LUIS FERNANDO SANTOS FERREIRA

RELATÓRIO TÉCNICO SOBRE A CONDUÇÃO DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE DO PONTO DE VISTA DE UM ANALISTA DE REQUISITOS E DE UM PRODUCT OWNER

LAVRAS – MG

2024

LUIS FERNANDO SANTOS FERREIRA

RELATÓRIO TÉCNICO SOBRE A CONDUÇÃO DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE DO PONTO DE VISTA DE UM ANALISTA DE REQUISITOS E DE UM PRODUCT OWNER

Relatório Técnico apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para obtenção do título de Bacharel.

PROF. DR. PAULO AFONSO PARREIRA JÚNIOR
Orientador

LAVRAS - MG

2024

LUIS FERNANDO SANTOS FERREIRA

RELATÓRIO TÉCNICO SOBRE A CONDUÇÃO DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE DO PONTO DE VISTA DE UM ANALISTA DE REQUISITOS E DE UM PRODUCT OWNER

Relatório Técnico apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para obtenção do título de Bacharel.

APROVADA em 11 de Junho de 2024.

Prof. Dr. Paulo Afonso Parreira Júnior	UFLA
Profa. Dra. Renata Teles Moreira	UFLA
Prof. Dr. Maurício Ronny de Almeida Souza	UFLA

gov.br
Documento assinado digitalmente
PAULO AFONSO PARREIRA JUNIOR
Data: 12/06/2024 15:42:25-0300
Verifique em https://validar.iti.gov.br

PROF. DR. PAULO AFONSO PARREIRA JÚNIOR

Orientador

AGRADECIMENTOS

A Deus, por conceder-me o dom da vida e, junto a ele, a oportunidade de despertar a cada dia com força e saúde, para alcançar meus objetivos e superar os obstáculos que se apresentam em meu caminho.

À minha família, por permanecerem sempre ao meu lado, inclusive nos momentos mais desafiadores e conflituosos. Vocês constituíram a base fundamental para minha jornada, permitindo-me evoluir até me tornar o homem que sou hoje.

À Zeeway, empresa que fundei dentro da UFLA, por me permitir aplicar todos os meus ideais e o meu coração. Nela, empenho todas as minhas forças com o objetivo de oferecer soluções tecnológicas que não apenas resolvam problemas e gerem lucro, mas também aproximem o mundo.

À Universidade Federal de Lavras e a todos os seus funcionários, por me proporcionarem a vivência dos momentos mais intensos e enriquecedores em termos de sabedoria. Sou grato pela oportunidade de conviver com uma infraestrutura de excelência e por aprender com os melhores mestres e doutores.

RESUMO

Este relatório técnico explora a experiência do autor como Analista de Requisitos e *Product Owner* na empresa Zeeway, no âmbito da condução de projetos de desenvolvimento de software. O documento está estruturado em cinco capítulos. A introdução fornece uma visão geral do trabalho e seu contexto. Em seguida, é apresentado o percurso formativo na Zeeway, incluindo uma descrição da empresa, seus produtos e serviços, além do modelo de organização adotado. Posteriormente, são abordados os principais *frameworks*, conceitos e tecnologias utilizados nos projetos. As atividades desempenhadas como Analista de Requisitos e *Product Owner* são detalhadas, oferecendo uma visão do cotidiano profissional do autor. Por fim, as considerações finais apresentam discussões sobre o trabalho desenvolvido, destacando conclusões e possíveis direções futuras.

Palavras-chave: *Product Owner*, Análise de Requisitos, Projetos, Software, *Backlog*.

ABSTRACT

This technical report explores the author's experience as a Requirements Analyst and Product Owner at Zeeway, in the context of conducting software development projects. The document is structured into five chapters. The introduction provides an overview of the work and its context. Next, the formative journey at Zeeway is presented, including a description of the company, its products and services, and the organizational model adopted. Subsequently, the main frameworks, concepts, and technologies used in the projects are discussed. The activities performed as a Requirements Analyst and Product Owner are detailed, offering an insight into the author's professional daily life. Finally, the concluding remarks present discussions on the work carried out, highlighting conclusions and possible future directions.

Keywords: Product Owner, Requirement Analysis, Projects, Software, Backlog.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Segundo Castells (2013), na era digital em que estamos imersos, os produtos de software ocupam uma posição de destaque, exercendo um papel indispensável na interconexão da sociedade contemporânea. Nesse contexto, de acordo com Magela (2006), a diversificada gama de produtos de software existentes, englobando desde aplicativos móveis até sistemas complexos, passa um ciclo de desenvolvimento que, embora varie nos modelos de criação e gestão existentes, compartilha as seguintes fases: Requisitos, Projeto, Codificação, Teste e Manutenção.

É relevante ressaltar que, embora essa divisão “estática” seja frequentemente empregada, especialmente em modelos de desenvolvimento tradicionais, a atual predominância dos modelos ágeis traz uma dinâmica mais flexível (Prikladnicki, Will e Milani, 2014). Nesses modelos, as fases podem se entrelaçar de maneira mais fluida e os ciclos podem ser iterativos, se repetindo a um determinado período de tempo, sem uma fronteira rígida entre uma fase e outra. Em muitos casos, as fases coexistem em diferentes momentos do processo, enriquecendo a adaptabilidade do desenvolvimento e aprimorando a resposta às mudanças.

De acordo com o Swebok (2020), na fase inicial, conhecida como Requisitos, ocorre a coleta das informações essenciais para nortear o desenvolvimento do software. Nesse estágio, a análise dos objetivos e propósitos do software é minuciosamente conduzida. A relevância da fase de Requisitos transcende a das outras, uma vez que um defeito nos requisitos pode comprometer as fases subsequentes e, pior ainda, desencadear a quebra das expectativas dos clientes, assim como exemplificado em Valente (2022). Compreender que na base desse processo reside a compreensão precisa das necessidades e desejos do cliente é essencial, pois qualquer falha nessa interpretação poderá resultar em um sistema que, apesar de tecnicamente correto, não se alinha às reais demandas do usuário. Assim, a atenção dedicada à fase de Requisitos é imperativa, pois é nela que se estabelece o alicerce para a concretização do que verdadeiramente se espera do projeto (SWEBOK, 2020).

No contexto abordado, a função do Analista de Requisitos se destaca como

essencial, uma vez que é incumbido da coleta, análise e documentação dos requisitos do sistema. Por meio de técnicas de elicitação de requisitos, tais como entrevistas, oficinas em grupo, entre outros, o Analista identifica as demandas dos clientes e usuários finais, transmitindo-as de maneira clara e concisa para todas as partes interessadas. Adicionalmente, ele valida e verifica os requisitos para garantir sua pertinência e coerência antes do início do desenvolvimento. Sua atuação é crucial para o êxito do projeto, assegurando que o software desenvolvido atenda plenamente às expectativas e necessidades dos usuários.

Ademais, no contexto do desenvolvimento ágil de software, o *Product Owner* (PO) desempenha um papel crucial como representante das partes interessadas e como responsável por definir e priorizar as funcionalidades do produto (Scrum Guide, 2020). O PO é o principal responsável por manter a visão do produto alinhada com os objetivos estratégicos da organização. Sua importância reside na sua capacidade de tomar decisões rápidas e informadas, garantindo que o produto atenda às expectativas dos usuários finais e gere valor para o negócio. Em conjunto com o Analista de Requisitos, ambos desempenham um papel crucial como elo essencial entre as metas do produto e a equipe de desenvolvimento.

O objetivo deste trabalho é apresentar um relato de experiência, do ponto de vista de um Analista de Requisitos e de um PO, sobre a condução de dois projetos distintos: um no segmento de construção e treinamento e outro no setor agrícola. Com isso, pretende-se compartilhar vivências e lições aprendidas, contribuindo para que boas práticas sejam reutilizadas por outras empresas e que armadilhas e erros sejam evitados. Ambos os projetos foram realizados na Zeeway, uma empresa de desenvolvimento de software, com foco em criação de sistemas para diversos setores, tais como agrícola, segurança bancária, logística, entre outros.

O restante deste trabalho está organizado como segue:

No Capítulo 2, será apresentada a empresa Zeeway, seus produtos e serviços, bem como o modelo de organização que moldou o cenário no qual as experiências aqui relatadas se desdobraram.

No Capítulo 3, é abordado o referencial teórico, que abrange a definição de conceitos fundamentais e a apresentação das principais tecnologias utilizadas neste trabalho. São discutidos o conceito de requisito e as metodologias *Scrum* e *Kanban*,

amplamente empregadas no desenvolvimento de software. Ademais, são destacadas duas ferramentas essenciais para o projeto: Figma, uma plataforma de *design* colaborativo; e Microsoft Azure DevOps, uma solução de computação em nuvem amplamente utilizada na indústria de tecnologia.

O Capítulo 4 adentra de forma mais detalhada no relato de experiência, apresentando as atividades realizadas enquanto Analista de Requisitos e PO e explorando sua interligação com o ciclo de vida do desenvolvimento de software.

Finalmente, no Capítulo 5, o trabalho é concluído com as considerações finais, destacando os pontos mais relevantes desenvolvidos ao longo do trabalho e as lições aprendidas.

2 A EMPRESA ZEEWAY

Neste capítulo, busca-se descrever a organização empresarial da Zeeway, contextualizando a fundação da empresa, seus valores, objetivos, serviços e modelo de organização.

2.1 A Empresa

A Zeeway foi fundada em 2022, na cidade de Lavras, MG, por dois alunos da Universidade Federal de Lavras. Desde o início, a Zeeway direcionou seus esforços para o desenvolvimento de produtos e serviços voltados para diversos setores, destacando-se no agrícola, segurança bancária, roteamento, logística, marketing, recrutamento e saúde.

Atualmente, com a expansão da empresa, a Zeeway possui clientes em diferentes regiões do Brasil e da Europa, como Portugal, Minas Gerais, Rio Grande do Sul, Bahia e São Paulo, demonstrando sua presença e relevância em áreas geograficamente distintas. Inicialmente, a empresa adotou o modelo *home office*. Entretanto, devido à cultura fundamentada nos pilares “trabalho duro”, “qualidade” e “honestidade”, os sócios seguiram o exemplo de Elon Musk, que é expressamente contra o *home office* (Forbes, 2022), e reconheceram a importância de trazer toda a equipe estratégica e gerencial para um ambiente presencial. O intuito foi consolidar e transmitir os valores acima mencionados, mantendo, no entanto, parte da equipe operacional (desenvolvedores) no modelo *home office*. Atualmente, a sede da empresa está localizada na cidade de Lavras, MG.

2.2 Produtos e Serviços

A Zeeway, empresa focalizada no desenvolvimento de soluções tecnológicas, oferece um leque diversificado de serviços direcionados às necessidades específicas das empresas, tais como Desenvolvimento de Software e Alocação de TI.

Desenvolvimento de Software. A empresa é especializada na criação de

sistemas de software, abrangendo plataformas *web*, *desktop* e *mobile* (iOS e Android).

Alocação de TI. A Zeeway também se destaca na oferta de uma ampla gama de profissionais talentosos (*Product Owners*, *Product Managers*, especialistas em UI/UX, Desenvolvedores, Analistas e Engenheiros de Dados, entre outros), aptos a atender demandas específicas de TI. Simplificando a gestão, a Zeeway elimina processos de RH, burocracias de contratação e preocupações administrativas, além de eliminar os encargos trabalhistas.

Além destes serviços, a Zeeway disponibiliza os seguintes produtos desenvolvidos internamente para seus clientes: MarcaFácil e Anti-Churn.

MarcaFácil. Por meio de inteligência artificial, o MarcaFácil oferece uma agenda personalizada para agendamento via WhatsApp, dispensando a necessidade de aplicativos externos. Com o MarcaFácil, é possível realizar o gerenciamento completo das informações, programação de folgas, cancelamento de agendamentos e visualização de disponibilidade em tempo real para os clientes.

