto de alta qualidade e proporciona uma melhoria substancial no processo de desenvolvimento.

6 CONCLUSÃO

A pesquisa em questão teve como objetivo a análise do *framework Scrum* ao empregá-lo como metodologia ágil no desenvolvimento de um *software* gratuito, acessível, projetado para auxiliar tanto os docentes quanto os discentes na compreensão e identificação de estilos de aprendizagem. Sendo assim, o presente estudo teve como objetivo aperfeiçoar os processos de ensino pedagógico por meio da aplicação. Além disso, buscou-se demonstrar às equipes que atuam no ambiente acadêmico que a adoção de uma metodologia de desenvolvimento estruturada pode trazer benefícios significativos.

Para a ferramenta trazer relevância para os usuários, foram utilizadas duas metodologias de identificação com grande aceitação na área psico-pedagógica, o Modelo de Kolb (1984) e o Modelo de Alonso, Gallego e Honey (1994). Ambas fornecem a capacidade de analisar e diagnosticar os Estilos de Aprendizagem. Durante o decorrer desse trabalho, foram realizados 39 testes com usuários para garantir a eficácia e a precisão dessas metodologias, assim como o bom desempenho da aplicação. A aplicação *Web* apresenta uma proposta de simplicidade e agilidade, permitindo que o usuário acesse a tela de teste de aprendizagem com o mínimo de cliques necessário. Da mesma forma, o professor pode criar e visualizar suas turmas de forma rápida e eficiente, com poucos cliques necessários.

A aplicação *Web* foi projetada por meio de pesquisas com base em teorias da área da educação, podendo ser acessada gratuitamente e de qualquer dispositivo com acesso à Internet pelo link <http://applstyle.sh.utfpr.edu.br/>.

Durante o desenvolvimento ao utilizar o *framework* ágil *Scrum*, a equipe passou por diversas adaptações. As metodologias ágeis surgiram em resposta ao insucesso de apresentados na indústria de *software*. Contudo, percebeu-se que os métodos ágeis não se tratam de um padrão de desenvolvimento, e podem ser adaptados ou utilizados em conjunto buscando o melhor desempenho.

A adoção do método ágil *Scrum* no projeto resultou em significativas melhorias nos aspectos de comunicação e produtividade. O impacto foi positivo, promovendo o trabalho colaborativo da equipe e aprimorando seu desempenho. Além disso, observou-se uma maior interação com os interessados, bem como uma maior receptividade à obtenção de *feedbacks*, tanto em relação à ferramenta, quanto ao processo de trabalho.

Destaca-se a importância das reuniões frequentes, as quais desempenham um papel crucial no fortalecimento da confiança dentro da equipe, bem como no aprofundamento do conhecimento sobre o produto e os processos utilizados pelos membros. Essas reuniões permitem uma comunicação eficaz, possibilitando que os membros compartilhem ideias, solucionem problemas e alinhem suas atividades de forma mais colaborativa e eficiente.

O uso de ferramentas que facilitem o gerenciamento é de grande importância para o sucesso do projeto. No entanto, é relevante considerar que a adoção do *ClickUp* pode não ser a opção mais adequada, especialmente devido às limitações presentes em sua versão gratuita.

É fundamental avaliar cuidadosamente as necessidades da equipe e buscar uma ferramenta que ofereça os recursos necessários para o gerenciamento eficiente das tarefas, considerando aspectos como funcionalidades, escalabilidade e viabilidade financeira. Dessa forma, é possível garantir um ambiente de trabalho produtivo e satisfatório para a equipe.

A entrevista desempenhou um papel fundamental no processo, fornecendo dados valiosos sobre a percepção da equipe em relação ao método adotado, bem como a satisfação do orientador. Essa abordagem permitiu uma análise mais abrangente e embasada sobre a eficácia do processo e seus impactos no projeto na totalidade.

É importante ressaltar que o envolvimento da equipe no processo foi fundamental para o sucesso do projeto. Houve uma participação ativa e colaborativa, e a proposta de adoção do *Scrum* não encontrou resistência por parte dos membros.

Ao final deste trabalho é possível concluir que os objetivos propostos foram alcançados. Inicialmente com a implementação da aplicação *Web* que apesar de ainda ter espaço para melhorias, já é funcional e cumpre seu papel de forma eficiente, simples e democrática. Também, o *Scrum* foi estudado e implementado com sucesso no projeto, e após diversos experimentos e adaptações, apresentou-se uma análise sobre o processo e seus benefícios.

