UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUIZ CARLOS PEDROSO GOMES

SCRUM ROOT:
FERRAMENTA PARA INTEGRAR APF E SCRUM

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

LUIZ CARLOS PEDROSO GOMES

SCRUM ROOT

FERRAMENTA PARA INTEGRAR APF E SCRUM

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. MSc. Clayton Kossoski

PONTA GROSSA

2017

TERMO DE APROVAÇÃO

SCRUM ROOT:
FERRAMENTA PARA INTEGRAR APF E SCRUM
por

LUIZ CARLOS PEDROSO GOMES

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 31 de outubro de 2017 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Clayton Kossoski
Prof. Orientador

_____
Vinicius Camargo Andrade
Membro titular

_____
Rafael dos Passos Canteri
Membro titular

_____
Profª. Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra. Mauren Louise Sguario
Coordenadora do curso

AGRADECIMENTOS

Agradeço ao meu orientador, Prof. Msc. Clayton Kossoski, por todo o apoio prestado, toda a dedicação e confiança empregadas neste trabalho, além da ilustre orientação que colaborou para atingir os objetivos almejados.

Agradeço também a toda minha família e amigos que sempre estiveram ao meu lado me incentivando para que eu atingisse meus objetivos e me auxiliando nas escolhas certas.

A todos os envolvidos na realização deste trabalho e na minha vida acadêmica, os meus sinceros agradecimentos.

Por fim, agradeço a todas as pessoas que fizeram parte da minha vida nesta longa caminhada realizada na UTFPR, em especial aos professores que sempre nos ajudaram a obter o conhecimento necessário e a estar preparados para os desafios do mercado de trabalho.

RESUMO

PEDROSO GOMES, Luiz Carlos. **SCRUM ROOT: Ferramenta de integração de APF e SCRUM**. 2017. 85 f. Trabalho de Conclusão de Curso Tecnologia em Análise e Desenvolvimento de Sistemas – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

Este trabalho propõe uma técnica para uso de Análise de Pontos de Função (APF) em projetos que utilizam o *framework* Scrum. Para aplicação dessa técnica, foi desenvolvida uma ferramenta *open-source* que visa facilitar o lançamento de informações de controle. Essa técnica pode ser utilizada em qualquer projeto gerenciado por Scrum sem a perda de agilidade ou características por parte do *Scrum Team*. Para embasá-la foram consultados diversos trabalhos relevantes e também diversos profissionais da área. Essa técnica se mostrou efetiva para a construção da ferramenta e não alterou a dinâmica de desenvolvimento. Esse aplicativo pode ser utilizado para construção de outras aplicações, facilitando dessa forma a aplicação da técnica e verificação de histórico de estimativas.

Palavras-chave: Scrum. APF. Metodologias ágeis. Métricas. Estimativas.

ABSTRACT

PEDROSO GOMES, Luiz Carlos. SCRUM ROOT: APF and SCRUM integration tool.2017. 85 p. Work of Conclusion Course Systems Analysis and Development - Federal Technology University - Paraná. Ponta Grossa, 2017.

This paper proposes a technique to use Function Point Analysis (FPA) in projects that use the Scrum framework. For the application of this technique, an open-source tool was developed to facilitate the release of control information. This technique can be used on any Scrum-managed project without the loss of agility or features on the part of the Scrum Team. To support it, several relevant works were consulted as well as several professionals in the area. This technique proved effective for the construction of the tool and did not alter the dynamics of development. The tool can be used to build other applications, thus facilitating the application of the technique and verification of estimation history.

Keywords: Scrum. APF. Methodologie agile. Metrics. Estimates.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AIE	Arquivo de Interface Externa
ALI	Arquivo Lógico Interno
APF	Análise de Pontos de Função
API	Interface de Programação de Aplicativos
CE	Consulta Externa
CSS	Cascading Style Sheets (Folha de estilos em cascata)
EE	Entrada Externa
GPL	Licença Pública Geral
HTML	Linguagem de Marcação de Hipertexto
PF	Pontos de Função
SE	Saída Externa
SGDB	Sistema de Gerenciamento de Banco de Dados
SQL	Linguagem de Consulta Estruturada
UML	Linguagem de Modelagem Unificada
UX	User Experiencie

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software é uma atividade que demanda muito tempo e esforços na sua elaboração. Tendo isso como uma certeza, foram propostas metodologias[1] de desenvolvimento visando diminuir a complexidade do desenvolvimento e melhorar o processo em como um todo. Em geral, as metodologias de desenvolvimento de software são divididas em tradicionais e ágeis (SOMMERVILLE, 2011).

Uma metodologia de desenvolvimento tradicional se caracteriza pela baixa participação do cliente, pouca ou nenhuma flexibilidade sobre mudanças no projeto e documentação extensa. O modelo Cascata proposto na década de 1970 tem essas características, podendo ser empregado em casos quais os requisitos dos sistemas não sofrem alterações (SOMMERVILLE et al., 2008).

Até a década de 1990 a metodologia tradicional dominava o mercado de desenvolvimento de software. Entretanto, os produtos que foram desenvolvidos com estas metodologias não satisfaziam plenamente aos clientes. Alguns problemas eram comuns como: altos custos para modelagem e desenvolvimento, atrasos nas entregas e equipes desmotivadas (SUTHERLAND, 2016).

A dificuldade em desenvolver softwares está na constante mudança de requisitos, de orçamento, de tecnologias e regras de negócio (MANSUR, 2007). Por exemplo, atualmente as empresas de desenvolvimento possuem programadores distribuídos entre vários países e a concorrência é cada vez mais acirrada o que aumenta o desafio para desenvolvedores e empresas (COHN, 2011).

Em contraposição, às metodologias tradicionais de desenvolvimento de software, no ano de 2001, um grupo de entusiastas de tecnologias elaborou o Manifesto Ágil. Esse documento compilou as soluções que vinham sendo empregadas com sucesso na indústria. Esses especialistas sugeriram que um projeto de desenvolvimento deveria ter mais colaboração com o cliente, software em funcionamento, valorização de indivíduos e interações, respostas a mudanças e menos documentação (BEEDLE et al., 2015).

Para tentar amenizar essas dificuldades e aumentar a qualidade do software, muitas empresas passaram adotar metodologias ágeis de desenvolvimento. Uma

[1] Metodologia é um conjunto de regras para realizar um projeto (MICHAELIS, 2017).

metodologia ágil é caracterizada pela simplicidade, colaboração com o cliente e entrega constante de versões do produto. Alguns exemplos de metodologias ágeis são: Kanban, XP, ASD (Desenvolvimento de Software Adaptativo), DSDM (Método de Desenvolvimento de Sistemas Dinâmicos), Crystal, FDD (Desenvolvimento Dirigido à Funcionalidade), LSD (Desenvolvimento de Software Enxuto), AM (Modelo Ágil) e AUP (Processo Unificado Ágil) (PRESSMAN; MAXIM, 2016).

Atualmente no mercado desenvolvimento as metodologias ágeis ocupam um amplo espaço, pois corroboram com as tendências do redução de custos, maior eficiência e melhoramento constante de processos (COHN, 2011). A popularização da internet, dos *smartphones*, a evolução da capacidade de processamento e armazenamento gerou uma demanda ainda maior por programas (PRESSMAN; MAXIM, 2016). Dessa forma, o estudo e melhoramento dos processos de produção de software são essenciais para a geração de ferramentas capazes de suportar o aumento drástico da demanda por aplicativos.

Dentre as Metodologias Ágeis o Scrum tem ganhado cada vez mais adeptos, porque possui, na essência, os valores do Manifesto Ágil e a possibilidade de adaptar-se para pequenos, médios e grandes projetos. Esses projetos podem estar distribuídos globalmente e/ou suas equipes podem trabalhar separadamente (COHN, 2011).

O Scrum possui ferramentas que melhoram a qualidade do software a ser entregue ao cliente: *Product Backlog* é uma lista de requisitos, *Sprint* é o tempo que se leva para gerar uma versão do produto, *Product Owner* é o dono do produto – pode ser representado pelo cliente ou alguma pessoa que conheça o produto, o *Scrum Master* é um facilitador de tarefas e o *Scrum Team* é um conjunto de pessoas que desenvolvem o produto (SUTHERLAND; SCHWABER, 2013).

O *Scrum Team* é auto organizado e independente. Essas características são importantes para as pessoas que fazem parte do *Scrum Team* mantenham se comprometidas com aquilo que elas próprias definiram. Para que esse grupo de pessoas consiga atender aos prazos com qualidade são utilizadas técnicas para auxiliar na estimativa do projeto de desenvolvimento de software, como o *Planning Poker* (SUTHERLAND, 2016).

Entretanto, com a ineficiência dessa técnica os envolvidos no desenvolvimento de software têm procurado outras abordagens mais confiáveis e estabelecidas no mercado. A Análise de Pontos por Função (APF) é uma métrica

bastante robusta e é utilizada por diversas empresas. A APF mede o tamanho funcional de um software a partir dos requisitos que o projeto possui. Esses requisitos ou funcionalidade são obtidos sobre o ponto de vista do usuário (IFPUG, 2010).