Anti-Churn. O termo *churn* refere-se à taxa de clientes que deixam de utilizar os produtos ou serviços de uma empresa ao longo de um determinado período de tempo. Essa métrica é frequentemente utilizada para avaliar a fidelidade e a satisfação dos clientes em relação a uma organização. A *Zeeway* oferece uma abordagem customizada no sistema Anti-Churn, proporcionando um conjunto de funcionalidades adaptadas às demandas específicas de cada cliente, visando reduzir o seu *churn*. Essas funcionalidades incluem:

- Incentivo aos clientes por meio de cupons de desconto para manter o engajamento.
- Disponibilização de conteúdo exclusivo e gratuito aos clientes ou benefícios especiais para fortalecer o relacionamento.
- Envio de mensagens personalizadas baseadas em comportamentos predefinidos para uma comunicação mais direcionada e eficaz.
- Possibilidade de retorno de parte do valor gasto como forma de fidelização (*cashback*).
- Promoção de sorteios ou concursos visando o engajamento ativo dos clientes.

- Identificação de inatividade ou ausência de compras para ações proativas de reengajamento.

Estas funcionalidades são configuráveis de acordo com as necessidades de cada empresa, possibilitando estratégias personalizadas para a retenção de clientes.

2.3 Modelo de organização

A estrutura organizacional da Zeeway apresenta uma divisão em dois principais estratos: o estratégico/gerencial e o operacional. O primeiro é composto pelo *Chief Executive Officer* (CEO), *Chief Technology Officer* (CTO), Gerente de Projetos e engloba os times financeiro, jurídico e de *growth*, este último responsável pelos times de marketing e comercial. Por sua vez, o operacional é constituído pelo time de desenvolvimento.

É importante salientar que o time estratégico/gerencial desempenha suas atividades principalmente no escritório central da empresa, enquanto o time operacional, em sua maioria, atua em regime de *home office*. Essa dinâmica se deve à necessidade de atrair profissionais altamente qualificados, cuja escassez é crescente no mercado atual.

O CEO é o responsável pelas decisões estratégicas e pelos relacionamentos institucionais da empresa, enquanto o CTO assume a responsabilidade pelas decisões técnicas referentes à tecnologia. Suas atribuições englobam desde a seleção de linguagens de programação para cada projeto até a resolução de problemas de alta complexidade para a equipe, incluindo treinamentos para os desenvolvedores, entre outras responsabilidades.

O Gerente de Projetos é uma peça-chave e detém um papel crucial, sendo incumbido de garantir a qualidade e o cumprimento dos prazos em todos os projetos. Sua função abarca a alocação e realocação de recursos conforme necessidade, além da identificação e resolução de problemas que possam surgir durante o desenvolvimento dos projetos.

O time de *growth* é encarregado pelo crescimento da empresa, pautado no aumento da base de clientes e receita. Dessa forma, sua responsabilidade se estende aos times de marketing e comercial. Na Zeeway, o departamento de

marketing tem como principal foco servir o departamento comercial, gerando *leads* e realizando a sua triagem para otimizar a taxa de conversão.

Já os setores Financeiro e Jurídico assumem a gestão administrativa e financeira da empresa. Suas atribuições incluem a organização do fluxo de caixa, o gerenciamento das contas a pagar, a renovação e assinatura de contratos, entre outras incumbências.

O time operacional é composto pelo time de desenvolvimento, os quais desempenham um papel fundamental na execução prática dos projetos. Cada projeto é conduzido por uma equipe de desenvolvimento, tipicamente composta por um Analista de Requisitos, um PO, um *Quality Assurance* e os desenvolvedores.

3 REFERENCIAL TEÓRICO

Este capítulo destaca as principais tecnologias, *frameworks* e conceitos empregados ao longo do trabalho, incluindo ferramentas que otimizam processos, reduzem o tempo manualmente investido e aumentam a eficiência produtiva.

3.1 Requisitos

A seção foi elaborada tendo como referência o livro de Valente (2022). Os requisitos delimitam as funcionalidades e restrições de um sistema, sendo categorizados em Requisitos Funcionais, que abordam as funcionalidades do sistema, e Requisitos Não-Funcionais, que tratam das restrições impostas sobre o sistema ou sobre o processo de desenvolvimento. Por exemplo, em um sistema de gerenciamento escolar, os requisitos funcionais englobam a capacidade de cadastrar alunos, gerar boletins e gerenciar presença, entre outros. Já os requisitos não-funcionais estão relacionados à qualidade do serviço prestado, incluindo desempenho, disponibilidade, segurança, portabilidade e privacidade, entre outros aspectos.

Nesse contexto, a definição dos requisitos é uma etapa fundamental na construção de sistemas de software. Um sistema bem desenhado e implementado, com processo de desenvolvimento eficiente e alta cobertura de testes, pode ser ineficaz se não atender às necessidades dos usuários. Problemas na especificação dos requisitos podem acarretar custos elevados, demandando trabalho adicional para corrigir especificações incorretas ou suprir requisitos ausentes, após a conclusão do sistema. Assim, há o risco de o sistema não atender ao propósito inicial, resultando em desperdício de investimento.

Nesse âmbito, os requisitos funcionais geralmente são expressos em linguagem natural, enquanto os não-funcionais são definidos quantitativamente por meio de métricas. A utilização de métricas evita especificações vagas, como “o sistema deve ser rápido e ter alta disponibilidade”, preferindo-se estabelecer metas específicas, como 98% de disponibilidade e tempo de resposta máximo de 1 segundo para 99% das transações realizadas em qualquer janela de 3 minutos.

Nesse contexto, a Engenharia de Requisitos compreende um conjunto de atividades voltadas para a descoberta, análise, especificação e manutenção dos requisitos de um sistema. O termo “engenharia” é empregado para ressaltar a necessidade de conduzir essas atividades de forma sistemática ao longo de todo o ciclo de vida do sistema, utilizando técnicas bem estabelecidas sempre que possível.

As atividades relacionadas à descoberta e compreensão dos requisitos de um sistema são denominadas Elicitação de Requisitos. Elicitar implica em fazer emergir ou extrair. Nesse contexto, o termo refere-se às interações entre os desenvolvedores do sistema e seus *stakeholders*, que podem incluir clientes, usuários finais, gerentes de projeto e outros interessados no produto, com o propósito de extrair e compreender os principais requisitos do sistema em construção.

Diversas técnicas são empregadas para a elicitação de requisitos, incluindo entrevistas com *stakeholders*, aplicação de questionários, análise de documentos e formulários da organização contratante, realização de *workshops* com os usuários, desenvolvimento de protótipos e análise de cenários de uso. Além disso, existem técnicas baseadas em estudos etnográficos. Originário da Antropologia, o termo “etnografia” refere-se ao estudo de uma cultura em seu ambiente natural. Analogamente, na Engenharia de Requisitos, a etnografia é uma técnica de elicitação que sugere que o responsável se integre ao ambiente de trabalho de todas as partes interessadas e observe suas atividades por um período determinado, sem interferir ou opinar sobre as tarefas e eventos observados.

Após a etapa de elicitação, os requisitos tornam-se pontos cruciais a serem cuidadosamente documentados, validados e priorizados. Na abordagem adotada pela Zeeway, o Analista de Requisitos assume a responsabilidade pela elicitação, documentação e validação dos requisitos, enquanto o PO se encarrega da crucial tarefa de priorização, garantindo assim que os recursos sejam direcionados para os requisitos mais relevantes e estratégicos para o sucesso do projeto.

3.2 *Scrum*

Para escrita de toda essa seção, o Scrum Guide (2020) foi usado como referência. O *Scrum* é uma estrutura (*framework*) leve que auxilia pessoas, equipes

e organizações a gerar valor por meio de soluções adaptativas para problemas complexos, caracterizando-se como uma abordagem ágil para o gerenciamento de projetos. O *Scrum* fundamenta-se no empirismo e no pensamento enxuto. Assim, o conhecimento é obtido pela experiência e tomada de decisões com base na observação, enquanto que o pensamento enxuto visa reduzir o desperdício e concentrar-se no essencial.

O *Scrum* utiliza uma abordagem iterativa e incremental para melhorar a previsibilidade e mitigar riscos, envolvendo grupos de indivíduos que coletivamente possuem as habilidades e conhecimentos necessários para desempenhar suas funções, compartilhando ou adquirindo essas habilidades conforme necessário.

Nesse contexto, a implementação do Scrum requer a constituição de uma Equipe *Scrum*, composta por um *Scrum Master*, um PO e os desenvolvedores. Os desenvolvedores são responsáveis por desenvolver qualquer aspecto de um incremento utilizável a cada iteração, ajustando seu plano diariamente em direção aos objetivos do ciclo e garantindo a qualidade por meio da adesão a critérios de aceitação estabelecidos.

O PO é encarregado de maximizar o valor do produto resultante do trabalho da equipe *Scrum*, gerenciando eficientemente o *backlog* do produto (*Product Backlog*), que será detalhado mais adiante neste trabalho. Isso envolve definir e comunicar os objetivos do produto de forma clara, bem como priorizar e organizar os itens do *backlog* de maneira compreensível. Além disso, o PO garante que o *Product Backlog* seja transparente, visível e compreendido por todos os envolvidos no projeto. Para que o PO alcance sucesso, é essencial que toda a organização respeite suas decisões, que são refletidas no conteúdo e na ordem do *Product Backlog*, assim como no incremento inspecionável na revisão do ciclo.

O *Scrum Master* é responsável por “estabelecer o *Scrum*”, auxiliando todos a entenderem a teoria e a prática do *Scrum*, tanto dentro da Equipe *Scrum* quanto na organização. Ele também é responsável pela eficácia da equipe *Scrum*, capacitando-a a melhorar suas práticas dentro do *framework* do *Scrum*. Os *Scrum Masters* são líderes que servem à Equipe *Scrum* e à organização maior, desempenhando diversas funções, como treinar os membros da equipe em auto-gerenciamento, remover impedimentos ao progresso da equipe e garantir a

realização de eventos do *Scrum* de forma positiva e produtiva. Além disso, o *Scrum Master* presta suporte ao PO, ajudando-o na definição eficaz dos objetivos do produto, na gestão do *Product Backlog* e na facilitação da colaboração dos interessados.

Após a constituição da equipe *Scrum* e a compreensão dos papéis de *Scrum Master*, PO e desenvolvedores, é essencial entender os eventos que compõem o *Scrum*. Estes eventos incluem a *Sprint*, *Planning*, *Daily*, *Review* e *Retrospective*.

O *Sprint* é a base para todos os outros eventos do *Scrum*. Os *Sprints* são eventos de duração entre 2 a 4 semanas, que começam imediatamente após a conclusão do *Sprint anterior*. Durante o *Sprint*, é crucial manter a integridade do *Sprint Goal*, que representa de forma clara o que a equipe busca alcançar no *Sprint*. Para garantir isso, nenhum ajuste que possa comprometer o *Sprint Goal* é feito durante a iteração. O *Product Backlog* é refinado apenas quando necessário, a fim de manter a direção do projeto, garantindo assim que as necessidades do cliente sejam atendidas de forma eficaz.

A *Planning* inicia o *Sprint* definindo o trabalho a ser realizado, o PO garante que os participantes estejam preparados para discutir os itens mais importantes do *Product Backlog* e como eles se relacionam com o *Sprint Goal*. O planejamento do *Sprint* aborda por que o *Sprint* é valioso, o que pode ser feito neste *Sprint* e como o trabalho será realizado.

A *Daily Scrum*, por sua vez, tem como propósito inspecionar o progresso em direção ao *Sprint Goal* e adaptar o *Sprint Backlog*, conforme necessário. Trata-se de um evento de 15 minutos para os Desenvolvedores, no qual eles discutem o progresso desde o último encontro e planejam o trabalho para o próximo. Este evento promove a comunicação, identifica impedimentos e promove a tomada de decisões rápidas.

A *Review* tem como objetivo inspecionar o resultado do *Sprint* e determinar adaptações futuras. Durante o evento, a equipe *Scrum* e os *stakeholders* revisam o que foi realizado no *Sprint*. Os *stakeholders* desempenham um papel importante ao fornecer *feedback* sobre o incremento do produto entregue durante o *Sprint*. Pois, embora não façam parte da equipe *Scrum*, os *stakeholders* são essenciais para garantir que o produto atenda às necessidades e expectativas do cliente e do

mercado.