Para trabalhos futuros, sugere-se a implantação do desenvolvimento guiado por testes TDD. O TDD é uma abordagem que enfatiza a criação de testes automatizados antes mesmo da implementação do código, o que contribui para uma maior confiabilidade e qualidade do *software* desenvolvido. Ao adotar o TDD, a equipe poderá obter benefícios adicionais, como uma cobertura de testes abrangente e uma melhor compreensão dos requisitos do sistema. Além disso, essa prática auxilia na detecção precoce de possíveis problemas e facilita a manutenção e evolução. Portanto, a sugestão é explorar a implantação do TDD como uma iniciativa para aprimorar ainda mais a qualidade e a eficiência no desenvolvimento de projetos futuros.

REFERÊNCIAS

ADMINER. **Adminer**. Adminer org, 2023. Disponível em: https://www.adminer.org/. Acesso em: 21 jun. 2023.

ALONSO, C. M.; GALLEGO, D. J.; HONEY, P. **Los Estilos de Aprendizaje: Procedimientos de diagnóstico y mejora**. 7. ed. Bilbao: Editorial Mensajero, 1994.

ARAÚJO, R. B. C. *et al.* Supporting the selection of software requirements. *In*: XIX COLÓQUIO INTERNACIONAL DE GESTÃO UNIVERSITÁRIA. **Anais Eletrônicos ...** UFSC, 2019. p. 1–15. Disponível em: https://repositorio.ufsc.br/handle/123456789/201867.

ASSUNÇÃO, T. V.; NASCIMENTO, R. R. O inventário de estilos de aprendizagem de david kolb e os professores de ciências e matemática: diálogo sobre o método de ensino. **Góndola, Enseñanza y Aprendizaje de las Ciencias**, v. 14, n. 1, p. 14–34, 2019.

BASSI FILHO, D. L. **Experiências com desenvolvimento ágil**. 2008. 153 p. Dissertação (Dissertação de Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2008. Orientador: Prof. Dr. Eduardo Colli. Disponível em: https://www.ime.usp.br/~dairton/files/Dissertacao-DairtonBassi.pdf.

BECK, K. *et al.* **Manifesto for Agile Software Development**. 2001. agilemanifesto.org. Disponível em: https://agilemanifesto.org/. Acesso em: 13 mar. 2023.

BELLO, T. C. de. COMPARISON OF ELEVEN MAJOR LEARNING STYLES MODELS: VARIABLES, APPROPRIATE POPULATIONS, VALIDITY OF INSTRUMENTATION, AND THE RESEARCH BEHIND THEM. **Journal of Reading, Writing, and Learning Disabilities International**, Routledge, v. 6, n. 3, p. 203–222, 1990. Disponível em: https://doi.org/10.1080/0748763900060302.

BRASIL, G. W. S.; HUFF, G. F.; ITO, G. C. L-style: uma soluçáo para reconhecimento de estilos de aprendizagem. *In*: UTFPR SANTA HELENA. **Anais do XII Seminário de Extensão e Inovação & XXVII Seminário de Iniciação Científica e Tecnológica da UTFPR**. Santa Helena(PR), 2022. Disponível em: https//www.even3.com.br/anais/seisicite2022/548714-L-STYLE--UMA-SOLUCAO-PARA-RECONHECIMENTO-DE-ESTILOS-DE-APRENDIZAGEM.

CLICKUP. **ClickUp**. ClickUp, 2023. Disponível em: https://clickup.com/. Acesso em: 21 jun. 2023.

COHN, M. **Agile Estimating and Planning**. Nova Jersey: Prentice Hall, 2006.

COHN, M. **Desenvolvimento de Software com Scrum**. Porto Alegre: Bookman, 2011. 471 p. ISBN 9788577808199.

DANTAS, L. A. d. O. Educação E Formação Profissional, **Aplicação Do Teste De Kolb Na Análise Dos Estilos De Aprendizagem Em Ingressantes Do Curso De Ciências Contábeis**. Frei Paulo - SE: [*s.n.*], 2011. 14 p. Disponível em: https://semanaacademica.org.br/system/files/artigos/artigo_1_0.pdf.