Com o emprego cada vez maior de tecnologias ágeis algumas de suas vulnerabilidades se tornaram evidentes. Uma delas é a estimativa do tamanho software. Para resolver esse problema uma das estimativas que podem ser utilizadas é a APF, que pode ser empregada por equipes de desenvolvimento, *Product Owner* (dono do produto) e *stackholder* (parte interessada).

A APF já vem sendo utilizada por equipe ágeis, de várias formas e com resultados variados. Estudos indicam que para melhorar a precisão das estimativas em metodologias ágeis a APF se sobressai (IFPUG, 2017; AGUIAR; SYMONS; VLIET, 2017). Acerca desse assunto, para este trabalho, foram realizadas diversas pesquisas na literatura e até o momento não foi encontrada uma técnica descrita para aplicação da APF com SCRUM, deixando muitas dúvidas e espaço para implementações erradas de estimativas de pontos de função (AGUIAR; SYMONS; VLIET, 2017).

Tendo em vista essa necessidade, este trabalho busca propor uma técnica de uso de APF com SCRUM. Posteriormente, será desenvolvida uma ferramenta que utilize esta técnica para estimar a complexidade das tarefas, mantendo os pontos fortes e comuns dessas duas tecnologias, sem amenizar as características que tornam o Scrum e APF tão eficientes. Essa ferramenta será desenvolvida utilizando tecnologia de desenvolvimento *web* e aplicando os conceitos de Scrum e APF.

1.1 OBJETIVOS

O objetivo geral desse trabalho é propor uma técnica de utilização de APF em um projeto Scrum e o desenvolvimento de uma ferramenta *web* que será utilizada para integrar as funcionalidades de APF e Scrum.

Os objetivos específicos são:

- Propor uma técnica para utilização de APF em projeto Scrum
- Desenvolver uma ferramenta web para auxiliar na aplicação da técnica proposta
- Aplicar a técnica em um caso de estudo

- Estudar o Scrum e APF

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido nos seguintes capítulos. O Capítulo 2 exibe a Fundamentação Teórica que embasou este trabalho. O Capítulo 3 retrata a Metodologia que foi utilizada. O Capítulo 4 expõe a Técnica de integração entre APF e Scrum proposta. O Capítulo 5 demonstra o software utilizado para auxiliar na aplicação da técnica. O Capítulo 6 apresenta as Conclusões e Trabalhos Futuros que podem ser desenvolvidos a partir desse trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este Capítulo exibe a fundamentação teórica que embasou esse trabalho. A Seção 2.1 exibe as metodologias ágeis. A Seção 2.2 mostra a engenharia de requisitos. A Seção 2.3 apresenta o *framework*[2] Scrum. A Seção 2.4 apresenta análise de pontos de função. A seção 2.5 exibe trabalhos que sugerem uso de APF e Scrum. A Seção 2.6 apresenta caso de uso. A Seção 2.7 apresenta as tecnologias utilizadas para desenvolver a ferramenta.

2.1 METODOLOGIAS ÁGEIS

Nos últimos anos o desenvolvimento ágil tem tomado espaço e demonstrado que sua utilização gera resultados muito superiores ao desenvolvimento tradicional (TAVARES, 2015). Atualmente, muitas empresas contêm empregados distribuídos globalmente e precisam responder rapidamente a mudanças e com isso o uso de metodologias ágeis vem ganhando força nos últimos anos (NUNES, 2013).

Os projetos realizados com metodologias tradicionais exigem pouca ou nenhuma participação do cliente ao decorrer do projeto e é dada abrangência excessiva à documentação. Metodologias tradicionais como o RUP (Processo Unificado da Racional) praticamente não aceitam mudanças ao longo de sua elaboração, por isso têm baixa participação do cliente (MARTINS, 2007).

Diante disso, a indústria de software começou uma mudança para melhorar sua eficiência. Em 2001, um grupo de pesquisadores, desenvolvedores e entusiastas se reuniu para definir alguns pontos em uma nova forma de construir um software. O Manifesto Ágil foi escrito a partir desta reunião e documentou as melhores práticas e novas formas de desenvolvimento que já estavam sendo aplicadas no mercado de desensolvimento. Foram acordados os seguintes valores (BEEDLE et al., 2015):

- Indivíduos e interações são mais importantes que processos.
- Software em funcionamento é mais importante que documentação abrangente.

[2] *Framework* é um conjunto de funcionalidades que visam facilitar a adoção de uma ferramenta (BAZILIO, 2015).

- Colaboração com o cliente é mais importante que negociação de contratos.
- Resposta a mudanças são mais importantes que seguir um plano.

Os valores da metodologia ágil já vinham sendo aplicados antes do Manifesto Ágil. O Scrum e o XP foram desenvolvidos em meados da década de 1990 e têm gerado bons resultados ao longo dos anos, sendo bem sucedidas em projetos que haviam falhado com o uso de metodologias tradicionais (SUTHERLAND, 2016).

2.2 ENGENHARIA DE REQUISITOS

A Engenharia de Requisitos ocupa papel importante no desenvolvimento de software, ela é responsável pela definição de escopo, projeto, implementação e testes (MEDEIROS, 2014). Dessa forma, ocupa um amplo espaço em projetos tradicionais, em projetos ágeis desempenha papel fundamental na qualidade do software (ZARDETTO, 2016). As fases de levantamento de requisitos e projeto são as mais importantes no desenvolvimento de software e quando mal elaboradas são a causa da falha em quase metade dos projetos (VAZQUEZ; SIMÕES, 2016).

Em projetos tradicionais, as etapas de levantamento de requisitos e projeto do sistema geram uma extensa documentação e são essenciais para outras fases do desenvolvimento (MUZETTI, 2014).

O desenvolvimento de software seguindo as metodologias ágeis considera a importância da Engenharia de Requisitos, porém a extensa documentação e a necessidade de possuir todos os requisitos logo no início do projeto perde um pouco de espaço. As metodologias ágeis partem do pressuposto que o cliente não vai possuir todos os requisitos no início do desenvolvimento e novas funcionalidades podem surgir no decorrer do projeto (COHN, 2011; FONSECA, 2008).

No Scrum o principal responsável pelo levantamento e validação de requisitos é o *Product Owner*. Para isso é utilizado o *backlog* que considera a história do usuário ou outros dados obtidos (SUTHERLAND; SCHWABER, 2013).

Tanto em projetos ágeis como tradicionais, a figura dos *stakeholders,* ou parte interessadas é muito importante para o levantamento de requisitos. Uma vez que esses estão diretamente ligados ao resultado final do projeto (VAZQUEZ; SIMÕES, 2016). Existem diversas técnicas de levantamento de requisitos, tais como:

entrevistas, questionários, observação, análise do usuário sobre os protótipos (MEDEIROS, 2014).

Após o levantamento de requisitos inicia-se a modelagem. Nesta etapa pode ser utilizados diagramas de casos de uso (UML - Linguagem de Modelagem Unificada) ou histórias de usuário (PRESSMAN; MAXIM, 2016). Os requisitos funcionais descrevem as funcionalidade executadas por usuários no software e são utilizados em APF para definir a complexidade do software (IFPUG, 2010). Portanto, quando o levantamento de requisitos contém falhas, o projeto como um todo pode fracassar.

O bom uso da engenharia de software traz vantagens para o cliente e para o desenvolvedor: reduzindo custos, aumentando a qualidade do software, antecipando entrega e tornando o projeto escalável (VAZQUEZ; SIMÕES, 2016).

2.3 SCRUM

Scrum é um *framework* ágil, extremamente flexível, com abordagem moderna e objetiva. Tem se mostrado uma excelente alternativa para empresas que pretendem agregar eficiência em seu trabalho e qualidade a seu produto (SUTHERLAND, 2016). Agrega os valores do manifesto ágil tais como: valorização da equipe, colaboração com o cliente, respostas a mudanças e baixo índice de documentação (BEEDLE et al., 2015). O Scrum se destaca por aprofundar esses valores utilizando ferramentas como *Product Owner, Scrum Team, Product Backlog, Sprint, Scrum Master, Planning Poker* (SUTHERLAND; SCHWABER, 2013) e outras que vão de encontro com o manifesto ágil.

Product Backlog é uma lista de requisitos que o produto deve conter, é considerado parte viva do produto. O *Product Owner* é responsável por organizá-lo, mantendo os itens de maior prioridade no topo da lista. Em contato com o *Scrum Team*, o *Product Owner* pode e deve refinar esta lista (SUTHERLAND; SCHWABER, 2013).

O *Sprint* é o coração do Scrum, a cada 30 dias no máximo é entregue uma versão utilizável do produto. Esse ciclo de no máximo de 30 dias permite a equipe limitar o horizonte. Dessa forma, reduz os riscos, possibilidade de mudanças no que será construído e mudanças de complexidade (SUTHERLAND e SCHWABER, 2013). Segundo o seu cocriador, *Jeff Sutherland*, o *Sprint* foi idealizado a partir de um projeto

no MIT (Instituto de Tecnologia de Massachusetts) em que os alunos apresentavam as suas ideias e elas deveriam ser evoluídas a cada três 3 semanas; se o projeto não fosse bom ou não funcionasse era encerrado (SUTHERLAND, 2016).