A *Retrospective* visa planejar maneiras de aumentar a qualidade e a eficácia da equipe *Scrum*. A equipe analisa como o último *Sprint* foi em relação a indivíduos, interações, processos e ferramentas. Eles identificam problemas encontrados e discutem melhorias para aumentar a eficácia e o relacionamento da equipe.

Ao entender com clareza os eventos, é necessário compreender também os artefatos do *Scrum*, pois eles representam trabalho ou valor e são projetados para maximizar a transparência das informações chave. Os artefatos mencionados neste texto são: *Product Backlog*, *Sprint Backlog* e Incremento.

O *Product Backlog* é uma lista dinâmica e priorizada de todos os requisitos, funcionalidades, melhorias e correções que precisam ser feitas em um produto.

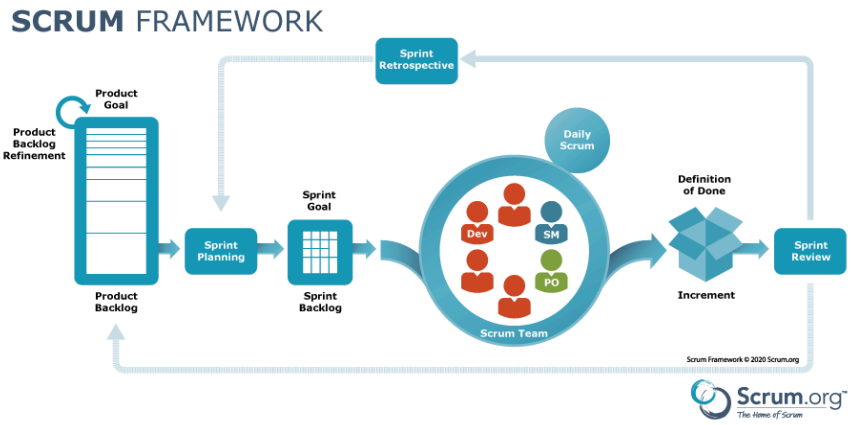
O *Sprint Backlog* é uma lista de itens do *Product Backlog* selecionados pela equipe *Scrum* para serem trabalhados durante um *Sprint* específico. Ele contém todas as tarefas, atividades e requisitos que a equipe planeja realizar durante o *Sprint* para alcançar a *Sprint Goal*.

O Incremento é um resultado tangível e concreto do trabalho realizado pela equipe durante um *Sprint.* É uma parte integral do desenvolvimento iterativo e incremental do produto. Cada Incremento é uma adição ao produto que deve ser completamente funcional, potencialmente entregável e devidamente testado.

A Figura 1 ilustra como esses artefatos, juntamente com os eventos e papéis definidos, compõem o *framework Scrum,* permitindo que equipes e organizações adotem uma abordagem ágil para gerenciar projetos e desenvolver produtos de maneira eficaz e adaptativa.

No contexto da *Zeeway*, o *Scrum* foi adotado como *framework* para a gestão de diversos projetos na empresa. As práticas fundamentais do *Scrum*, tais como a realização de *Sprints*, a definição e priorização do *Product Backlog*, e a realização de reuniões de *Planning*, *Daily*, *Review* e *Retrospective*, foram aplicadas em sua totalidade. Embora tenham sido feitas algumas adaptações específicas para atender às necessidades de cada projeto, os princípios fundamentais do *Scrum*, como a transparência, inspeção e adaptação, foram mantidos, garantindo assim uma abordagem ágil e eficaz para o gerenciamento de projetos em toda a empresa.

Figura 1 - Processo *Scrum*



Fonte: Scrum (2024)

3.3 *Kanban*

Para a escrita desta seção, a publicação Totvs (2023) foi usada como referência. O *Kanban* foi originado na indústria automotiva e posteriormente adaptado para diversos contextos, incluindo o de desenvolvimento de software. Ele é um *framework* que se baseia na visualização do trabalho, limitação do trabalho em progresso, gestão do fluxo e na busca pela melhoria contínua.

O *Kanban* oferece uma visualização clara do fluxo de trabalho planejado e real, permitindo identificar possíveis gargalos e ajustar o processo para garantir a fluidez das atividades da empresa. Embora a estrutura padrão inclua as três colunas básicas (“A fazer”, “Fazendo” e “Feito”), como exemplifica a Figura 2, o *Kanban* pode ser adaptado conforme as necessidades específicas de cada equipe ou projeto.

Na coluna “A fazer”, são listadas as tarefas que ainda precisam ser realizadas, geralmente posicionadas à esquerda do quadro. Cada cartão representa uma tarefa a ser executada sequencialmente. Na coluna “Fazendo”, são exibidas as tarefas em andamento, representando o trabalho atual do time ou colaborador. Devido à natureza do processo de entrega contínua, assim que uma tarefa é concluída, outra pode ser iniciada imediatamente. A coluna “Feito” registra as tarefas que foram concluídas com sucesso. O objetivo é mover todos os cartões para esta coluna com a máxima eficiência, indicando progresso e produtividade.

Figura 2 - Quadro *Kanban*

Fonte: Totvs (2023)

Além disso, é essencial estabelecer limites para o trabalho em progresso, como por exemplo, não iniciar uma nova produção sem antes concluir um pedido em andamento. Essa prática evita atrasos nas entregas e mantém um fluxo de trabalho consistente. O foco deve sempre estar na entrega de atividades com qualidade, minimizando possíveis gargalos entre as etapas do processo.

Na *Zeeway*, o *Kanban* atua em conjunto com o *Scrum*, proporcionando um complemento valioso. Todos os itens do *backlog* da *sprint* são incorporados a um quadro *Kanban*, conforme apresentado no Capítulo 4.

3.4 Figma

Esta seção foi escrita tendo como referência base o artigo o artigo disponível em Alura (2023). O software Figma é amplamente reconhecido como uma ferramenta versátil e poderosa para a criação de interfaces e protótipos. Conforme explicado por Lowdermilk (2019), os protótipos são instrumentos indispensáveis que possibilitam aos usuários antever como será a aplicação quando estiver concluída, engajando-os ativamente no processo de desenvolvimento. Neste contexto, o Figma se sobressai, graças à sua interface que permite a elaboração de protótipos ricos em detalhes. Além disso, sua natureza colaborativa facilita o trabalho entre profissionais de diferentes localidades em um mesmo projeto, promovendo a comunicação e o

compartilhamento de ideias de maneira eficiente.

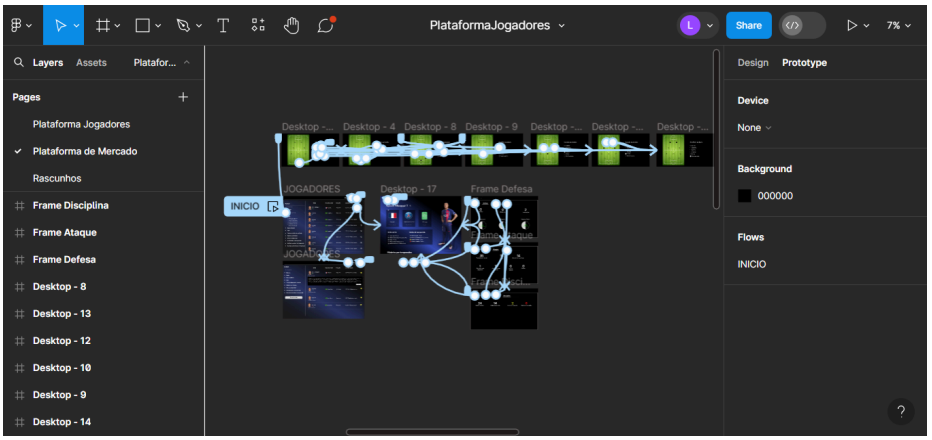
O destaque do Figma reside na sua ampla gama de recursos avançados, que capacitam os *designers* a transformarem suas visões em realidade. Desde formas básicas até efeitos de sombra e desfoque, o Figma oferece um conjunto completo de ferramentas para criar interfaces impressionantes e altamente personalizadas.

Um dos pontos mais relevantes do Figma é o uso de componentes, que possibilitam aos *designers* criar elementos reutilizáveis e consistentes em todo o projeto. Ao simplificar a edição e atualização desses elementos, os componentes garantem uma experiência de usuário coesa e intuitiva, além de economizar tempo no processo de *design*.

A introdução do conceito de variáveis no Figma é outro aspecto crucial, permitindo aos *designers* armazenar e reutilizar estilos de *design*, promovendo consistência visual em todos os projetos da empresa. Essa funcionalidade desempenha um papel fundamental na manutenção de uma identidade de marca coesa e na facilitação da colaboração entre equipes de *design*.

Os recursos de prototipagem do Figma também merecem destaque, pois permitem aos *designers* criar experiências interativas e imersivas sem a necessidade de escrever código. Esses protótipos ajudam a validar conceitos de *design* e aprimorar a usabilidade do produto final, simulando o fluxo de navegação do usuário. A Figura 3 exemplifica esse processo, demonstrando como, ao clicar em pontos específicos da tela, o usuário é redirecionado para outras partes do protótipo.

Figura 3 - Exemplo de navegação em um protótipo no Figma

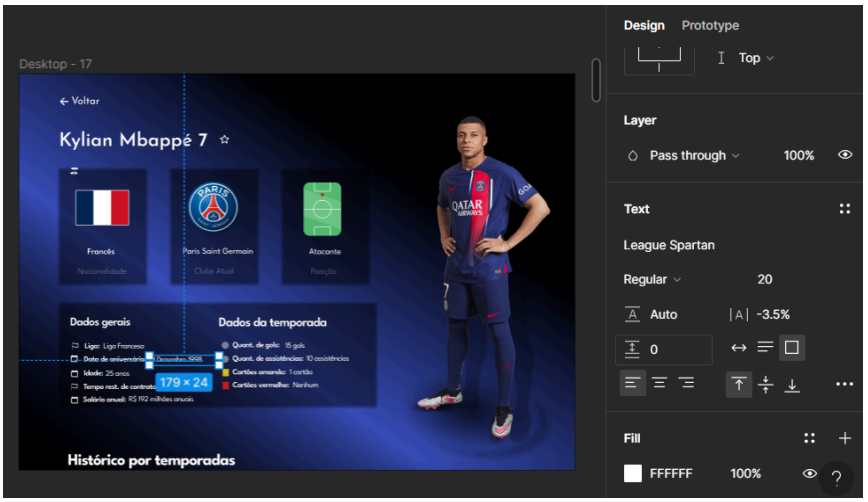


Fonte: Autor

Em relação às interações e animações, o Figma oferece uma ampla variedade de opções, desde simples arrastar e soltar até animações mais complexas, permitindo a criação de experiências de usuário dinâmicas e cativantes.

Por fim, o módulo "*Dev Mode*" do Figma simplifica a colaboração entre *designers* e desenvolvedores, como evidenciado na Figura 4, na qual esta ferramenta fornece informações essenciais para a implementação do *design*. Ao oferecer medidas precisas e especificações de estilo, o "*Dev Mode*" agiliza o processo de desenvolvimento, garantindo uma comunicação eficaz entre equipes, otimizando o tempo e a qualidade do produto final.

Figura 4 - Detalhes para Desenvolvedores no Figma



Fonte: Autor

Na Zeeway, o Figma desempenha um papel fundamental em várias etapas do processo de desenvolvimento, desde a análise de requisitos e prototipação até o suporte aos desenvolvedores durante a fase de implementação. Sua utilização para criar *designs* de média e alta fidelidade, validar conceitos com os clientes e fornecer informações precisas para a equipe de desenvolvimento demonstra o valor e a utilidade dessa ferramenta no contexto empresarial.

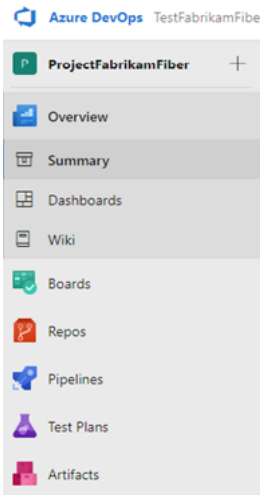
3.5 Azure DevOps

Para a escrita desta seção, o artigo disponível em Microsoft (2024) foi usado como referência. O Azure DevOps é uma plataforma abrangente que engloba

controle de versão, geração de relatórios, gerenciamento de requisitos, gestão de projetos, compilações automatizadas, testes e recursos de gerenciamento de versões.