DBEAVER. **DBeaver**. dbeaver, 2023. Disponível em: https://dbeaver.com/download/. Acesso em: 21 jun. 2023.

DISCORD. **Discord**. Discord, 2023. Disponível em: https://discord.com/. Acesso em: 21 jun. 2023.

DOCKER. **Docker**. Docker Inc., 2023. Disponível em: https://www.docker.com/. Acesso em: 21 jun. 2023.

DRAW. **draw.io**. drawio, 2023. Disponível em: https://www.drawio.com/. Acesso em: 21 jun. 2023.

EXPRESS. **Express.js**. Express, 2023. Disponível em: https://expressjs.com/pt-br/. Acesso em: 21 jun. 2023.

FELDER, R. M.; SILVERMAN, L. K. Learning and teaching styles in engineering education. **Engr. Education**, v. 78, n. 7, p. 674–681, 1988. Disponível em: https://www.engr.ncsu.edu/wp-content/uploads/drive/1QP6kBI1iQmpQbTXL-08HSl0PwJ5BYnZW/1988-LS-plus-note.pdf.

FIGMA. **Figma**. figma, 2023. Disponível em: https://www.figma.com. Acesso em: 21 jun. 2023.

FONTOURA, F. **Uso de Metodologias de Desenvolvimento de Software e de Engenharia de Requisitos em empresas de Tecnologia: um estudo a partir de um Survey**. 2019. 66 p. Monografia (Bacharel em Engenharia de Software) — Departamento de Informática e Matemática Aplicada, Rio Grande do Norte, 2019. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/34255/1/UsoMetodologiasSoftwareEmpresasTecnologia_Fontoura_2019.pdf.

FOWLER, M. The new methodology. **martinfowler.com**, 2001. Disponível em: https://martinfowler.com/articles/newMethodology.html.

FOWLER, M. **Extreme Programming**. 2013. Martinfowler. Disponível em: https://martinfowler.com/bliki/ExtremeProgramming.html. Acesso em: 25 out. 2022.

FRANZONI, A. L. *et al.* Student learning styles adaptation method based on teaching strategies and electronic media. *In*: **2008 Eighth IEEE International Conference on Advanced Learning Technologies**. [*S.l.*: *s.n.*], 2008. p. 778–782.

GAMBÚS, A. L. M.; ARAÚJO, L. d. S.; PORTILHO, E. M. L. Estilos de aprendizagem: O estado do conhecimento de 2011 a 2021. **Educação em Foco**, v. 27, n. 1, p. 27064, jan. 2023. Disponível em: https://periodicos.ufjf.br/index.php/edufoco/article/view/38666.

GITHUB. **Documentos do GitHub**. GitHub Inc, 2023. Disponível em: https://docs.github.com/en/get-started. Acesso em: 21 jun. 2023.

GOOGLE. **Google Sheets**. Google, 2023. Disponível em: https://www.google.com/sheets/about/. Acesso em: 21 jun. 2023.

HIGHSMITH, J. **Agile Project Management: Creating Innovative Products**. USA: Addison Wesley Longman Publishing Co., Inc., 2004. 312 p. ISBN 0321219775.

HIRAMA, K. **Engenharia de Software**. e-book. Rio de Janeiro: Elsevier Editora Ltda, 2011. ISBN 9788595155404.

HUFF, G. F. *et al.* L-Style: uma ferramenta para identificar estilos de aprendizagem no ensino superior. *In*: CIET:ENPET|CIESUD:ESUD|2022. **Anais...** São Carlos: Grupo Horizonte UFSCar, 2022. p. 1225–1239. ISSN 2316-8722. Disponível em: https://ciet.ufscar.br/submissao/index.php/2022/article/view/2237.

INSOMNIA. **Insomnia**. Kong Inc., 2023. Disponível em: https://insomnia.rest/download. Acesso em: 21 jun. 2023.

JESUS, W. O. d. J.; CARVALHO, C. V. M.; SILVA, L. A. S. Estilo de aprendizagem de kolb: reflexões acerca do diagnóstico de um curso de licenciatura em química. **Revista Brasileira de Ensino de Ciência e Tecnologia**, v. 12, n. 3, p. 285–306, 2019. Disponível em: https://periodicos.utfpr.edu.br/rbect/article/view/8611.