Por ser uma parte tão importante da filosofia do SCRUM, o *Sprint,* é analisado diversas vezes pela equipe. Uma dessas análises é chamada de Reunião de Planejamento do *Sprint*. Para um *Sprint* de 30 dias ela deve durar no máximo 8 horas. Nesta reunião o *Scrum Master* certifica-se que os participantes entendam o objetivo e cumpram os prazos de duração da reunião. O *Product Owner* expõe o objetivo da *Sprint* e itens do *Backlog*. O *Scrum Team* define as tarefas que serão realizadas, seguindo a lista de prioridades, no decorrer do *Sprint* (SUTHERLAND e SCHWABER, 2013).

Na fase de planejamento da *Sprint* são utilizados técnicas para estimar o esforço que o *Scrum Team* realiza em cada tarefa, também é verificado se todos os membros da equipe possuem uma quantidade de carga de trabalho parecida (DIMES, 2014). Uma dessas técnicas é o *Planning Poker*, este utiliza um baralho de 13 cartas com a sequência de *Fibonacci*. Cada membro do *Scrum Team* lança uma carta de acordo com a complexidade da tarefa (RITTER, 2014).

Após o início da execução das tarefas pelo *Scrum Team*, o *Scrum Master* passa a dirigir uma reunião que dura no máximo 15 minutos e visa responder 3 perguntas: O que eu fiz ontem? O que farei hoje? Existe algum impedimento? Essa reunião é executada diariamente (SUTHERLAND; SCHWABER, 2013). Membros da equipe podem observar, mas somente o *Scrum Team* vai falar. Reuniões diárias melhoram a comunicação desse grupo, reduzem os riscos sobre o projeto e melhoram a tomada de decisão (JUNIOR, 2013).

Após a execução do *Sprint* é necessário revisá-lo. A Revisão da *Sprint* tem duração de 4 horas para um *Sprint* de 30 dias. Esta etapa visa apresentação do incremento e promove a colaboração das partes interessadas. O *Scrum Master* garante que o evento ocorra e que os participantes entendam seu objetivo. O *Product Owner* esclarece quais itens foram "Prontos" e quais ainda restam. Toda a equipe discute e colabora para gerar mais informações para o planejamento do *Sprint* e revisar o *Backlog* do produto (SUTHERLAND; SCHWABER, 2013). É sugerido que o *Scrum Team* apresente o incremento em funcionamento para o *Product Owner* e outros interessados (DIMES, 2014).

Para melhorar o processo que levou a mais uma versão do produto é feita uma reunião chamada de Retrospectiva do *Sprint*, que dura 3 horas para um *Sprint* de 30 dias (DIMES, 2014). Essa reunião visa melhorar os processos que o *Scrum Team* utilizou para gerar o incremento da Sprint. Essas melhorias devem ser aplicadas na próxima *Sprint*. O *Scrum Master* atua como incentivador para o *Scrum Team* encontrar e melhorar o processo e aumentar a qualidade do produto entregue ao final da *Sprint* (SUTHERLAND; SCHWABER, 2013).

Product Owner (PO) ou dono do produto, pode ser representado pelo cliente ou por uma pessoa interessada no projeto. O PO é responsável por viabilizar o projeto, definir a lista de funcionalidades, o *Backlog*; definir os critérios de aceite de cada funcionalidade e esclarecer dúvidas do *Scrum Team* quanto aos requisitos (PRIKLADNICKI; ORTH, 2009). O PO deve estar em contato com o cliente e com o *Scrum Team* a fim de garantir que maior valor possível seja entregue (CRUZ, 2015), uma vez que é o único responsável pelo *backlog* e pela sua lista de prioridades (SUTHERLAND; SCHWABER, 2013).

O *Scrum Master* é um líder servidor, responsável pela manutenção dos valores Scrum, relacionamento do projeto com os demais envolvidos. Quem assume este papel trabalha para a organização, para o *Scrum Team* e para o *Product Owner* (SUTHERLAND; SCHWABER, 2013). O *Scrum Master* guia a equipe (CRUZ, 2011) e serve o *Product Owner* para integrar, comunicar, ensinar, facilitar e compreender as práticas do Scrum. Faz a integração do *Scrum Team* com o *Backlog* do produto. Comunica a visão, objetivos e itens do *Backlog* do produto ao *Scrum Team*. Ensina o *Scrum Team* a criar itens do *backlog*. Compreende qual o objetivo do produto em longo prazo (SUTHERLAND; SCHWABER, 2013). Dessa forma serve como um direcionador e facilitador de tarefas ao *Product Owner* (SATURI, 2013).

Um *Scrum Team* é auto-organizado, multifuncional, responsável em transformar o *backlog* em incremento e tem um tamanho limitado (SUTHERLAND; SCHWABER, 2013). Atualmente se aceita a possibilidade de um *Scrum Team* estar fisicamente separado do restante da equipe ou mesmo a própria equipe estar distante (COHN, 2011).

Auto-organizado significa que não deve sofrer interferência externa e nem do *Scrum Master* (SUTHERLAND; SCHWABER, 2013). Dessa forma, é o *Scrum Team* quem define a tecnologia, implementações e padrões de software a fim de garantir a qualidade do software a ser entregue (CRUZ, 2015). A auto-organização não implica

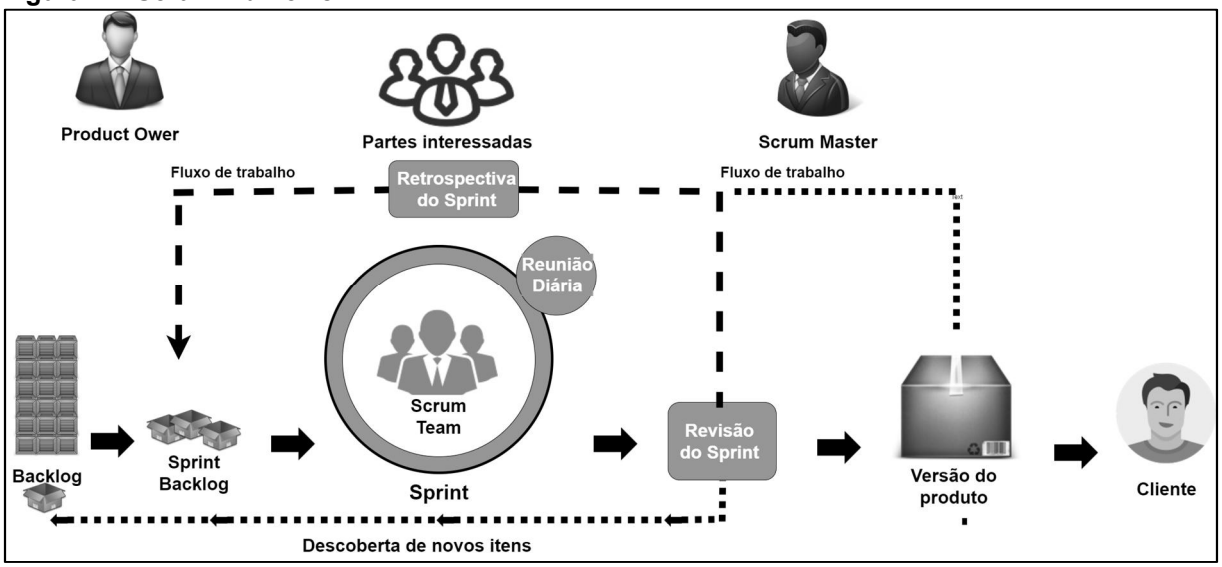
que a própria equipe irá decidir qual vai ser sua estrutura. A tarefa de como organizar a equipe deve ser tomada por um *Scrum Master* ou *stakeholders* (COHN, 2011).

Multifuncional significa que cada membro do *Scrum Team* pode realizar diversas tarefas e não existe no *Scrum Team* alguém especializado. Dessa forma a equipe sempre pode prosseguir quando um indivíduo for impedido (SATURI, 2013). A multifuncionalidade leva a equipe a conduzir o projeto buscando benefícios para todos os membros (SUTHERLAND, 2016).

Scrum Team menores, de 3 a 6 membros, tendem a ser mais produtivas. Quanto menor a equipe maior a interação entre os indivíduos. Uma equipe pequena é capaz de entregar mais projetos, com menos defeitos em relação a uma equipe maior e o seu tamanho reduzido facilita a comunicação entre seus membros (COHN, 2011).

A Figura 1 apresenta o Scrum e os principais processos presentes nessa metodologia.

Figura 1 – *Scrum framework*



Fonte: Adaptado de SCRUM (2017).

A Figura 1 apresenta o *Product Owner,* o dono do produto. As partes interessadas, grupo de pessoas ligadas ao projeto (setores da empresa, usuários afetados). O *Scrum Master* trabalha como um líder servidor e é o responsável pela manutenção dos valores Scrum na organização. O *backlog* que é a lista de requisitos do projeto, essa lista pode ser alterada durante o projeto. O *Sprint backlog* é a lista de itens que derivam do *backlog.* O *sprint,* que é um período de desenvolvimento, deve durar no máximo 30 dias. A reunião diária visa melhorar a comunicação da equipe. O

Scrum team é o grupo de desenvolvedores. A revisão do *Sprint* é utilizada para apresentar o produto ao PO. Ao final dessa etapa é entregue uma versão do produto ao cliente.