Ele fornece um conjunto integrado de serviços e ferramentas para gerenciar projetos de software, como exemplificado na Figura 5. Utilizando um modelo cliente/servidor, a maioria dos serviços pode ser acessada através de uma interface *web*, disponível em todos os principais navegadores. Além disso, alguns serviços, como controle de origem, *pipelines* de compilação e rastreamento de trabalho, também podem ser acessados através de um cliente dedicado.

Figura 5 - Serviços do Azure DevOps



Fonte: Microsoft (2024)

Dentre os principais serviços oferecidos pelo Azure DevOps, destacam-se: *Repos*, *Boards*, *Pipelines* e *Test plans*.

Repos: este serviço fornece sistemas de controle de origem, como Git ou *Team Foundation Version Control* (TFVC), permitindo que os desenvolvedores colaborem no código e rastreiem as alterações feitas na base de código. O controle do código-fonte é fundamental para projetos que envolvem múltiplos desenvolvedores, permitindo a organização de arquivos em pastas, ramificações e repositórios.

Boards: os projetos de desenvolvimento de software exigem maneiras eficazes de compartilhar informações e rastrear o *status* do trabalho. O serviço

Boards oferece ferramentas úteis para planejamento e acompanhamento do trabalho, permitindo a organização de tarefas através de quadros.

Pipelines: a automação é essencial para garantir o lançamento rápido e confiável do software. O serviço *Pipelines* suporta a automação de compilação, teste e lançamento, possibilitando a definição de compilações que são executadas automaticamente sempre que houver alterações no código, bem como a gestão da implantação em ambientes de desenvolvimento ou produção.

Test Plans: este serviço oferece suporte à criação e gerenciamento de testes manuais, exploratórios e contínuos. Ele permite a rastreabilidade completa desde requisitos até casos de teste e *bugs*, facilitando a execução e acompanhamento de testes em qualquer plataforma.

Tudo isso faz do Azure DevOps uma ferramenta extremamente eficiente para o gerenciamento de projetos de software. Nesse contexto, o Azure DevOps é uma ferramenta integral para o gerenciamento de projetos na Zeeway. Sua adoção é uma prática padrão em todos os projetos internos, sendo utilizada desde a fase inicial de planejamento até a entrega do produto final. Através da plataforma abrangente oferecida pelo Azure DevOps, a equipe da Zeeway consegue coordenar eficientemente todas as etapas do ciclo de vida do desenvolvimento de software, garantindo uma gestão eficaz e resultados de alta qualidade.

4 ATIVIDADES REALIZADAS

Neste capítulo, é apresentado o detalhamento das atividades executadas pelo autor como Análise de Requisitos e *Product Owner* na Zeeway.

A Zeeway adota o ciclo de desenvolvimento de software delineado por Magela (2006), abrangendo as etapas essenciais de Requisitos, Projeto, Codificação, Teste e Manutenção. Dentro dessa estrutura, o Analista de Requisitos trabalha nas fases de Requisitos e Projeto, enquanto o PO atua nas etapas de Codificação, Teste e Manutenção. Nesse contexto, ao término da fase de Projeto, o Analista de Requisitos entrega ao PO elementos essenciais de seu trabalho, como o *design* de alta fidelidade e os requisitos não funcionais. Essa entrega habilita o PO a conduzir o projeto até sua conclusão, mantendo a continuidade do desenvolvimento com base nos fundamentos estabelecidos.

4.1 Análise de Requisitos

Nesta seção, será apresentado o processo de Análise de Requisitos adotado pela Zeeway que, conforme explicado anteriormente, compreende as fases de Requisitos e Projeto da abordagem proposta por Magela (2006). O diagrama da Figura 6 oferece uma visão panorâmica desse processo, que se inicia com a recepção do e-mail inicial do time comercial pelo Analista de Requisitos e segue pelas etapas de solicitação de documentos complementares, elaboração de *sketch*, desenvolvimento de *design* de média fidelidade, definição de requisitos não funcionais, proposta comercial e finalização da prototipação com *design* de alta fidelidade.

Ao término deste fluxo, têm-se uma série de documentos para o prosseguimento do projeto, incluindo o *design* de alta fidelidade e requisitos não funcionais, que são fundamentais para as próximas etapas como mencionado acima.

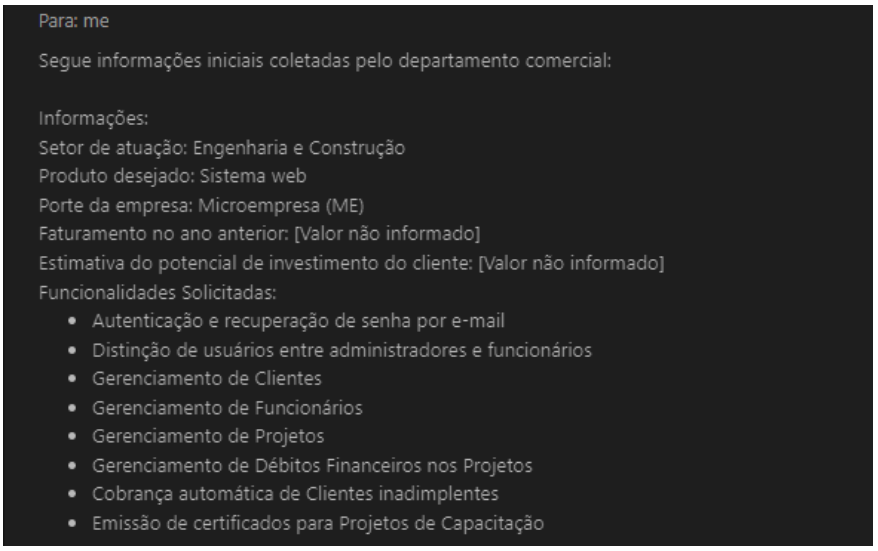
Figura 6 - Processo de Análise de Requisitos

Fonte: Autor

4.1.1 Recebimento do Email Inicial

A análise de requisitos se inicia durante o processo de vendas, quando o departamento comercial repassa ao Analista de Requisitos as solicitações dos potenciais clientes. É neste momento que se dá o pontapé inicial dos trabalhos. Geralmente, as informações coletadas com os clientes são encapsuladas e enviadas em um email conciso, elaborado pelo time comercial, destacando as principais características e funcionalidades desejadas pelo cliente, um exemplo típico de email está contida na Figura 7.

Figura 7 - Email enviado pelo Comercial


Para: me

Segue informações iniciais coletadas pelo departamento comercial:

Informações:
Setor de atuação: Engenharia e Construção
Produto desejado: Sistema web
Porte da empresa: Microempresa (ME)
Faturamento no ano anterior: [Valor não informado]
Estimativa do potencial de investimento do cliente: [Valor não informado]
Funcionalidades Solicitadas:

- Autenticação e recuperação de senha por e-mail
- Distinção de usuários entre administradores e funcionários
- Gerenciamento de Clientes
- Gerenciamento de Funcionários
- Gerenciamento de Projetos
- Gerenciamento de Débitos Financeiros nos Projetos
- Cobrança automática de Clientes inadimplentes
- Emissão de certificados para Projetos de Capacitação

Fonte: Autor

4.1.2 Solicitação de Documentação Complementar

Após receber o email, o Analista de Requisitos solicita ao cliente o fornecimento de dados para embasar a análise de requisitos. Esses dados podem incluir documentos, formulários e sistemas atualmente em uso, a fim de auxiliar na elaboração de uma ideia inicial para o produto de software. No exemplo mencionado anteriormente, foram requisitados os seguintes itens: uma planilha Excel contendo os dados armazenados dos clientes e funcionários, uma cópia de um projeto completo, bem como de todos os tipos de projetos conduzidos pela empresa, e todos os tipos de certificados emitidos.

Essas informações desempenham um papel crucial na determinação dos requisitos funcionais do sistema. Esses requisitos são responsáveis por descrever as funcionalidades específicas e os comportamentos esperados do sistema, além de identificar os dados que serão manipulados e processados. Na Zeeway, a formalização dos requisitos funcionais não segue um padrão convencional de documento, mas sim através do *design* de média fidelidade. Quando necessário estabelecer regras específicas, estas são incorporadas diretamente no *design*, por exemplo, para a funcionalidade de cobrança automática de clientes inadimplentes, é definido que o sistema deve enviar e-mails a cada 15 dias até que o pagamento seja efetuado, sendo esta regra descrita em texto ao lado da tela, como exemplificado na Figura 8.

Figura 8 - Exemplo de Definição de Requisito

Fonte: Autor

Este *design*, por sua vez, assume o papel de documento de requisitos ao ser anexado ao contrato, estabelecendo com clareza o escopo do trabalho a ser executado pela equipe. Essa abordagem é justificada pela coleta detalhada dos requisitos nesta fase, sendo o *design* de média fidelidade capaz de visualmente representar como o sistema operará com base nesses requisitos, facilitando a compreensão de todas as partes envolvidas.

Seguindo com o exemplo mencionado anteriormente, no contexto dos documentos de projetos solicitados, a análise dessas informações é fundamental para compreender quais dados serão armazenados, especialmente dada a diversidade de projetos existentes na empresa, como projetos de obras e de treinamento. Através dessa análise detalhada, é possível identificar quais campos são compartilhados entre ambos os tipos de projetos e quais são específicos de cada um, o que orienta de forma precisa o planejamento subsequente.

Além disso, a solicitação dos certificados é relevante para entender o conteúdo padrão e a variação de elementos presentes nesses documentos, bem como os padrões adotados. Essas informações são cruciais para garantir a adequação do sistema às necessidades específicas da empresa e dos clientes.

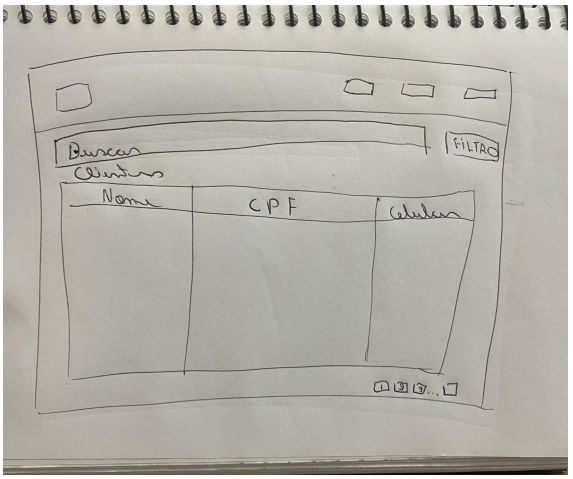
4.1.3 Elaboração do *Sketch*

Com base nas informações coletadas, é iniciada a etapa de prototipação, na qual o *designer* e o Analista de Requisitos colaboram para elaborar um esboço da solução inicial. Este esboço, muitas vezes feito à mão e conhecido como *sketch*, delineia a estrutura que o site ou sistema seguirá. Ele atua como ponto de partida crucial para o desenvolvimento subsequente, oferecendo uma representação visual e simplificada do projeto, como exemplificado na Figura 9. Nesse estágio, o protótipo não apresenta detalhes de informações e *design*, focando-se exclusivamente na estrutura básica para iniciar os trabalhos.

Em seguida, é agendada uma reunião com o potencial cliente para validar o *sketch*, garantindo que corresponda à estrutura esperada e faça sentido para a solução proposta. Durante a reunião, geralmente são solicitados ajustes que são feitos em tempo real em colaboração com o cliente. Ao final da reunião, é obtida

uma versão aprovada do *sketch*.

Figura 9 - Esboço de Tela Feito à Mão



Fonte: Autor

4.1.4 Elaboração do *Design* de Média Fidelidade

Após esse passo, o *sketch* é convertido em um *design* de média fidelidade no Figma (Seção 3.3) pelo *designer*. Esse *design*, conforme representado na Figura 10, concentra-se principalmente no fluxo do usuário e na arquitetura da informação, não tendo a estética como sua prioridade principal.