KALATZIS, A. C. **Aprendizagem baseada em problemas em uma plataforma de ensino a distância com o apoio dos estilos de aprendizagem: uma análise do aproveitamento dos estudantes de engenharia**. Nov 2008. 102 p. Dissertação (Mestrado) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, Nov 2008. Disponível em: https://teses.usp.br/teses/disponiveis/18/18157/tde-05112008-145409/publico/AdrianaCasaleKalatzis.pdf.

KARLSSON, J.; RYAN, K. Supporting the selection of software requirements. *In*: PROCEEDINGS OF THE 8TH INTERNATIONAL WORKSHOP ON SOFTWARE SPECIFICATION AND DESIGN. **Anais...** Germany: IEEE, 1996. p. 146–149. Disponível em: https://ieeexplore.ieee.org/document/501157/citations?tabFilter=papers#citations.

KEEFE, J. W. **Assessing student learning styles: An overview**. Ann Arbor, Michigan: ERIC, 1982.

KNIBERG, H. **Scrum e XP direto das Trincheiras: Como fazemos Scrum**. InfoQ.com: C4Media Inc, 2007. 148 p. ISBN 978-1-4303-2264-1. Disponível em: https://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches/.

KOLB, A.; KOLB, D. **The Kolb Learning Style Inventory—Version 3.1 2005 Technical Specifications**. Cleveland, 2005. Disponível em: https://www.researchgate.net/publication/241157771_The_Kolb_Learning_Style_Inventory-Version_31_2005_Technical_Specifi_cations.

KOLB, D.; KOLB, A. **The Kolb Learning Style Inventory 4.0: Guide to Theory, Psychometrics, Research & Applications**. [*s.n.*], 2013. Disponível em: https://learningfromexperience.com/downloads/research-library/the-kolb-learningstyle-inventory-4-0.pdf.

KOLB, D. A. **Experiential learning: experience as thesource of learning and development**. New Jersey: Prentice-Hall Inc, 1984.

LABS, T. **Traefik Proxy**. Traefik Labs, 2023. Disponível em: https://doc.traefik.io/traefik/. Acesso em: 21 jun. 2023.

LINDVALL, M. *et al.* Agile software development in large organizations. **IEEE Computer**, v. 37, p. 26–34, 01 2004.

LOPES, W. M. G. **ILS - Inventário de estilos de aprendizagem de Felder-Soloman: investigação de sua validade em estudantes universitários de Belo Horizonte**. 2002. 85 p. Dissertação (Dissertação de Mestrado) — Mestrado em Engenharia de Produção da Universidade Federal de Santa Catarina, Florianópolis, 2002. Disponível em: https://repositorio.ufsc.br/xmlui/handle/123456789/82278.

MADEYSKI, L. **Test-Driven Development: An Empirical Evaluation of Agile Practice**. [*S.l.*]: Springer Berlin Heidelberg, 2009. 245 p. ISBN 9783642042874.

MAIELLARO, V. R.; PASQUALI, V. H. C. Estilos de aprendizagem na fatec zona leste. **South American Development Society Journal**, South American Development Society, v. 07, n. 19, p. 19–31, Abr 2021. Disponível em: http://www.sadsj.org/index.php/revista/article/view/386.

MARTINS, N. **Personalização do Ensino: Identificação dos estilos de aprendizagem por meio do software L-Style**. 2022. 39 p. Monografia (Bacharelado em Ciências da Computação)

— Ciências da Computação da Universidade Tecnológica Federal do Paraná, Santa Helena, 2022.

MASCHIETTO, L. G. *et al.* **Desenvolvimento de Software com Metodologias Ágeis**. Porto Alegre: SAGAH, 2021. 242 p.

MASCHIETTO, L. G. *et al.* **Processos de Desenvolvimento de Software**. Porto Alegre: SAGAH, 2020. 249 p. ISBN 9786556900520.

MDN. **JavaScript**. mdn web docs, 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 21 jun. 2023.

META, o. s. **React**. Meta open sourse, 2023. Disponível em: https://react.dev/. Acesso em: 21 jun. 2023.

NODE. **node.js**. node.js org, 2023. Disponível em: https://nodejs.org/en. Acesso em: 21 jun. 2023.