2.4 APF

A Análise de Pontos de Função (APF) tem sido amplamente utilizada, cerca de 60% dos profissionais envolvidos em algum tipo de projeto de desenvolvimento a utilizam. A APF mede o tamanho do software pela quantificação de suas funcionalidades, baseadas no projeto lógico ou a partir do modelo de dados segundo a visão e os requisitos do usuário final (IFPUG, 2010).

APF é utilizada em contrato de todas as modalidades possíveis. Isso vai desde de contratos homem/hora até licitações governamentais (VAZQUEZ; SIMÕES; ALBERT, 2008). Além disso, a APF possui independência da tecnologia utilizada pela empresa e pode ser empregada a qualquer momento no projeto (fase inicial, final ou manutenção).

Essa métrica não tem relação com a linguagem em que o software será construído. Ela determina o tamanho do software do ponto de vista do usuário. É utilizada nas fases inicias de projetos, para verificar e calcular a complexidade das tarefas. Em projetos em construção é aplicada para verificar se os itens construídos atendem o que foi pedido pelo cliente. Para projetos de manutenção é utilizada para verificar o esforço para realizar a manutenção do software (PINHEIRO, 2017).

Para empregar a APF em um projeto é necessário verificar qual será o tipo: projeto de desenvolvimento, aplicações instaladas ou projetos de manutenção. Após identificar qual o tipo de projeto, é necessário verificar e classificar os requisitos funcionais em dados e transações (PINHEIRO, 2017).

Para a estimativa de pontos de função em projeto é necessário verificar qual abordagem melhor se adéqua. Atualmente as instituições NESMA (Associação de Métricas de Software da Holanda) e IFPUG (Grupo Internacional de Usuários de Pontos de Função) adotam abordagens diferentes. A IFPUG adota a abordagem tradicional enquanto a NESMA adota conceitos mais práticos, porém todas as abordagens sugeridas por elas são autossuficientes (NESMA, 2017; IFPUG, 2017).

Quando um arquivo é classificado em dados ele é denominado ALI (Arquivos Lógicos Internos) ou AIE (Arquivos de Interface Externa). ALI é um dado persistido pela aplicação, por exemplo: as tabelas dos sistemas. Para contar um ALI devemos verificar quantas tabelas esse requisito afeta na base de dados. AIE são dados mantidos em outras aplicações (PINHEIRO, 2017). O Quadro 1 apresenta os níveis de complexidade ALI e AIE.

Quadro 1 – Complexidade de ALI e AIE

ALI AIE	TR TD	<20	20-50	>50
	1	Baixa	Baixa	Média
	2-5	Baixa	Média	Alta
	>5	Média	Alta	Alta

Fonte: Adaptado de IFPUG (2010)

Em estimativas de transações são analisados os dados fornecidos para o usuário. Existem 3 tipos de transações: Entrada Externa (EE), Saída Externa (SE) e Consultas Externas (CE) (PINHEIRO, 2017). É possível exemplificar as EE como as telas do sistema, pois a responsabilidade de uma EE é manter os dados ALI ou AIE (MACORATTI, 2010). Uma SE é responsável pelas informações obtidas através de cálculos e segundo o Manual IFPUG, (2010) resultam em alterações em dados ALI ou AIE, por exemplo: relatórios (MACORATTI, 2010). A CE ao contrário de uma SE não realiza cálculo para exibir seus dado, não causa alteração de nenhum ALI ou AIE (PINHEIRO, 2017). O Quadro 2 apresenta os níveis de complexidade para EE, SE e CE.

Quadro 2 – Complexidade EE, SE e CE.

EE	TR TD	<5	5-15	>15	SE CE	TR TD	<6	6-19	>19
	<2	Baixa	Baixa	Média		<2	Baixa	Baixa	Média
	2	Baixa	Média	Alta		2-3	Baixa	Média	Alta
	>2	Média	Alta	Alta		>3	Média	Alta	Alta

Fonte: Adaptado de IFPUG (2010)

Segundo a NESMA podem ser feitos três tipos de estimativas: Indicativa, Estimativa e Detalhada (NESMA, 2015). Uma estimativa indicativa é mais ágil, porém menos precisa; utiliza a seguinte formula: $PF = (35\ X\ número\ de\ ALIs) +$

$(15\ X\ números\ de\ AIEs)$. A contagem Estimada é muito parecida com a contagem tradicional, mas a complexidade é pré-definida como baixa para ALI e AIE e média para EE, SE e CE. A estimativa detalhada é idêntica a tradicional, nela é determina todos os tipos de funções, complexidade e é realizado o cálculo dos pontos de função (NESMA, 2015).

A complexidade de cada ALI e AIE, conforme o Quadro 1 é determinada pelo número de Tipo Dado e Tipo Registro. Tipo Dado é um campo único e não repetido. Tipo Registro é subconjunto de tipo dado dentro da ALI ou AIE. O Quadro 3 apresenta a contribuição de cada estimativa para a complexidade da tarefa.

Quadro 3 – Quadro de contribuição

	Tipo	Baixa	Média	Alta
Contribuição	ALI	7PF	10PF	15PF
	AIE	5PF	7PF	10PF
	EE	3PF	4PF	6PF
	SE	4PF	5PF	7PF
	CE	3PF	4PF	6PF

Fonte: Adaptado de IFPUG (2010)

O Fator de Ajuste é baseado em 14 características que afetam o sistema: Comunicação de Dados, Processamento Distribuído, Performance, Utilização do Equipamento, Volume de Transações, Entrada de Dados *On-line*, Eficiência do Usuário Final, Atualização *On-line*, Processamento Complexo, Reutilização de Código, Facilidade de Implantação, Facilidade Operacional, Múltiplos Locais e Facilidade de Mudanças (IFPUG, 2010). É dado um peso a cada fator, este peso vai de 0 (sem importância) a 5 (muito importante). O Fator de Ajuste é indicado como 1 em sistemas em que as essas características não causam efeito (LICHIRGU, 2016).

O processo de contagem segue os seguintes passos: definir o tipo de projeto, definir a fronteira da aplicação, estimar ALI e AIE, estimar EE, SE e CE, definir a complexidade e aplicar a equação de cálculo

$$(PF = \ (PF\ não\ ajustado + PF\ incluido\ + PF\ alterado\ atual - PF\ Alterado\ anterior - PF\ excluido\ )\ X\ Fator\ de\ ajuste\ atual\ ).$$

Dessa forma, o número de pontos de função para realizar o projeto e baseado em históricos de desenvolvimento anterior das equipes pode ser estimado um prazo (PINHEIRO, 2017).

Assim é possível perceber que a APF é muito flexível e pode ser aplicada em diversos tipos de projetos. Particularmente, no *framework* Scrum ela pode ser aplicada pelo *Scrum Team*, *Product Owne*r ou partes interessadas.

2.5 TRABALHOS QUE SUGEREM O USO DE APF EM SCRUM

A integração de APF e Scrum não é uma novidade. Profissionais e empresas sugerem que a integração deve ser realizada para melhorar o processo de desenvolvimento de software.

Segundo Onvlee (2015), Análise de Pontos de Função pode ser utilizada para encontrar o esforço total para desenvolvimento de um projeto Scrum. Ainda é possível usar as estimativas em APF para indicar melhorias no processo de desenvolvimento com Scrum. E verificar o esforço das organizações para adaptarem seus processos ao Scrum.

Segundo Aguiar, Symons e Vliet (2017), a APF responde questões referentes ao desempenho do *Scrum Team* e ao orçamento do projeto. Pode ser introduzida ao nível de *Sprints* ou iterações e auxiliam o gerenciamento de projetos na tomada de decisão. As principais instituições de estimativas em APF reconhecem a necessidade de manter as características do Scrum ao integrá-lo a essa estimativa.

Profissionais defendem o uso de APF em relação ao *Planning Poker*, uma vez que, a APF tende a ser mais precisa. Embora, a terminologia cause confusão em desenvolvedores sem experiência com essa estimativa (MATOS, 2010). Alguns profissionais exemplificam o uso desta estimativa em Scrum como Lichirgu (2016) porém, sem grande aprofundamento sobre uma técnica eficiente sobre a integração.

A recomendação de uso da APF em metodologias ágeis vem também do Governo Federal Brasileiro. O Ministério do Planejamento elaborou em 2015 o “Guia de contagem de pontos de função”. Esse guia orienta a aplicação da APF para as organizações que prestam serviços ao Governo Brasileiro. Ainda, esse guia aponta que é possível o uso dessas tecnologias em conjunto. Uma vez que o desenvolvimento ágil é a prática mais aplicada nas empresas e a APF foi adotada pelo governo como estimativa padrão (BOMFIM; ANDRADE, 2015).

Entretanto, nenhum dos trabalhos citados descreve uma forma de manter a eficiência dos processos existentes no *framework* Scrum ao integrá-lo com APF. Porém, todos admitem a importância de realizar a integração a fim de melhorar ainda mais o processo de desenvolvimento ágil.

2.6 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso é um contrato entre partes interessadas que visa demonstrar as características que um sistema deve possuir (PRESSMAN; MAXIM, 2016). É construído em forma escrita e/ou com uso de diagramas, as duas formas são extremamente simples e pessoas com conhecimento técnico básico sobre desenvolvimento de sistemas conseguem entender. A simplicidade visa gerar discussões que possam levar a elucidação de requisitos (COCKBURN, 2007).