Figura 10 - Exemplo de *Design* de Média Fidelidade

Fonte: Autor

Durante a elaboração do *design* de média fidelidade, primeiramente é realizada a tradução do esboço aprovado pelo cliente. Em seguida, o Analista de

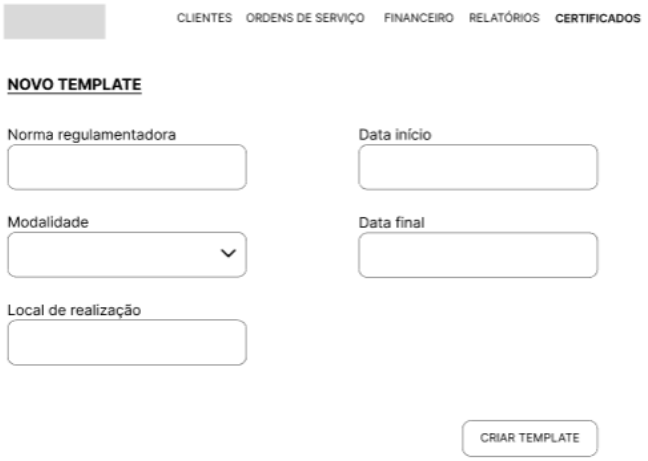
Requisitos, juntamente com o *designer*, buscam maneiras de otimizar o fluxo de trabalho proposto pelo cliente, utilizando sua experiência na construção de plataformas em prol do mesmo. Este é um processo criativo que depende diretamente da capacidade do profissional em questão.

Para exemplificar o que foi escrito acima, será discutido um exemplo desse trabalho. No caso em análise desde o início desta seção, o cliente solicitou uma funcionalidade de geração automática de certificados, sem a necessidade de criar um novo do zero cada vez que for necessário. Logo, ele expressou o desejo de cadastrar modelos que foram chamados de *templates*, onde todos os certificados possuem um texto padrão que não muda em nenhum caso e em cada *template* é possível alterar as variáveis, como a norma regulamentadora, modalidade, data de início, data de término e local de realização. Em seguida, a ideia era simplesmente clicar em "usar *template*", conforme exemplificado nas Figuras 11 e 12, inserir os dados do destinatário do certificado e gerá-lo.

Figura 11 - *Design* de Média Fidelidade 1

Fonte: Autor

Figura 12 - *Design* de Média Fidelidade 2



Fonte: Autor

O Analista de Requisitos identificou uma área de melhoria nessa funcionalidade. Foi observado que as turmas de certificação podem ter até 50 pessoas e ocorrer várias vezes ao longo do mês, tornando o processo de emissão de certificados bastante oneroso. Portanto, o Analista de Requisitos sugeriu uma melhoria na solução proposta, na qual, ao clicar em "usar *template*", seria possível inserir os dados de toda a turma de uma vez, conforme exemplificado na Figura 13, e os certificados seriam gerados em lote, sendo baixados em formato zip, compactados e agrupados.

Figura 13 - Geração de Certificados em Bloco

Fonte: Autor

A solução inicial já aumentaria a eficiência da equipe responsável pela geração dos certificados. No entanto, com a experiência e o olhar atento da equipe nesse processo, foi introduzida uma opção que consideravelmente ampliaria a eficiência e melhoraria a experiência do usuário, tornando o uso da ferramenta menos trabalhoso e mais fluido.

Após a finalização dessa versão, uma reunião foi agendada com o potencial cliente para validar o *Design* de Média Fidelidade, onde todos os fluxos criados pela equipe foram apresentados. Durante essa reunião, é comum o cliente solicitar alguns ajustes, os quais são registrados, implementados e posteriormente validados novamente com o cliente. Esse processo se repete até que o cliente esteja plenamente satisfeito de que o que está sendo apresentado faz sentido e representa a versão final desejada para sua empresa.

4.1.5 Elaboração dos Requisitos Não Funcionais

Após a aprovação do design de média fidelidade o Analista de Requisitos elucida os requisitos não funcionais, que abrangem as qualidades ou atributos do sistema, tais como desempenho, segurança, usabilidade e escalabilidade. Nessa etapa, o Analista de Requisitos se reúne com o CTO, o executivo responsável por supervisionar a estratégia tecnológica da organização, e o potencial cliente para coletar informações cruciais relacionadas ao desempenho, segurança e outros aspectos relevantes da ferramenta.

Dentro do exemplo discutido, um requisito não funcional de importância crítica foi a necessidade de registrar as atividades de cada usuário no sistema, visando à auditoria e à identificação precisa de possíveis causadores de problemas. A responsabilidade pela coleta desses requisitos não funcionais recai principalmente sobre o CTO, cujo papel é essencial na definição da estratégia tecnológica e na identificação dos aspectos cruciais para o sucesso do projeto, enquanto o analista acompanha para documentar as decisões tomadas. Para facilitar esse processo, é utilizada uma planilha, conforme exemplificado na Figura 14. Essa planilha é posteriormente anexada ao contrato, garantindo a proteção e alinhamento de ambas as partes envolvidas.

Figura 14 - Tabela de Requisitos Não Funcionais

Código	Descrição	Grau de Importância
RNF - 001	Todos os dados sensíveis dos usuários como senha, documentos e endereço devem ser criptografados.	Alta
RNF - 002	Apenas usuários autorizados devem ter permissão para acessar o sistema.	Alta
RNF - 003	Todas as atividades do usuário devem ser registradas para fins de auditoria.	Alta
RNF - 003	O sistema deve estar disponível 98% do tempo de operação.	Média
RNF - 004	O sistema deve fazer backup dos dados todo dia às 18h.	Alta
RNF - 005	O sistema deve ser projetado para suportar um aumento de até 1000% no número de usuários simultâneos dentro de um período de 2 anos, sem comprometer o desempenho.	Alta
RNF - 006	O código-fonte e a documentação do sistema devem ser claros e bem documentados para facilitar futuras atualizações e manutenção.	Baixa

Fonte: Autor

4.1.6 Elaboração da Proposta Comercial

Nesse momento, o Analista de Requisitos assume uma função de apoio, disponível para esclarecer dúvidas, enquanto a equipe comercial da Zeeway já dispõe do *design* de média fidelidade, que representa os requisitos funcionais, e a tabela contendo os requisitos não funcionais. O CTO então determina a composição da equipe necessária para a construção da plataforma, estabelece prazos, faz estimativas de custos de infraestrutura e qualquer outra despesa pertinente. Essas informações são então encaminhadas ao departamento comercial. Com base nesses dados, o setor comercial elabora uma proposta e a apresenta ao cliente. Se o cliente não aprovar a proposta, o processo serve como aprendizado para futuras propostas, uma vez que todas as informações são documentadas, agilizando processos subsequentes.

No caso de aprovação do projeto pelo cliente, o departamento comercial toma as medidas necessárias junto ao departamento jurídico. Com tudo pronto, eles dão o sinal verde para avançar para a etapa final da prototipagem: o *design* de alta fidelidade.

4.1.7 Elaboração do *Design* de Alta Fidelidade

Com a aprovação da equipe comercial, o Analista assume o papel de

representante do cliente junto ao *designer*, transmitindo todas as diretrizes e validando cada tela proposta. Dessa forma, o *designer* inicia a elaboração do *design* de alta fidelidade, que representa a etapa mais avançada no processo de *design* de produtos ou sistemas digitais. Nessa fase, são desenvolvidas representações visuais altamente detalhadas e precisas do produto final, englobando elementos como *layout*, cores, tipografia, ícones e interações. A riqueza de detalhes presente, como exemplificado na Figura 15, proporciona uma compreensão nítida e abrangente de como será o produto finalizado às partes interessadas.

Figura 15 - Exemplo de *Design* de Alta Fidelidade

Fonte: Autor

Nesse contexto, o *design* de alta fidelidade permite a realização de testes de usabilidade mais eficazes. Os usuários podem interagir de forma mais realista com o *design*, navegando pelo fluxograma do Figma e compreendendo exatamente para onde cada ação os leva. Essa interatividade facilita a identificação de problemas de usabilidade, promovendo aprimoramentos que resultam em uma experiência do usuário mais satisfatória.

Essa etapa também desempenha um papel crucial na validação do conceito junto às partes interessadas. Ao apresentar um *design* minucioso, que se assemelha ao produto final, é possível assegurar que o produto esteja alinhado com as expectativas e necessidades do cliente, antes mesmo de alocar recursos significativos para o desenvolvimento. Além disso, a proximidade do *design* com o produto final minimiza a probabilidade de grandes alterações no projeto, uma vez

que reflete de forma precisa o sistema. Isso não apenas previne retrabalho futuro, mas também otimiza a alocação de recursos, promovendo uma abordagem mais eficiente e econômica.

Por fim, o *design* de alta fidelidade facilita a comunicação entre as equipes de desenvolvimento, *designers*, partes interessadas e clientes. Ao fornecer uma representação visual precisa do produto a ser desenvolvido, promove uma compreensão clara e alinhada de todos os envolvidos. Isso reduz a probabilidade de mal-entendidos e conflitos, resultando em um processo de desenvolvimento mais eficiente e colaborativo.

Após a conclusão dessa etapa, é agendada uma reunião com o cliente para validar o *Design* de Alta Fidelidade, durante a qual o projeto é apresentado de forma abrangente, detalhando precisamente como o sistema será estruturado. Durante esse encontro, é comum que o cliente solicite ajustes, os quais são minuciosamente registrados, implementados e submetidos à validação novamente pelo cliente. Esse processo se repete até que o cliente esteja completamente satisfeito de que o que está sendo apresentado corresponde à visão final desejada para sua empresa.

Após a confirmação, é elaborado um termo de aceitação, no qual o cliente declara estar ciente de que a solução será desenvolvida conforme o *design* apresentado e que quaisquer alterações futuras serão objeto de acordo adicional, sujeitas a encargos comerciais adicionais. Esse procedimento visa proteger a *Zeeway* de possíveis contratempos decorrentes de mudanças nos requisitos ou nos fluxos de trabalho acordados, reduzindo consideravelmente a ocorrência de solicitações de alterações durante o desenvolvimento do software.

A assinatura do termo de aceitação marca o término da etapa de Requisitos e o início da etapa de Projeto. Nesta etapa, predominantemente liderada pelo Líder Técnico da equipe em colaboração com o CTO da empresa, após a entrega dos documentos gerados na etapa anterior, são realizadas atividades como a montagem do banco de dados, definição da arquitetura do sistema, criação de repositórios e demais aspectos relacionados. O autor, enquanto Analista de Requisitos, não participa diretamente dessas atividades, mas fica disponível para esclarecer dúvidas e fornecer informações adicionais.

Com a conclusão da etapa de projeto e o encerramento das atividades

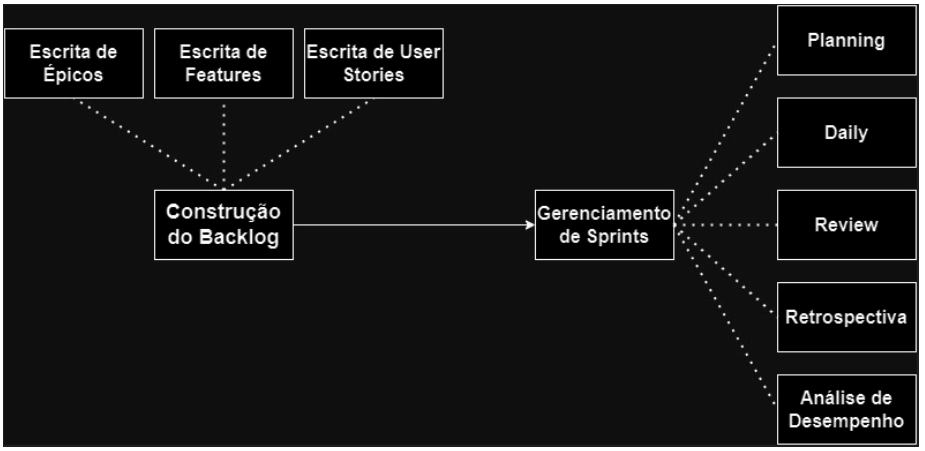
realizadas pela equipe técnica, o Analista de Requisitos conduz uma reunião com o Líder Técnico do projeto, o CTO e o PO. Durante esse encontro, são compartilhadas todas as informações, documentos e aspectos pertinentes ao projeto. Este momento marca o encerramento das responsabilidades do Analista de Requisitos e o início do papel do PO.