NUNES, R. D. A implantação das metodologias ágeis de desenvolvimento de software scrum e extreme programming(xp): uma alternativa para pequenas empresas do setor de tecnologia da informação. **ForScience**, v. 4, n. 2, p. 1–14, 2016. Disponível em: http://www.forscience.ifmg.edu.br/forscience/index.php/forscience/article/view/117.

PORTILHO, E. M. L. **Questionário honey-alonso de estilos de aprendizagem**. s.d. [s.d]. Escoladigital.pr.gov.br. Disponível em: https://professor.escoladigital.pr.gov.br/sites/professores/arquivos_restritos/files/documento/2019-11/questionario_honey_alonso.pdf. Acesso em: 23 out. 2022.

POSTGRESQL. **About PostgreSQL**. PostgreSQL, 2023. Disponível em: https://www.postgresql.org/. Acesso em: 21 jun. 2023.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: AMGH, 2016. 940 p. ISBN 9788563308781.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 9. ed. Porto Alegre: AMGH, 2021. 658 p.

PRISMA. **Prisma**. Prisma, 2023. Disponível em: https://www.prisma.io/. Acesso em: 21 jun. 2023.

ROCHA, R. **L-STYLE: uma ferramenta para identificação de estilos de aprendizagem**. 2021. 84 p. Monografia (Bacharelado em Ciências da Computação) — Ciências da Computação da Universidade Tecnológica Federal do Paraná, Santa Helena, 2021. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/27588.

ROCHA, R. M. **Análise da aplicação da metodologia SCRUM para gerenciamento de projetos de software de forma remota em uma instituição de ciência e tecnologia de Fortaleza, durante a Pandemia do COVID-19**. 2022. 53 p. Monografia (Graduação em Engenharia Metalúrgica) — Departamento de Engenharia Metalúrgica e de Materiais, Fortaleza, 2022. Disponível em: http://www.repositorio.ufc.br/handle/riufc/64666.

SBROCCO, J. H. T. d. C.; MACEDO, P. C. d. **Metodologias Ágeis - Engenharia de Software sob Medida**. 1. ed. São Paulo: Érica, 2012. ISBN 9788536519418.

SCHWABER, K.; SUTHERLAND, J. **O Guia do Scrum: O Guia definitivo para o Scrum: as regras do jogo**. s. l.: s. n., 2020. 21 p.

SCHWALBE, K. **Information technology project management**. Boston, MA: Cengage Learning, 2014.

SCRUMPOKER-ONLINE.ORG. **About Scrum Poker and Paul & Kai (Impressum)**. Scrumpoker-online.org, 2023. Disponível em: https://www.scrumpoker-online.org/en/about. Acesso em: 21 jun. 2023.

SCRUMSTUDY. **Um Guia para o CONHECIMENTO EM SCRUM (Guia SBOK®)**. 4. ed. Arizona: SCRUMstudyTM, uma marca da VMEdu, Inc., 2022. 409 p. ISBN 978-0-9899252-0-4. Disponível em: www.scrumstudy.com.

SOARES, M. d. S. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. **INFOCOMP Journal of Computer Science**, v. 3, n. 2, p. 8–13, 2004. Disponível em: https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/68.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. 529 p.

SOUZA, D. R. d. **Implementação da metodologia ágil scrum em um ambiente de desenvolvimento**. 2014. 63 p. Monografia (Bacharelado em Tecnologias da Informação e Comunicação) — Tecnologias da Informação e Comunicação da Universidade Federal de Santa Catarina, Araranguá, 2014. Disponível em: https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/130043/TCC\%20Final.pdf?sequence=1&isAllowed=y.

SOUZA, M. V. R. d. **Abordagens de testes: Uma comparação entre as metodologias tradicional e ágil**. 2016. Monografia (Bacharel em Sistemas de Informação) — Centro de Informática da Universidade Federal de Pernambuco, Recife, 2016. Disponível em: https://www.cin.ufpe.br/~tg/2016-2/mvrs.pdf.

STUDIO, V. **Visual Studio Code**. Visual Studio, 2023. Disponível em: https://code.visualstudio.com/. Acesso em: 21 jun. 2023.

TELES, V. M. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. 2. ed. São Paulo: Novatec Editora, 2017.

TOMÁS, M. R. S. **Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação**. [*S.l.*], 2009. Disponível em: https://run.unl.pt/handle/10362/2003. Acesso em: 14 ago. 2022.