Um diagrama caso de uso possui quatro partes: cenário, ator, caso de uso e comunicação. O cenário é todo o conjunto de funcionalidades que o caso de uso possui, o ator é um tipo de usuário, o caso de uso é uma funcionalidade do sistema e associação relaciona o caso de uso ao ator (RIBEIRO, 2012).

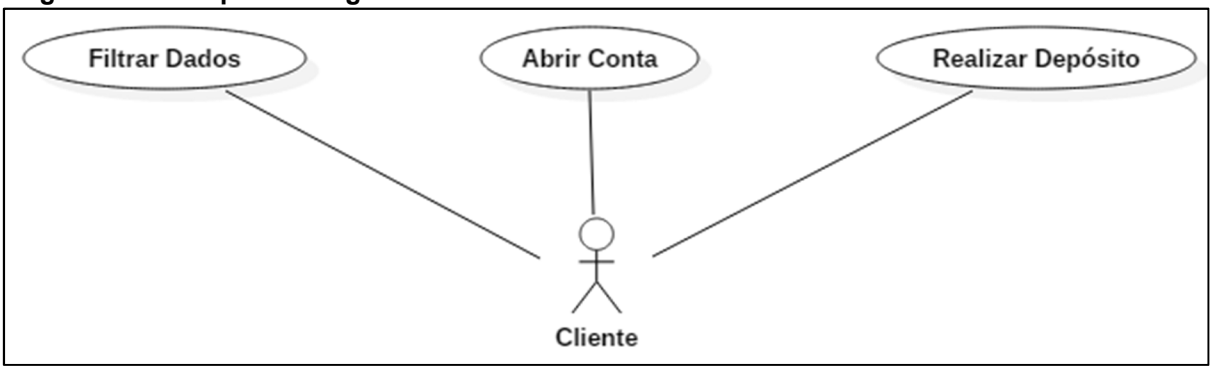
Casos de uso podem ser utilizados para documentação do software, planejamento de teste, para gerar novos casos de usos e validação dos atributos dos sistemas. O uso mais amplo dessas técnicas ocorre para levantamento de requisitos. Quando o caso de uso é utilizado para coleta de requisitos é levado em consideração o ponto de vista do usuário (WAZLAWICK, 2013).

Uma vez que um caso de uso é importante, mas não suficiente, podem ser utilizados outros diagramas UML (Linguagem Unificada de Modelagem) para complementa-lo.

A Figura 2 apresenta um exemplo de diagrama de caso de uso que contém um ator, casos de uso e a associação (setas que indicam o fluxo das iterações) (GUEDES, 2011).

Como um caso de uso visa expor características e ações de usuários no sistema, nesse trabalho será utilizado para exemplificar ações de atores dentro do aplicativo. Além disso pode ser utilizado para verificar requisitos do *backlog* e expor ao Scrum Team como a tarefa deve ser realizada para integrar o sistema.

Figura 2 – Exemplo de diagrama de caso de uso



Fonte: Adaptado de Guedes (2011)

O diagrama de exemplo descreve um cliente e suas opções no sistema: filtrar dados, abrir conta e realizar depósito.

2.7 APLICAÇÕES WEB

O aumento da velocidade da *Internet* e redução de custos de *hardware* sãos as causas desse crescimento. Com o aumento da velocidade da *Internet* sugiram novas aplicações para atender a demanda da sociedade (COMER, 2016). Entre essas novas aplicações, ferramentas *web* ocupam grande espaço devido as suas características (PRESSMAN; MAXIM, 2016).

O desenvolvimento de uma aplicação *web* requer cuidados como segurança, navegabilidade e usabilidade. Esses aspectos de uma aplicação *web* devem ser atendidos devido a sua abrangência. Que gera oportunidade para usuário fazer uso da aplicação por meio de diversos dispositivos (computador, TV, *Smartphone* etc.) o que torna aplicações *web* complexas (MILETTO; BERTAGNOLLI, 2014).

2.7.1 Computação em nuvem

Devido ao aumento da importância da *Internet* na sociedade sugiram novas concepções de uso dessa tecnologia. Entre essas novas concepções está a computação em nuvem, definida como a nova arquitetura de T.I. (Tecnologia da Informação). Essa arquitetura oferece serviços de computação através da internet possibilitando que o cliente tenha acesso a servidores, banco de dados, armazenamento e outros serviços (VERAS, 2013).

As características da computação em nuvem que a tornam tão popular são: redução de custos, escalabilidade, redução de esforço para provisionamento de recursos, cobrança conforme a demanda, distribuição dos recursos.

Instituições Governamentais e empresas que utilizam serviços de computação em nuvem para reduzir seus custos com T.I. (VERAS,2015). Uma das maiores empresas que oferecem serviços de computação em nuvem é a *Amazon*[3], que criou serviços para atender a sua demanda de varejo, e somente depois disponibilizou o serviço para o público (TAURION, 2009).

O aumento da velocidade da *Internet* tornou comum oferta de serviços em nuvem para facilitar algumas tarefas do dia a dia. Existem três serviços principais: Infraestrutura como Serviço (*IaaS*), Plataforma como Serviço (*PaaS*) e Software como Serviço (*SaaS*). O *PaaS* é voltado para o desenvolvimento de aplicativos e empresas como o *Google*, *Amazon, Microsoft e Heroku*[4] que oferecem ambiente de implantação e desenvolvimento na nuvem (MICROSOFT, 2017).

O *Heroku* é uma plataforma que oferece serviços *PaaS*, que tem suporte a uma ampla quantidade de linguagens de programação, servidores, base de dados e repositórios. Para linguagens de programação se tem suporte a: *PHP*, *Python*, *Java*, *Javascript, Ruby, Scala* e *Go.* Utiliza servidores *Apache* e *Nginx,* mas essa configuração pode ficar de forma transparente para o desenvolvedor. As bases de dados são: *Postgres, Mysql e MongoDB.* A publicação da aplicação é realizada por *script* após configuração de poucos arquivos. Para planos gratuitos o *Heroku* oferece de 550 a 1000 horas/mensais. É utilizado por *startups,* desenvolvedores e grandes empresas (HEROKU, 2017).

2.7.2 Banco de Dados

Banco de dados são ativos de empresas que contém informações importantes sobre determinada atividade. Estão presentes na maioria das aplicações na forma de arquivos ou sistemas (FEITOSA, 2008). São responsáveis por armazenar dados sensíveis que podem comprometer todo o sucesso de um empreendimento, não importando o porte deste (CARVALHO, 2017).

---

[3]Site: https://www.amazon.com.br/
[4]Site: https://www.heroku.com

Existem dois conceitos de Banco de Dados: relacional e não relacional. Banco de Dados relacionais possuem coleções de dados armazenados e inter-relacionados entre tabelas e essas tabelas possuem três campos básicos: colunas, linhas e campos. São indicados para casos que o sistema executa muitas atualizações na base de dados (NADEAU et al., 2013). Os Bancos de Dados não relacionais ou *NoSQL* (Não somente SQL) possuem coleções de dados orientados de diversas formas (documentos, chave-valor, grafos, colunas). São indicados para projetos em que a estrutura e o volume de dados é incerta, aplicações escaláveis e dados de diversas fontes (SADALAGE; FOWLER, 2013).

Os SGBD (Sistemas de Gerenciamento de Banco de Dados) são responsáveis por manipular, gerenciar, organizar e persistir os dados em uma base de dados. Entre os sistemas mais conhecidos de SGDBs estão: *Mysql, PostgresSQL, OracleDB, SQL-SERVER, MongoDB.* SGDBs possuem ferramentas que auxiliam desenvolvedores a melhorar a performance na leitura e escrita de dados em disco (CARNEIRO; MOREIRA; FREITAS, 2011). Auxiliam na manutenção da base de dados com tarefas automatizadas, como o *PostgresSQL.*

O *PostgresSQL* se destaca por sua confiabilidade, integridade de dados, robustez, facilidade de manutenção, *open-source*[5], tamanho da base de dados é ilimitado, multiplataforma (*Linux* e *Windows* e com uma comunidade extensa (POSTGRES, 2017). Suporta linguagens como *Python, Perl* e *TCL.* Essas linguagens podem ser usadas para escreverem *Store Procedures* que visam aumentar a performance ou gerar novos conjuntos de dados (CARVALHO, 2017).

2.7.3 Server-side

Server-side é o termo usado para definir linguagens que são executadas pelo servidor em que o servidor é responsável por processar os dados e enviar a resposta ao cliente (ESTROZI et al., 2010).