4.2 Gestão de Projetos como *Product Owner*

Após a conclusão da fase de projeto, dá-se início às etapas de Codificação e Teste. Na Zeeway, essas duas fases são executadas em paralelo devido à adoção da metodologia ágil *Scrum*. Durante um *sprint,* as atividades são progressivamente finalizadas e disponibilizadas para testes unitários pelo time de *Quality Assurance*. Ao término do *sprint*, o *Quality Assurance* testa o incremento de forma abrangente e realiza testes regressivos no sistema para garantir a integridade do sistema após a atualização. Dessa forma, as etapas de codificação e teste se entrelaçam no processo de desenvolvimento.

Nesta seção, serão apresentadas as atividades realizadas pelo autor enquanto PO da Zeeway. O diagrama da Figura 16 oferece uma visão panorâmica desse processo, que se inicia com a construção do *backlog* (Escrita de Épicos, *Features* e *User Stories*) e gerenciamento de *sprints* (Realização de *Planning, Daily, Review, Retrospective* e Análise de Desempenho).

Figura 16 - Processo de Gestão de Projetos



Fonte: Autor

4.2.1 Construção do *Backlog*

Na etapa de Análise de Requisitos, são produzidos o *design* de alta fidelidade e o documento de requisitos não funcionais, os quais constituem a base para a construção do *backlog* do projeto. Na Zeeway, adota-se a organização entre épicos, *features* e *user stories*, com base nas diretrizes propostas por (Aguiar; Caroli, 2021). Os épicos representam grandes funcionalidades ou conjuntos de funcionalidades que não podem ser entregues em um único *sprint*, geralmente refletindo objetivos amplos do projeto. As *features* representam funcionalidades específicas do sistema que agregam valor para o usuário e podem ser entregues em um *sprint*. Já as *user stories* são descrições de funcionalidades do ponto de vista do usuário. Dentro da Zeeway, as *user stories* são sempre escritas com suporte visual e critérios de aceitação, os quais estabelecem os requisitos e parâmetros que a tarefa deve atender para ser considerada finalizada.

Para uma compreensão mais clara desses conceitos e aplicações, será apresentado um exemplo. A Figura 17 ilustra o épico “Financeiro”, integrante de um sistema de gestão de projetos de uma empresa agrícola, com diversas funcionalidades. Este épico em particular destina-se a gerenciar toda a parte financeira da empresa, incluindo a emissão de boletos, controle de caixa, notificações de previsão, controle de pagamentos, entre outros. Dessa forma, devido à quantidade e complexidade de suas funcionalidades, não é possível finalizá-lo em um único *sprint*, o que o caracteriza como um épico.

Figura 17 - Exemplo de Épico

Fonte: Autor

Dentro deste épico, destaca-se a *feature* “Notificação de Previsão” ilustrada na Figura 18, uma funcionalidade específica e singular. Em contraste com o que foi explicado anteriormente, esta *feature* é de tamanho reduzido e não apresenta um

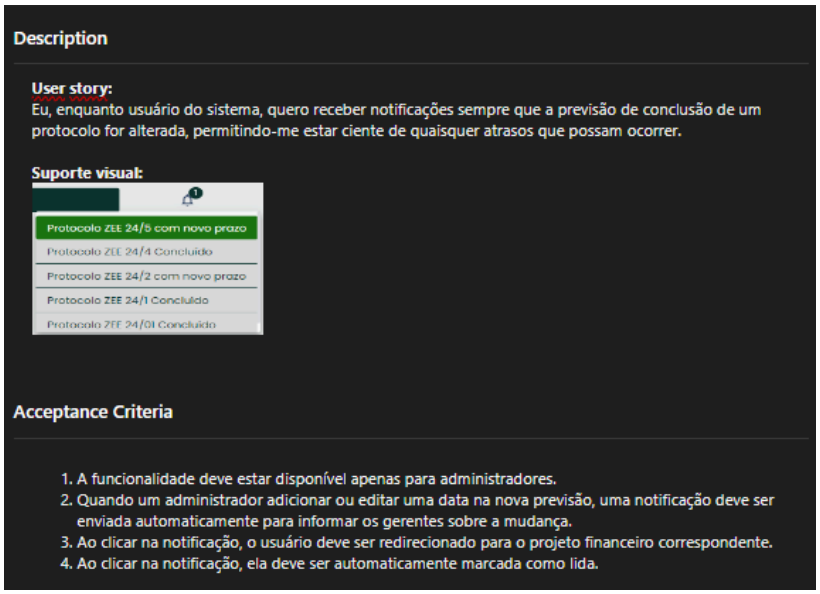
grau de complexidade elevado, o que a torna passível de ser entregue em um único *sprint*.

Figura 18 - Exemplo de *Feature*

Fonte: Autor

Dentro dessa *feature*, são definidas as *user stories,* sendo que cada *feature* pode englobar uma ou mais delas. Cada *user story* possui sua descrição detalhada, suporte visual da funcionalidade para facilitar o desenvolvimento e critérios de aceitação específicos, que deixam claro os requisitos e parâmetros que precisam ser entregues para a user story ser dada como finalizada, assim como exemplificado na Figura 19. Na Zeeway, as *user stories* de cada *feature* são subdivididas em *user stories* de backend, *frontend* e *mobile*. Essa abordagem permite que cada equipe assuma a responsabilidade por suas respectivas *user stories*, garantindo que a *feature* seja concluída somente quando todas as *user stories* associadas forem finalizadas.

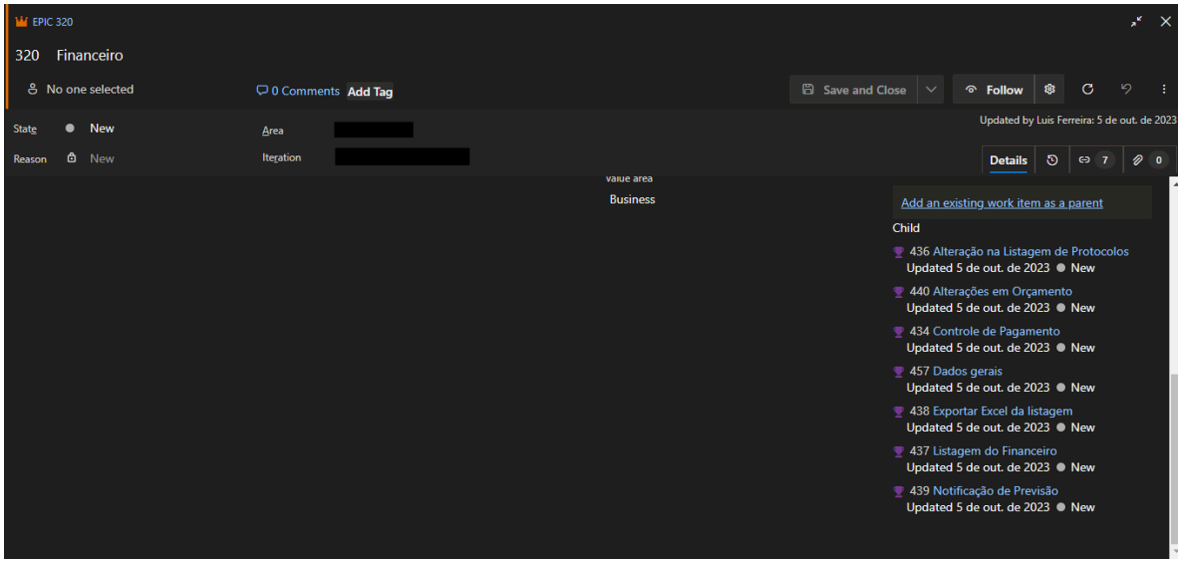
Figura 19 - Exemplo de *User Story*



Fonte: Autor

Além disso, uma das funcionalidades usadas na *Zeeway*, viabilizada pelo Azure DevOps, é a gestão de relacionamentos. Essencialmente, as *user stories* estão associadas às *features*, que, por sua vez, estão vinculadas aos épicos. Essa estruturação é exemplificada na Figura 20, onde o épico “Financeiro” é ilustrado com suas *childs*, as *features* “Dados Gerais”, “Controle de Pagamento” e outras, destacando a relação entre eles. Nesse contexto, o épico é o elemento pai da *feature*, que por sua vez é pai da *user story*. Essa estruturação permite uma clara hierarquia de tarefas e facilita a organização do trabalho. Além disso, o Azure DevOps automatiza o fechamento de tarefas: quando todas as *user stories* de uma *feature* são marcadas como concluídas, a própria *feature* é automaticamente encerrada. O mesmo princípio se aplica às *features* dentro de um épico.

Figura 20 - Exemplo de Relacionamento



Fonte: Autor

Nesse contexto, o papel do PO começa com a análise do *design* de alta fidelidade e do documento de requisitos não funcionais, a partir dos quais ele elabora um *backlog* abrangente, seguindo a estrutura de divisão explicada. Inicialmente, são criados todos os épicos, seguidos pelas respectivas *features* e, por fim, todas as *user stories*, cada uma com suas descrições, suporte visual e critérios de aceitação. Todo esse processo é conduzido na plataforma Azure DevOps. Ao concluir essa etapa, o time está pronto para receber as tarefas e iniciar o

desenvolvimento do sistema.

4.2.1 Gerenciamento de *Sprints*

Após a elaboração completa do *backlog*, os desenvolvedores ingressam no processo e a implementação efetiva do sistema tem início. É neste ponto que se adota o *framework Scrum* para gerenciar o projeto. Inicialmente, o PO convoca uma reunião de início de projeto com todo time *Scrum*. Nesse encontro, é apresentado o contexto geral do projeto, incluindo informações sobre o cliente, o setor de atuação e a natureza do projeto. São também discutidos os motivos que levaram à sua realização, bem como as expectativas em relação ao mesmo. Além disso, são alinhados os rituais de *planning, daily, review e retrospective*, bem como os dias e horários em que ocorrerão.

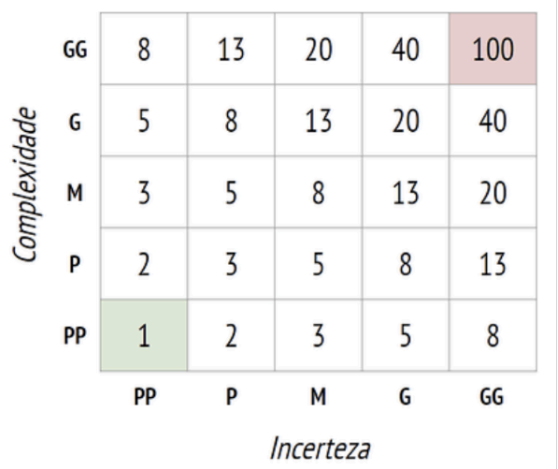
Com todos os detalhes alinhados, o PO inicia o planejamento do primeiro *sprint*. O primeiro passo é definir o *sprint backlog*, ou seja, os itens do *backlog* que serão priorizados para inclusão no *sprint*, determinando assim a duração do *sprint*. Na Zeeway, a orientação é que o *backlog* do *sprint* seja dimensionado para caber dentro de um *sprint* de 2 semanas. Além disso, é crucial estabelecer o *Sprint goal*, ou seja, o objetivo específico a ser alcançado ao final do *sprint*. Com todos esses elementos em mãos, a *planning* pode ser conduzida.

Durante a *planning*, o PO compartilha com o time *Scrum* o *Sprint goal* e a duração do *sprint*. Em seguida, cada *user story* a ser entregue ao final do *sprint* é revisada pela equipe, que atribui pontos de complexidade e incerteza a cada uma, usando a matriz da Figura 21 como base. Esses pontos são determinados considerando o quão difícil é a *user story* (complexidade) e o quão bem compreendida está pelos membros da equipe (incerteza). Na *Zeeway*, há uma regra que limita as *user stories* a no máximo 13 pontos, caso ultrapassem esse limite, devem ser divididas em mais de uma.