TORRES, L. F. **Fundamentos do gerenciamento de projetos**. 1. ed. Rio de Janeiro: Elsevier Editora Ltda, 2014. ISBN 9788535271720.

TURING, A. M. Computing machinery and intelligence. **Mind**, Oxford University Press, v. 59, n. 236, p. 433–460, 1950.

TYPESCRIPT. **TypeScript**. typescriptlang org, 2023. Disponível em: https://www.typescriptlang.org/. Acesso em: 21 jun. 2023.

APÊNDICE A – QUESTIONÁRIO DESTINADO À ENTREVISTA COM OS MEMBROS

Questionário destinado à equipe

Nome:

Data de início:

1. Em uma escala de 1 a 5, você considera que a mudança para o *Scrum* foi, no geral:

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

2. Em uma escala de 1 a 5, como você avalia a comunicação da equipe durante o projeto?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

3. Em uma escala de 1 a 5, como você avalia a eficácia das reuniões diárias (*dailies*) durante o projeto?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

4. Em uma escala de 1 a 5, como você avalia a sua produtividade ao utilizar o *Scrum*?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

5. Em uma escala de 1 a 5, como você avalia o autogerenciamento da equipe, e a transparência durante o projeto?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

6. Qual é a sua opinião geral sobre o processo do *Scrum* e como ele foi aplicado no projeto?

7. Você acha que o processo do *Scrum* é adequado para a equipe e para o tipo de projeto que vocês trabalharam?

8. Você acha que sem o *Scrum*, a equipe chegaria no mesmo resultado?

9 (Apenas membros presentes na versão 1). Você concorda que o *Scrum* facilitou o processo em comparação à primeira versão? Justifique sua resposta.

APÊNDICE B – QUESTIONÁRIO DESTINADO AO INTERESSADO

Questionário destinado ao interessado

1. Em uma escala de 1 a 5, como você avalia a qualidade do produto final entregue pela equipe?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

2. Em uma escala de 1 a 5, como você avalia a comunicação com a equipe durante o projeto?

Totalmente negativa | 1 | 2 | 3 | 4 | 5 | Totalmente positiva

3. Você sentiu que suas necessidades e expectativas foram atendidas durante o projeto?

4. O que você achou do processo do *Scrum* como um todo? Você acha que ele ajudou a equipe a entregar um produto de alta qualidade?

5. Você recomendaria a utilização dessa metodologia a outras equipes?

ANEXO A – TABELA DE QUESTÕES HONEY-ALONSO

1	Tenho fama de dizer o que penso claramente e sem rodeios.
2	Estou seguro do que é bom e do que é mau, do que está bem e do que está mal.
3	Muitas vezes ajo sem olhar às consequências.
4	Normalmente, resolvo os problemas metodicamente e passo a passo.
5	Creio que a formalidade corta e limita a atuação espontânea das pessoas.
6	Interessa-me saber quais são os sistemas de valores dos outros e com que critérios atuam.
7	Penso que agir intuitivamente pode ser sempre tão válido quanto atuar reflexivamente.
8	Creio que o mais importante é que as coisas funcionem.
9	Procuro estar atento ao que acontece aqui e agora.
10	Agrada-me quando tenho tempo para preparar meu trabalho e realizá-lo com consciência.
11	Estou seguindo, porque quero uma ordem na alimentação, no estudo, fazendo exercícios regularmente.
12	Quando escuto uma nova ideia, em seguida começo a pensar como colocá-la em prática.
13	Prefiro as ideias originais e novas, mesmo que não sejam práticas.
14	Admito e ajusto-me às normas somente se servirem para atingir meus objetivos.
15	Normalmente me dou melhor com pessoas reflexivas e pior com pessoas demasiado espontâneas e imprevisíveis.
16	Escuto com mais frequência do que falo.
17	Prefiro as coisas estruturadas às desordenadas.
18	Quando possuo qualquer informação, trato de interpretá-la bem antes de manifestar alguma conclusão.
19	Antes de fazer algo, estude com cuidado suas vantagens e inconvenientes.
20	Estimula-me o fato de fazer algo novo e diferente.
21	Quase sempre procuro ser coerente com meus critérios e escala de valores; tenho princípios e os sigo.
22	Em uma discussão, não gosto de rodeios.
23	Não me agrada envolvimento afetivo no ambiente de trabalho; prefiro manter relações distantes.