Entre as muitas linguagens que possuem a características de *server-side*, o PHP (*Hypertext Preprocessor*) que se destaca pela sua simplicidade e robustez. Atende desde profissionais iniciantes até profissionais experientes (PHP, 2017). As suas características são: estruturada ou orientada a objetos, interpretada,tipagem

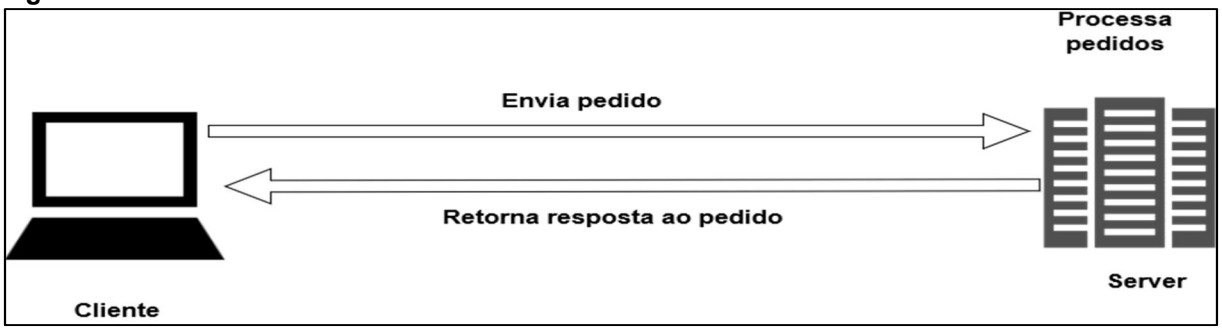
[5] *Open-source* um modelo de desenvolvimento para distribuição universal de software (CAMPOS, 2009).

dinâmica (durante a execução o tipo da variável pode ser alterado), não é *case-sensistive* e multiplataforma (diversos sistemas operacionais) (ESTROZI et al., 2010).

PHP está presente em quase 80% de todos os sites em que a linguagem de construção é conhecida e os três principais gerenciadores de conteúdo são escritos em PHP. É uma linguagem com ecossistema construído para facilitar o desenvolvimento. Fazem parte deste ecossistema o *Git:* repositório da linguagem e o *composer* gerenciador de dependências (POTENCIER, 2012). Em 2016 foi apontada como a sexta linguagem mais utilizada (LEITE, 2016).

A forte presença do PHP no mercado de desenvolvimento faz com que a linguagem seja origem de muitos *frameworks* que ajudam a reduzir as tarefas repetitivas e agilizar a construção ou manutenção de um projeto. *Zend Framework, CodeIgniter e Slim* são alguns *frameworks* de destaque no mercado de desenvolvimento (MINETTO, 2007). A Figura 3 – *Server Side*demostra o fluxo de requisições e respostas presentes neste tipo de tecnologia.

Figura 3 – *Server Side*



Fonte: Baseado de Martins (2015).

O *framework Slim* é uma ferramenta para desenvolvedores que buscam uma forma rápida de desenvolver soluções em PHP. Seu código é simples e pode ser interpretado facilmente (SCHMITZ, 2013). Pode ser utilizado para fazer um site completo (*back-end* e *front-end*) ou uma API (Interface de Programação de Aplicação). Atualmente está na versão 3.8 e possui suporte a ferramentas de resolução de dependências como o *composer* (SLIM, 2017).

2.7.4 Client-side

Client-side é o termo que distingui linguagens executadas no dispositivo do usuário em aplicações *web,* é responsável pela exibição de dados e capturas de ações

do usuário. Essas linguagens têm o objetivo de enriquecer a experiência do usuário com a aplicação (PRESCOTT, 2016).

Para o desenvolvimento do *client-side* de uma aplicação *web* são utilizadas basicamente três linguagens: HTML (Linguagem de Marcação de Hipertexto) que é responsável pela semântica e o conteúdo de página web, CSS (Folhas de Estilo em Cascata) que é responsável pela formatação e layout da página web, e Javascript, que é responsável por toda dinâmica e interatividade da página web e conexões assíncronas com o lado servidor (MILETTO; BERTAGNOLLI, 2014*).*

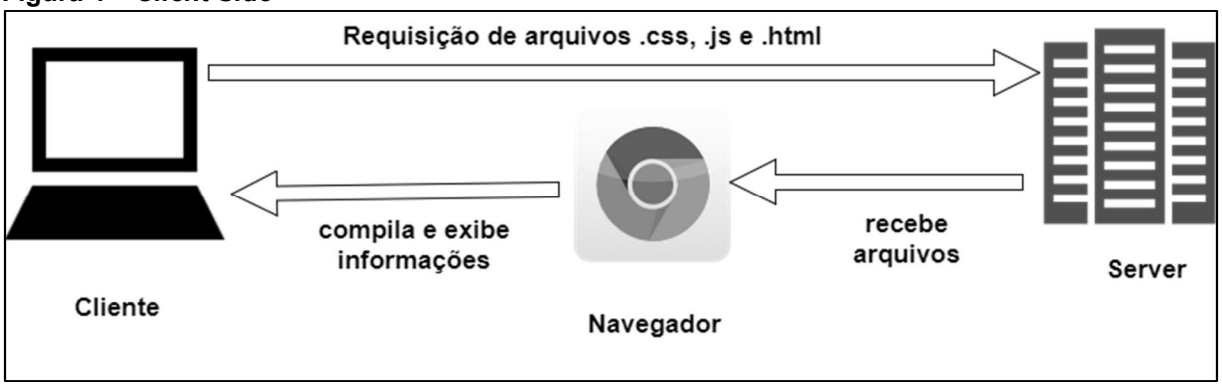
HTML é uma linguagem de marcação de conteúdos *web* de forma não linear. O HTML possui cerca de 108 elementos (t*ags*) para expor ao usuário um conteúdo ou capturar uma ação (SAMY, 2015). Atualmente está na versão 5, a qual faz do HTML uma linguagem mais versátil e voltada a apresentação de conteúdo, com mais recursos e totalmente compatível com os padrões das aplicações *web* (EIS; FERREIRA, 2012; OLIVEIRA, 2017)*.*

O CSS é uma linguagem de estilo utilizada para apresentar e formatar a apresentação de documentos HTML. Tem a função de descrever como os elementos são exibidos, auxiliando a usabilidade, acessibilidade e intuitividade da página. A versão atual é CSS3 e é padronizada pela W3C (CAVALCANTE, 2017). Com o aumento da diversidade de aplicações *web* tem se tornado cada vez mais complexo criar e manter o código em CSS (FRASSON, 2016).

O Javascript é a linguagem mais popular do lado cliente para programação de interatividade de uma página web. Nos últimos anos tem evoluído bastante e hoje em dia seu uso não se restringe apenas a navegadores, pode ser usada em ambientes sem navegadores ou ainda como server-side (GARZIA, 2017).

Para diminuir a complexidade e melhorar a produtividade do programador que faz uso de CSS e Javascript, surgiram diversos frameworks front-end como Bootstrap , Materialize para CSS e JQuery e Angular para Javascript. O Bootstrap é um framework que oferece diversos recursos para desenvolvimento com HTML, CSS e Javascript que visam tornar aplicações web responsivas, utiliza grids para definir o layout e manter a interface parecida para múltiplos dispositivos (ZEMEL, 2015) (ALBINO et al., 2015). O JQuery é bastante utilizado porque torna mais simples as tarefas de percorrer ou manipular documentos HTML e é suportado pela ampla maioria de navegadores existentes e atualmente (WOOD, 2013). A Figura 1 apresenta o fluxo de informação presente no *client-side.*

Figura 4 – *Client Side*



Fonte: Adaptado de Martins (2015).

Entretanto, a diversidade de aplicações, dispositivos e a evolução rápida destas tecnologias motivaram a construção de um novo conjunto de ferramentas. Grandes empresas desenvolveram ferramentas a fim de facilitar e simplificar o seu processo de manutenção e desenvolvimento, como o Google e o Facebook. Alguns exemplos são: Angular, Vue e React. O Angular é o*pen-source* foi desenvolvido pelo Google e tem a premissa de reduzir o número de linhas de código necessário para a construção de uma aplicação web. O React é mantido pela equipe de desenvolvimento do Facebook, utiliza a Virtual DOM[6] para manipular elementos do HTML e se popularizou a partir de 2015 quando se tornou open-source. Dessa maneira é mais eficiente para renderizações dos navegadores (FACEBOOK, 2017).

[6]Virtual dom é uma abstração dos elementos do HTML, que tornam mais eficiente a renderização em navegadores.

3 METODOLOGIA

Este Capítulo apresenta a metodologia de desenvolvimento deste trabalho. A Seção 3.1 exibe o escopo do trabalho. A Seção 3.2 mostra a análise de sistemas similares. A Seção 3.3 demonstra a metodologia de desenvolvimento deste trabalho.

3.1 ESCOPO DO TRABALHO

Este trabalho propõe e descreve uma técnica de estimativa com APF em projetos que utilizam Scrum e para isso faz uso de uma ferramenta que possibilita aplicação dessa técnica. A avaliação desta será realizada em trabalhos futuros contando com cenários mais amplos e diversificados.

Nesse trabalho, a estimativa em APF será realizada da forma tradicional, conforme descrito pela IFPUG (2010). A fim de simplificar a aplicação da técnica não serão estimados os pontos de função não ajustados.

A ferramenta que auxilia na aplicação da técnica será acessível por meio da internet para uso em computadores ou *smartphones*. Cada usuário cadastrado pode assumir somente um papel: *Scrum Master*, *Product Owner* ou membro do *Scrum Team*. Cada papel pode estabelecer apenas uma sessão no navegador. Será disponibilizada em um domínio de teste, sem restrições de acesso e sem cobrança de uso pelas suas funcionalidades.