Durante a etapa de pontuação, todos os membros da equipe atribuem simultaneamente pontos a cada tarefa. Se houver divergências nos pontos atribuídos, os desenvolvedores dialogam entre si para esclarecer as razões das discrepâncias e revisam suas decisões até que um consenso seja alcançado. Essa

prática é essencial para detectar possíveis lacunas no entendimento das tarefas e garantir que todos os membros tenham uma visão compartilhada do esforço necessário para sua conclusão.

Figura 21 - Matriz de pontuação de *User Stories*



Fonte: Silvério (2019)

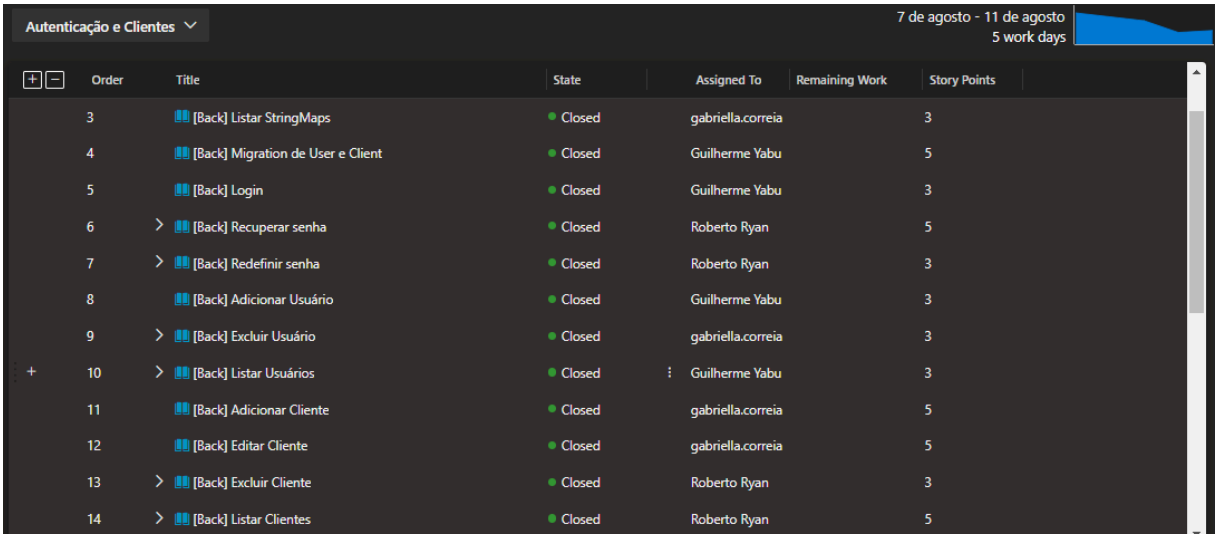
Com as tarefas devidamente pontuadas, o PO solicita que a equipe as distribua entre si, considerando não apenas as pontuações atribuídas, mas também a experiência e a senioridade de cada membro. Isso é exemplificado na Figura 22, onde, por exemplo, o desenvolvedor “Guilherme” ficou com 14 pontos, enquanto “Gabriella” e “Roberto” ficaram com 16, demonstrando uma distribuição justa e equilibrada das responsabilidades. Após essa etapa, a *planning* é concluída e a equipe dá início ao desenvolvimento das tarefas atribuídas.

Após a *planning*, todos os dias de trabalho da *sprint* são marcados pela realização da *daily*, uma reunião breve em que todo o time *Scrum* responde a três perguntas fundamentais: “O que eu fiz desde a última *daily*?”, “O que pretendo realizar até a próxima *daily*?” e “Existem quaisquer impedimentos que estejam afetando meu progresso?”.

Essas perguntas fornecem uma visão clara do progresso de cada membro da equipe e identificam rapidamente quaisquer obstáculos que possam estar surgindo. Esse processo promove um ambiente de colaboração, onde os membros da equipe estão prontos para se ajudar mutuamente. Se um membro enfrenta dificuldades, os

demais membros se colocam à disposição para oferecer assistência, garantindo assim que ninguém fique estagnado em problemas não resolvidos.

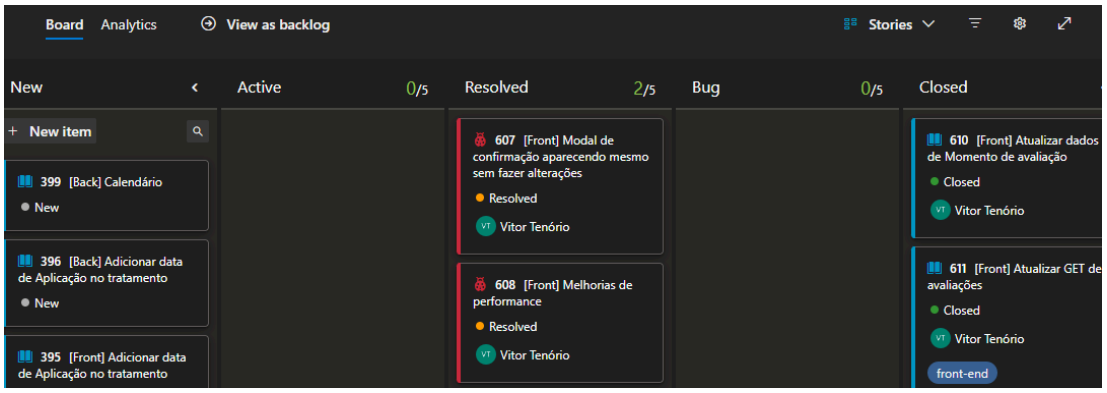
Figura 22 - Exemplo de Distribuição de Tarefas



Fonte: Autor

Além da *daily*, o PO conta com um quadro *Kanban* para visualizar o progresso da equipe, o Azure DevOps possibilita a criação desse quadro, conforme o modelo adotado pela Zeeway, que inclui colunas como "*New*", "*Active*", "*Resolved*", "*Bug*" e "*Closed*", conforme exemplificado na Figura 23.

Figura 23 - Quadro *Kanban* dentro do Azure DevOps



Fonte: Autor

- "*New*": Nesta coluna, são listadas as tarefas da *sprint* que ainda não

foram iniciadas.

- "*Active*": Aqui estão as tarefas que já foram iniciadas.
- "*Resolved*": Esta coluna é dedicada às tarefas finalizadas pelos desenvolvedores, prontas para serem testadas.
- "*Closed*" ou "*Bug*": Após o teste, as tarefas são movidas para a coluna "*Closed*" se estiverem livres de erros ou para a coluna "*Bug*" se houver algum problema identificado.

Na *Zeeway*, todos os desenvolvedores utilizam esse quadro e atualizam suas atividades conforme o andamento, o que possibilita monitorar o progresso da *sprint* de forma eficiente, identificar possíveis gargalos e obter uma visão abrangente de todas as atividades em andamento no time e no projeto.

Além disso, o Azure DevOps fornece o gráfico *burndown*, uma ferramenta valiosa para acompanhar o progresso do projeto. No gráfico *burndown*, o eixo vertical representa a quantidade de trabalho restante, geralmente medido pela pontuação das tarefas, enquanto o eixo horizontal indica o tempo.

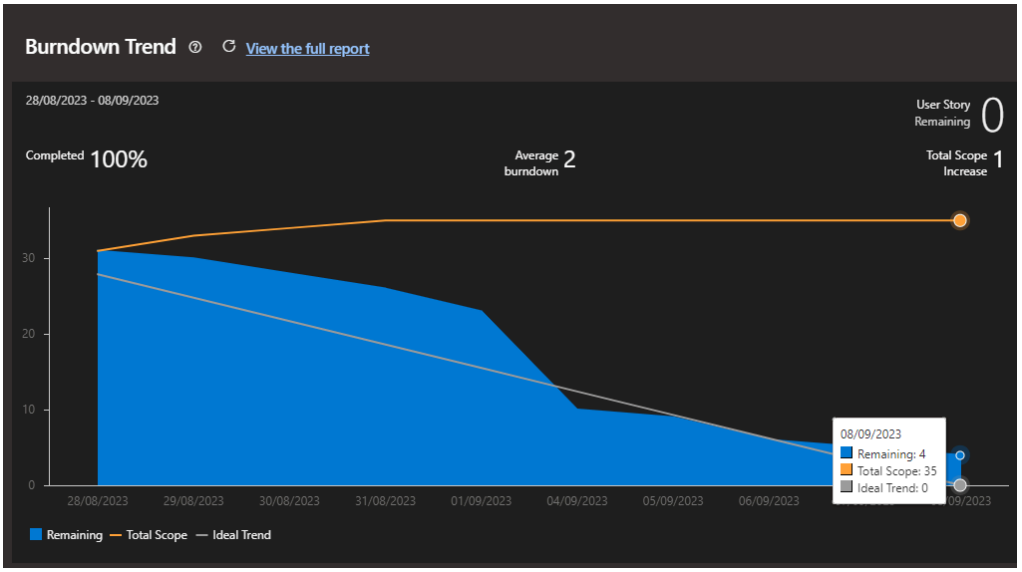
Idealmente, a linha do gráfico *burndown* inicia no ponto máximo, representando todo o trabalho planejado para o projeto, e diminui gradualmente à medida que o tempo avança e as tarefas são concluídas. O objetivo é que a linha alcance zero no final do *sprint*, indicando que todo o trabalho planejado foi concluído.

A análise do gráfico *burndown* fornece *insights* valiosos sobre o andamento do projeto. Se a linha estiver abaixo do previsto, isso pode indicar um progresso mais rápido do que o esperado. Por outro lado, se a linha estiver acima do previsto, pode sugerir atrasos ou identificação de mais trabalho durante o processo. Essas informações permitem que a equipe tome medidas corretivas, se necessário, para garantir que o projeto permaneça no caminho certo.

Um exemplo ilustrativo desse gráfico pode ser observado na Figura 24. No meio do *sprint*, o time estava ligeiramente atrasado, mas conseguiu recuperar o atraso e ficar abaixo da linha ideal, indicando que estavam em dia com o cronograma. No entanto, no final do *sprint*, quatro tarefas não foram concluídas e foram transferidas para o próximo *sprint*. O papel do PO é entender o motivo dessas tarefas não concluídas para evitar recorrências. Neste exemplo, a análise revelou

que houve uma mudança no escopo, evidenciada pela diferença entre o total de tarefas que existiam no começo do *sprint* e no final, 31 e 35 respectivamente, exatamente a quantidade de tarefas não realizadas, o que destaca a importância do controle de mudanças durante o desenvolvimento do projeto.

Figura 24 - Gráfico *Burndown*



Fonte: Autor

No último dia do *sprint*, à tarde, ocorre a *review*, um momento crucial em que todos os desenvolvedores apresentam as tarefas concluídas e as funcionalidades implementadas. Durante essa sessão, cada desenvolvedor abre suas *user stories* e demonstra como foram implementadas no sistema e no código, mostrando que atendeu aos critérios de aceitação previamente estabelecidos. Essa abordagem permite que o time avalie não apenas a conclusão das tarefas, mas também a qualidade do código produzido.

Ao mostrar como as tarefas foram implementadas no código, os desenvolvedores asseguram que o código esteja bem estruturado e desenvolvido. Durante a revisão, o time colabora na identificação de problemas e sugere melhorias sempre que necessário. Essa troca de conhecimento e experiência contribui para a qualidade geral do produto, envolvendo todos os membros do time no processo de garantia de qualidade e na busca contínua pela excelência técnica.

Além disso, a *review* permite ao PO avaliar se as entregas estão em conformidade com os requisitos, descrições e critérios de aceitação, dando o veredito final sobre a conclusão das tarefas. Se houver identificação de melhorias no código, uma nova *user story* de melhoria é criada. Caso o PO determine que a tarefa não foi concluída, o desenvolvedor recebe uma advertência e deve justificar-se, enquanto o time monta uma estratégia para evitar recorrências.

Por fim, o PO avalia se o *Sprint goal* foi alcançado ou não e formaliza o incremento no sistema. O alcance do *Sprint goal* é motivo de grande satisfação para o time, demonstrando seu desempenho e comprometimento. Por outro lado, não o alcançar é motivo de frustração, exigindo que o time identifique melhorias no processo e soluções para evitar repetições, sendo este um tema abordado durante a *retrospective*.