24	Gosto mais das pessoas realistas e concretas do que das teóricas.
25	É difícil ser criativo e romper estruturas.
26	Gosto de estar perto de pessoas espontâneas e divertidas.
27	A maioria das vezes expresso abertamente como me sinto.
28	Gosto de analisar e esmiuçar as coisas.
29	Incomoda-me o fato de as pessoas não levarem as coisas a sério.
30	Atrai-me experimentar e praticar as últimas técnicas e novidades.
31	Sou cautelosa na hora de tirar conclusões.
32	Prefiro contar com o maior número de fontes de informação; quanto mais dados houver para refletir, melhor.
33	Tenho tendência a ser perfeccionista.
34	Prefiro ouvir a opinião dos outros antes de expor a minha.
35	Gosto de levar a vida espontaneamente e não ter de planejá-la.
36	Nas discussões, gosto de observar como atuam os outros participantes.
37	Sinto-me incomodado com as pessoas caladas e demasiadamente analíticas.
38	Julgo com frequência as ideias dos outros por seu valor prático.
39	Angustio-me se me obrigam a acelerar muito o trabalho para cumprir um prazo.
40	Nas reuniões apoio às ideias práticas e realistas.
41	É melhor aproveitar o momento presente do que deleitar-se pensando no passado ou no futuro.
42	Incomodam-me as pessoas que sempre desejam apressar as coisas.
43	Apoio ideias novas e espontâneas nos grupos de discussão.
44	São mais consistentes as decisões fundamentadas em minuciosa análise do que as baseadas em intuição.
45	Detecto frequentemente a inconsistência e os pontos frágeis nas argumentações dos outros.
46	Creio que é preciso transpor as normas muito mais vezes do que cumpri-las.
47	Frequentemente, percebo outras formas melhores e mais práticas de fazer as coisas.
48	No geral, falo mais do que escuto.
49	Prefiro distanciar-me dos fatos e observá-los a partir de outras perspectivas.

50	Estou convencido de que deve impor-se a lógica e a razão.
51	Gosto de buscar novas experiências.
52	Gosto de experimentar e aplicar as coisas
53	Penso que devemos chegar logo ao âmago, ao centro das questões.
54	Procuro sempre chegar a conclusões e ideias claras.
55	Prefiro discutir questões concretas e não perder tempo com falas vazias.
56	Incomodo-me quando dão explicações irrelevantes e incoerentes
57	Comprovar antes se as coisas funcionam realmente
58	Faço vários borrões antes da redação final de um trabalho.
59	Sou consciente de que nas discussões ajudo a manter os outros centrados nos temas, evitando divagações
60	Observo que, com frequência, sou um dos mais objetivos e ponderados nas discussões
61	Quando algo vai mal, não dou importância e trato de fazê-lo melhor.
62	Desconsidero as ideias originais e espontâneas se não as percebo práticas.
63	Gosto de analisar diversas alternativas antes de tomar uma decisão.
64	Com frequência, olho adiante para prever o futuro.
65	Nos debates e discussões prefiro desempenhar um papel secundário a ser o líder ou o que mais participa
66	Incomodam-me as pessoas que não atuam com lógica.
67	Incomodam-me ter de planejar e prever as coisas.
68	Creio que o fim justifica os meios em muitos casos.
69	Costumo refletir sobre os assuntos e problemas.
70	O trabalho consciente me traz satisfação e orgulho.
71	Diante dos acontecimentos, trato de descobrir os princípios e teorias em que se baseiam.
72	Com o intuito de conseguir o objetivo que pretendo, sou capaz de ferir sentimentos alheios
73	Não me importa fazer todo o necessário para que o meu trabalho seja efetivado
74	Com frequência, sou uma das pessoas que mais anima as festas.
75	Frequentemente me aborreço com o trabalho metódico e minucioso.
76	As pessoas, com frequência, creem que sou pouco sensível a seus sentimentos.

77	Costumo deixar-me levar por minhas intuições.
78	Nos trabalhos de grupo, procuro que se sigam em método e uma ordem.
79	Com frequência, me interessa saber o que as pessoas pensam.
80	Evito os temas subjetivos, ambíguos e pouco claros.