Para uso da ferramenta é necessário conhecimento prévio dos papéis de Scrum e conhecimento sobre estimativas APF. Uma vez que não está no escopo de desenvolvimento da ferramenta esclarecer o usuário leigo como proceder ao utilizar a aplicação. Dessa forma, é mais indicado para equipes com no mínimo 3 membros e com conhecimento sobre Scrum e APF.

3.2 ANÁLISE DE SISTEMAS SIMILARES

O *Pipefy* se destaca no mercado por usar *templates* para múltiplas tarefas. Dentre estes, o *template* de desenvolvimento ágil, que foi analisado. Em sua configuração para desenvolvimento ágil a ferramenta não traz a possiblidade de estimativas de complexidade de tarefa ou a divisão clara entre as funcionalidades de

uma equipe Scrum. O que torna a ferramenta excessivamente genérica para gerenciamento de projetos Scrum.

O *Taga.io* é uma ferramenta desenvolvida com *Python* e *Angular*[7]*,* possui código fonte aberto e com foco de estimativas para UX[8] e histórias de usuário. Nesta ferramenta, o usuário pode escolher o tipo de projeto a ser criado, cadastrar projeto, cadastrar tarefa, adicionar notas sobre a tarefa, fazer estimativa de UX sobre a tarefa e outras funcionalidades. Esta ferramenta é mais completa que Pipefy, porém as estimativas e cada papel do Scrum podem ser realizadas por todos usuários, ou seja, não há restrição no tocante ao tipo de papel que se está utilizando, possibilitando a qualquer usuário realizar a atividade de outros papéis, confundindo o escopo de trabalho de cada ator no Scrum.

A ferramenta *Projects* Scrum do *ERP ODOO* tem características como definição de *Scrum Team,* criação de *Sprint,* criação de tarefas e outras. Entretanto essa ferramenta não faz estimativas e possui apenas dois níveis de acesso que são gerente e usuário*,* no qual o gerente tem acesso a todas as ações e o usuário apenas lista o que é cadastrado.

3.3 METODOLOGIA DE DESENVOLVIMENTO

Para a elaboração deste trabalho foram realizadas diversas etapas que podem ser agrupadas em: (a) fundamentação teórica, (b) entrevista com profissionais ligados à área de desenvolvimento, (c) proposta de uma técnica para utilizar estimativas em APF em projetos SCRUM, (d) desenvolvimento de uma ferramenta, utilizando está técnica, que possibilite integração de APF e SCRUM.

(a) Fundamentação teórica

A fim de realizar a fundamentação teórica deste trabalho foi realizada uma pesquisa em livros, artigos e repositórios especializados. Essa pesquisa se estendeu sobre os seguintes assuntos: engenharia de requisitos, metodologias ágeis, Scrum, APF e diagrama de caso de uso. Também foram pesquisados sites muito utilizados por desenvolvedores e usuários de Scrum ou APF.

(b) Entrevista com profissionais ligados a área de desenvolvimento

---

[7] Angular é um *framework* para *front-end* desenvolvido pelo Google (WALTENBERG, 2016).
[8] UX engloba estratégia de design e desenvolvimento de um produto (LOVI, 2017).

A entrevista foi realizada com profissionais atuantes nas áreas de desenvolvimento, gerência e suporte que desenvolvem seus trabalhos na cidade de Ponta Grossa/Paraná e São Paulo/SP. Cada entrevistado respondeu a um questionário sobre estimativas de software como tema central. Também foi verificada através do questionário a aceitabilidade da integração entre metodologias ágeis e APF. Além disso, foi levantado a experiência e o cargo que cada entrevistado possui. O questionário e as respostas obtidas estão disponíveis no Apêndice A.

(c) Proposta de uma técnica para utilizar estimativas em APF em projetos Scrum

A proposta de técnica para integrar APF e Scrum foi embasada em estudos que indicavam que há necessidade de aplicar uma técnica para integração de APF e Scrum, já citados no Capítulo da Fundamentação Teórica, que, entretanto, não apresentavam qualquer descrição de técnica ou método de aplicação. Nesse trabalho, após a elaboração da proposta foi elaborado um passo a passo que orienta como fazer uso da técnica proposta. O passo a passo da aplicação da técnica está disponível no Capítulo 4.

(d) Desenvolvimento de uma ferramenta que utiliza a técnica proposta para a integração de APF e Scrum.

O desenvolvimento da ferramenta foi realizado utilizando a técnica para estimar a complexidade de cada tarefa e/ou *Sprint.* Antes de iniciar cada *Sprint* foram analisados os itens de maior prioridade no *backlog.* Após a seleção de cada item tratou-se de criar um diagrama de caso de uso para relacionar as funcionalidades que seriam desenvolvidas. Também foi gerado um protótipo de baixa fidelidade para facilitar a estimativa e descoberta de eventos. Após analisar, tanto o diagrama de caso de uso como o protótipo, foi gerada uma lista de tarefas detalhadas para o *Sprint.* Por fim, foi realizada a estimativa em APF e a estimativa tradicional sem fatores de ajustes conforme apresentado pela IFPUG (2010).

Após cada *Sprint*, foram apresentados os resultados, realizados testes para verificar o comportamento da ferramenta e logo após o *deploy* da aplicação em servidores de produção no Heroku[9].

[9] Link principal do site Heroku: <https://www.heroku.com>

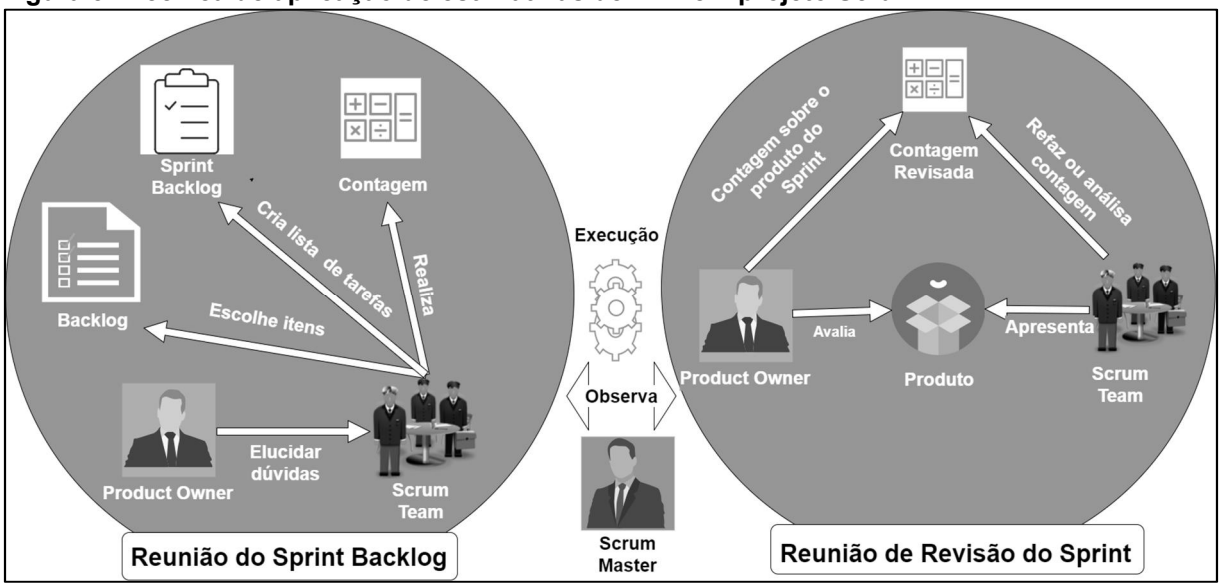
4 PROPOSTA DE UMA TÉCNICA PARA USO DE SCRUM COM ESTIMATIVAS APF

Este Capítulo propõe uma técnica para utilizar a APF em um projeto SCRUM. Para isso, foi considerada a pesquisa ampla sobre APF e Scrum em livros e repositórios de trabalhos científicos. A Seção 4.1 apresenta a descrição da técnica. A Seção 4.2 exibe o passo a passo para a aplicação e uso da técnica. Concluindo este Capítulo, a Seção 4.3 demonstra considerações sobre o uso da técnica.

4.1 DESCRIÇÃO DA TÉCNICA

A técnica de utilização de APF em um projeto Scrum aplica-se nas fases de desenvolvimento, manutenção e validação de software sob esse *framework* ágil. Basicamente, esse procedimento envolve todos os atores do Scrum e as estimativas da APF. A Figura 5 apresenta um modelo esquemático do método proposta. A fundamentação teórica apresentou a importância da independência do *Scrum Team* para realizar uma estimativa confiável, bem como a distinção de papéis na aplicação do Scrum para maior eficiência das estimativas. Portanto, essa técnica aplica essas sugestões na sua essência e acrescenta o uso de uma ferramenta para auxiliar no uso.

Figura 5 –Técnica de aplicação de estimativas de APF em projeto Scrum



Fonte: Autoria própria

Essa técnica se aplica a reunião do Sprint Backlog que envolve: *Product Owner*, *Scrum Master* e *Scrum Team*. O *Scrum Master*, o responsável pela manutenção dos valores do Scrum na organização, reúne o *Scrum Team* e o *Product Owner* para a reunião do *Sprint Backlog*. Nesta reunião o *Scrum Team* define quais itens serão inseridos no *Sprint.* Após a definição dos itens que serão incluídos, o *Scrum Team* passa a definir as tarefas que cada item vai possuir, neste momento caso exista dúvidas o *Product Owner* pode elucidar as dúvidas do *Scrum Team*.