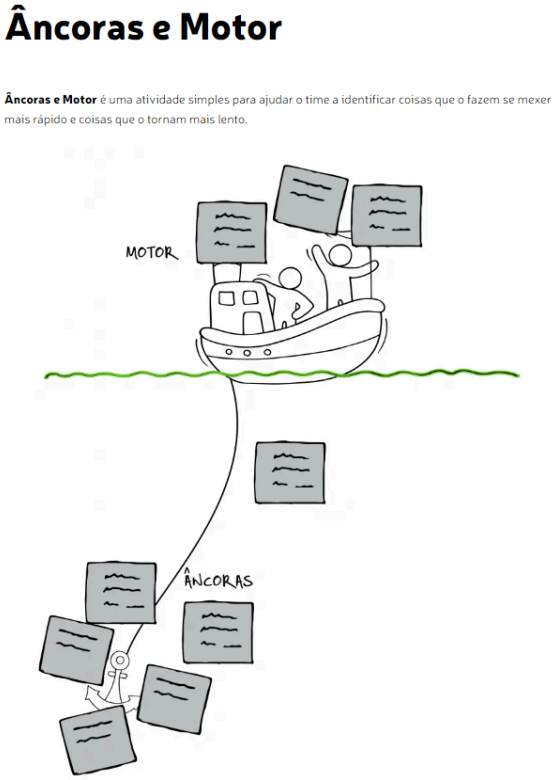
A *retrospective* se inicia meia hora após a *review* e marca o fim do *sprint*, o objetivo é refletir sobre o que funcionou bem, o que não funcionou tão bem e como a equipe pode melhorar o seu processo de trabalho.

Na Zeeway, segue-se um padrão estabelecido para as retrospectivas. Inicialmente, é realizada uma dinâmica de abertura, um momento de descontração e diversão para o time. Isso pode envolver atividades como jogos de memória ou cada membro compartilhando algo leve e descontraído sobre si, promovendo assim a intimidade entre os colegas.

Em seguida, passa-se para a dinâmica principal, onde são abordadas três perguntas fundamentais: “o que funcionou bem?”, “o que não funcionou?” e “o que podemos fazer para melhorar?”. Há uma variedade de possibilidades e dinâmicas para extrair essas informações, e na *Zeeway* foi optado por utilizar as sugestões apresentadas por Caroli (2021). Esse livro oferece diversas ideias de dinâmicas para envolver e engajar o time.

Um exemplo é a dinâmica “Âncora e Motor”, destacada na Figura 25. Esta é uma atividade simples que auxilia a equipe a identificar os fatores que impulsionam ou retardam seu progresso. O que impulsiona o progresso representa o que deu certo, enquanto o que retarda o progresso indica o que deu errado. Ao final, estratégias para resolver os obstáculos identificados, visando evitar que ocorram novamente no futuro, são discutidas.

Figura 25 - Dinâmica Âncora e Motor



Autor: Caroli, 2021

Após a conclusão da *retrospective*, o *sprint* é encerrado e o processo reiniciado, seguindo o padrão do *Scrum*. Este ciclo se repete até a conclusão do projeto, garantindo uma abordagem iterativa e incremental.

Após o incremento gerado pelo *sprint*, o PO agenda uma reunião com o cliente para apresentar todo o trabalho realizado naquele *sprint* e demonstrar o valor agregado pelo novo incremento ao sistema.

A última responsabilidade do PO é atualizar uma tabela de desempenho que registra a performance individual de cada membro da equipe. Nessa tabela, são registrados os pontos referentes às tarefas concluídas por cada desenvolvedor, levando em consideração apenas aquelas que foram validadas pelo PO na revisão, conforme ilustrado na Figura 26. Vale ressaltar que os nomes dos desenvolvedores são omitidos para preservar sua privacidade.

Figura 26 - Pontuação por *Sprint* dos Desenvolvedores

	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Desenvolvedor A	16	18	22	18
Desenvolvedor B	20	24	24	28
Desenvolvedor C	14	20	12	14

Fonte: Autor

Essa planilha desempenha um papel crucial ao auxiliar os gerentes na tomada de decisões relacionadas a promoções, aumentos salariais e até mesmo demissões. Ao fornecer uma visão clara do desempenho dos desenvolvedores, demonstrando quem está evoluindo mais e entregando os melhores resultados, ela fornece uma base objetiva para essas decisões.

A cada *sprint*, essa tabela é atualizada e enviada para o gerente de projetos, que a utiliza como ferramenta de análise e gestão. Dessa forma, ela não só ajuda a reconhecer e recompensar o bom desempenho, mas também a identificar áreas que possam precisar de desenvolvimento ou suporte adicional. Essa prática contribui para um ambiente de trabalho transparente e orientado para resultados na Zeeway.

Após a conclusão do projeto, se inicia a fase final do ciclo de vida do software: Manutenção. Esta etapa, embora simples, desempenha um papel essencial na garantia da qualidade e na satisfação contínua do cliente. Geralmente, estabelecemos com o cliente um acordo para um determinado número de horas mensais de suporte, variando entre 40 e 80 horas. Essas horas são reservadas para a correção de eventuais problemas, implementação de modificações menores e adição de novas funcionalidades conforme necessário ao longo do tempo.

É durante essa fase que a tranquilidade prevalece, visto que o produto já está implementado e em funcionamento. Na Zeeway, é comum não haver a presença ativa de um PO na equipe de manutenção. Em vez disso, o contato direto com o cliente é conduzido pelos desenvolvedores de suporte, enquanto o PO permanece disponível para esclarecer dúvidas e oferecer orientação quando necessário.

Vale ressaltar que essa fase geralmente perdura enquanto o software continua em uso. É somente quando o produto sai do mercado que ela se encerra.

5 CONSIDERAÇÕES FINAIS

Ao concluir este trabalho, o autor compartilha detalhes de suas experiências como Analista de Requisitos e *Product Owner* na Zeeway. A análise dessas atividades revela um valor significativo, demonstrando como várias metodologias, conceitos e tecnologias foram utilizadas e adaptadas para uma empresa em constante crescimento no mercado. Este relato transcende a teoria ao mostrar os desafios enfrentados no dia a dia, onde nem sempre é possível aplicar o melhor *framework* ou metodologia. No cerne, o lucro se destaca como o principal impulsionador do sistema e encontrar uma harmonia com esse aspecto fundamental da economia é crucial.

Dentre as atividades mencionadas, destaca-se a necessidade de desenvolver um documento de requisitos formal. Embora o detalhamento no Figma com o *design* de média fidelidade possa esclarecer o escopo do trabalho, um documento formal e numerado é de suma importância. A Zeeway já planeja implementar essa prática em projetos futuros de maior envergadura, embora permaneça um desafio para projetos de menor porte.

Adicionalmente, a adoção da pré-*planning* é sugerida para a empresa. Esse ritual, complementar ao *Scrum*, permite que a equipe se reúna dois dias antes da *planning* para revisar as tarefas a serem incluídas no *sprint*. Dessa forma, os desenvolvedores podem sugerir modificações e antecipar possíveis problemas, concedendo ao PO o tempo necessário para ajustar as *user stories* e alinhar os objetivos de acordo com o *feedback* coletivo da equipe, evitando mudanças após o começo do *sprint*.

Nesse contexto, vale ressaltar que a Universidade Federal de Lavras desempenhou um papel fundamental no apoio ao desenvolvimento dessas atividades. Nas disciplinas de Engenharia de Software, Sistemas de Informação, Processos de Software, Gestão de Tecnologia da Informação e Gerência de Projetos de Software, os professores apresentaram uma base teórica sólida. Combinando os conceitos apresentados em aula, com a experiência prática no mercado, a universidade proporcionou ao autor uma formação profissional e educacional de qualidade excepcional.

Considerando o trabalho apresentado, seria interessante em trabalhos futuros coletar dados sobre mudanças nos requisitos. Comparar o número de mudanças de requisitos entre projetos que possuem apenas o *design* de média fidelidade anexado ao contrato e aqueles que possuem um documento formal de requisitos, isso ajudaria a avaliar o impacto dessa prática na gestão de requisitos.

Além disso, a coleta de métricas comparativas pode fornecer uma compreensão mais aprofundada do impacto das metodologias ágeis. Enquanto o debate em torno do uso dessas metodologias é frequente, realizar um estudo que analise métricas de desempenho de equipes operando sob modelos tradicionais de gestão e, posteriormente, após a adoção das metodologias ágeis discutidas neste trabalho, é crucial. Por exemplo, implementar a pontuação de tarefas, conforme mencionado na Seção 4.2, e comparar a quantidade de pontos entregues pela equipe antes e depois da adoção das metodologias, pode fornecer *insights* valiosos para uma análise comparativa em um contexto real.

Portanto, por meio deste relato de experiência, o autor mostrou como integrou teoria e prática, consolidando seu desenvolvimento no mercado ao aplicar os conhecimentos adquiridos na universidade. Além disso, ao documentar suas atividades, o autor fornece um guia para aqueles que almejam ingressar nessa área, oferecendo *insights* sobre o funcionamento do mercado e práticas comumente utilizadas. Dessa forma, espera-se contribuir para a comunidade educacional, facilitando a jornada de outros estudantes e profissionais em sua inserção e progressão no mercado de trabalho.

REFERÊNCIAS

CASTELLS, Manuel. A sociedade em rede. 24ª ed. Rio de Janeiro: Paz & Terra, 2013.

MAGELA, Rogério. Engenharia de Software Aplicada. Rio de Janeiro: Alta Books, 2006.

IEEE Computer Society. (2020). Guide to the Software Engineering Body of Knowledge (SWEBOK). 3ª edição. IEEE Computer Society Press.

SCHWABER, K; SUTHELAND, J. Scrum Guide: O Guia Definitivo para o Scrum: As Regras do Jogo, 2020.

PRIKLADNICKI, Rafael; WILL, Renato; MILANI, Fabiano. Métodos Ágeis para Desenvolvimento de Software, 2014.

VALENTE, Marco Engenharia de Software Moderna. 1ª ed. Brasil: Independente, 2022.

Forbes. Musk pede que funcionários da Tesla retornem ao escritório ou deixem a empresa. Forbes Brasil, 20 de junho de 2022. Disponível em: https://forbes.com.br/carreira/2022/06/musk-pede-que-funcionarios-da-tesla-retornem-ao-escritorio-ou-deixem-a-empresa/. Acesso em: 28 de dezembro de 2023.

SCRUM. The Scrum Framework Poster. Disponível em: https://www.scrum.org/resources/scrum-framework-poster. Acesso em: 25 de março de 2024.

Totvs. Kanban: conceito, como funciona, vantagens e implementação. Totvs, 03 de Novembro de 2023. Disponível em: https://www.totvs.com/blog/negocios/kanban/#:~:text=O%20termo%20%E2%80%9CKanban%E2%80%9D%20%C3%A9%20de,ele%20se%20move%20pelo%20processo. Acesso em: 18 de março de 2024.

Alura. Figma: o que é a ferramenta, Design e uso. Alura, 05 de Outubro de 2023. Disponível em: https://www.alura.com.br/artigos/figma#:~:text=O%20Figma%20%C3%A9%20uma%20plataforma,Dylan%20Field%20e%20Evan%20Wallace. Acesso em: 18 de março de 2024.

Lowdermilk, T. (2019). Design Centrado no Usuário: um guia para o desenvolvimento de aplicativos amigáveis. Novatec Editora.

Microsoft. O que é o Azure DevOps?. Microsoft, 05 de Janeiro de 2024. Disponível em: https://learn.microsoft.com/pt-br/azure/devops/user-guide/services?view=azure-devops. Acesso em: 22 de março de 2024.

Silvério, O. Relação de Story Points com o Tempo de Desenvolvimento (Lead Time). 30 set. 2019. Disponível em: https://blog.plataformatec.com.br/2019/09/relacao-de-story-points-com-o-tempo-de-desenvolvimento-lead-time/. Acesso em: 09 de maio de 2024.
Aguiar, Fábio; Caroli, Paulo. Product Backlog Building: Um guia prático para criação e refinamento de backlog para produtos de sucesso. 1ª edição. Editora Caroli, 2021.

Caroli, Paulo. FunRetrospectives: Atividades e ideias para tornar suas retrospectivas ágeis mais envolventes. 1ª ed. Editora Caroli, 2021.