Para realizar a estimativa é necessário que cada membro do *Scrum Team* tenha conhecimento básico sobre APF, ou seja, cada membro deve saber identificar os conjuntos de dados (ALI ou AIE), conjunto de transações (EE, CE e SE), tipo dado e tipo registro. Dessa forma, o *Scrum Team* tem uma maior independência e assertividade na aplicação da técnica. Após a conclusão de cada *Sprint* a estimativa pode ser revisada pelo próprio *Scrum Team* ou partes interessadas, sempre mantendo o estilo de estimativa usado, na reunião do *Sprint Backlog*.

Com isso, esta técnica visa empoderar o *Scrum Team* com as vantagens do uso de estimativas de software e seus respectivos prazos. Possibilita também que as partes interessadas revisem ou façam sua própria estimativa, bem como comparem o desempenho das equipes em execução de um projeto.

Além da revisão bibliográfica sobre Scrum e APF, foram levadas em consideração as informações obtidas no questionário aplicado a profissionais da área, e a consulta em sites especializados sobre os dois temas.

A pesquisa com profissionais da área de desenvolvimento de software mostrou a necessidade de uma descrição de um passo a passo para que a técnica de estimativa com APF seja empregada de modo efetivo e correto em projetos Scrum. Após a análise das respostas do questionário foi observado que vários participantes que utilizam o Scrum não aplicam corretamente a metodologia. Por exemplo, indicam que o *Scrum Team* não é o responsável pela estimativa, o que indica uma falha na aplicação do próprio Scrum. Outros entrevistados indicam a insatisfação com os prazos na entrega e a intenção de usar uma nova forma de estimar suas tarefas. Assim, o resultado desse questionário constatou o uso incorreto do próprio Scrum, mesmo pelas equipes de desenvolvimento mais experientes.

A aplicação dessa técnica é exemplificada no Capítulo 5, que apresenta o desenvolvimento de um software que faz uso da metodologia Scrum, APF e facilita a

aplicação da técnica. Foram utilizados os artefatos e papéis presentes no Scrum e o processo já conhecido de estimativa de pontos de função para criar essa técnica.

4.2 PASSO A PASSO DA APLICAÇÃO DA TÉCNICA

Para aplicar APF sobre um projeto que utilize SCRUM devem ser seguidos os seguintes passos:

- Reunir todo o *Scrum Team* e o *Product Owner* (reunião *Sprint Backlog*);
- O *Scrum Team* define o *Sprint Backlog* baseado no *Backlog* do produto;
- O *Scrum Team* define qual o estilo de estimativa em APF;
- O *Scrum Team* discute as características de estimativa da tarefa como os números de ALI, AIE, EE, CE, SE
- *Scrum Team* soma os pontos para cada tarefa;
- Baseado na quantidade de pontos de função, o *Scrum Team* realiza uma estimativa sobre o prazo de entrega;
- Na reunião de apresentação do resultado do *Sprint* as estimativas podem ser revisadas e gerado o histórico de estimativas;
- Na reunião de revisão do *Sprint* deve se discutir quais foram as dificuldades em aplicar a APF;

4.3 ORIENTAÇÕES PARA APLICAÇÃO DA TÉCNICA

Vale ressaltar que qualquer estimativa está apoiada no histórico que cada empresa possui. Aplicar uma estimativa e estipular um prazo pode não ser eficiente se não for considerado um histórico de características e necessidades para um dado tipo de projeto. O histórico de um projeto deve conter quais tecnologias foram utilizadas, qual o propósito do software e outras informações relevantes para a empresa. Dessa forma, a equipe terá uma maior possibilidade de sucesso quanto ao cumprimento do prazo.

Para exemplificar a importância do histórico é possível utilizar como exemplo o desenvolvimento de um aplicativo *mobile* e outro com praticamente as mesmas funcionalidades para *desktop.* Uma estimativa usando APF traria resultados idênticos.

Porém a complexidade, a especialidade do *Scrum Team,* o *deploy* e a manutenção seriam diferentes, assim como os custos e o tempo de desenvolvimento.

Essa técnica apresenta diversas vantagens, como a possiblidade de revisão ou comparação das estimativas. É importante que o mesmo estilo de estimativa seja aplicado pela equipe e partes interessadas. Também, é necessário discutir as diferenças das estimativas entre as que são realizadas por papeis diferentes.

Empresas podem recorrer a esta técnica para suprir demandas governamentais e terceirização das etapas de construção de um software sem precisar alterar sua metodologia. O uso de estimativa por APF permite a realização mais eficiente de auditorias sobre os projetos, o que é bastante necessário para o desenvolvimento de software para entidades governamentais e/ou com muitas partes envolvidas. Isso melhora a realização de testes por terceiros e permite a análise de forma independente.

5 APLICAÇÃO DA TÉCNICA PROPOSTA

Este Capítulo aplica a técnica de uso de APF em um projeto que segue a metodologia Scrum na elaboração de uma ferramenta (site). Essa ferramenta auxilia a aplicabilidade da própria técnica. Assim, são apresentados os *Sprints* de desenvolvimento da ferramenta e a estimativa de cada Sprint. A Seção 5.1 apresenta o *backlog* inicial da ferramenta. A Seção 5.2 até a Seção 5.7 exibe o desenvolvimento e o resultado dos *Sprints*. Por fim a Seção 5.8 demonstra a comparação da técnica com outras formas de estimativa em Scrum.

5.1 REQUISITOS (*BACKLOG)*

O desenvolvimento da ferramenta começa no uso do *framework* Scrum e passa pela construção do *Backlog* do produto. Dessa forma, os seguintes itens são apresentados no Quadro 4:

Quadro 4 – *Product Backlog* Inicial

***Product Backlog* - inicial**	
Prioridade	**Nome**
1	Cadastro de usuários
2	*Login* de usuários
3	*Product Owner* cria/edita/lista/exclui projeto
4	*Product Owner* cria/edita/lista/exclui item ao backlog
5	*Product Owner* associa Scrum Master ao projeto
6	Scrum Master define *Scrum Team* para o projeto
7	*Scrum Team* lista *backlog* do *sprint*
8	Formulário para o *Scrum Team* gerar APF para cada tarefa

Fonte: Autoria própria

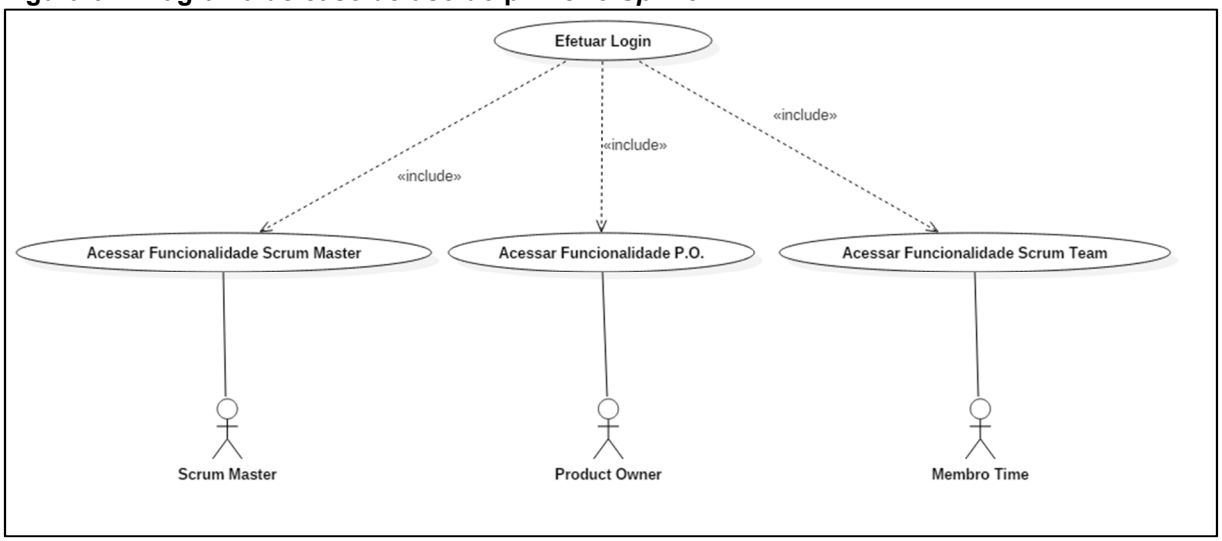
Os requisitos levantados até essa etapa do trabalho são básicos para o funcionamento da ferramenta, novos requisitos vão surgir e a tendência é que essa lista aumente ou mude ao longo do desenvolvimento. Essas mudanças são possíveis dentro do contexto do *framework Scrum*.

5.2 PRIMEIRO *SPRINT*

A partir do primeiro Sprint é possível aplicar a técnica de estimativa de pontos de função e realizar a construção do *Sprint Backlog* para o primeiro *Sprint.* Para esta etapa foram selecionados os itens de maior prioridade no *backlog* do projeto: “Cadastro Usuários” e “*Login* Usuários”.

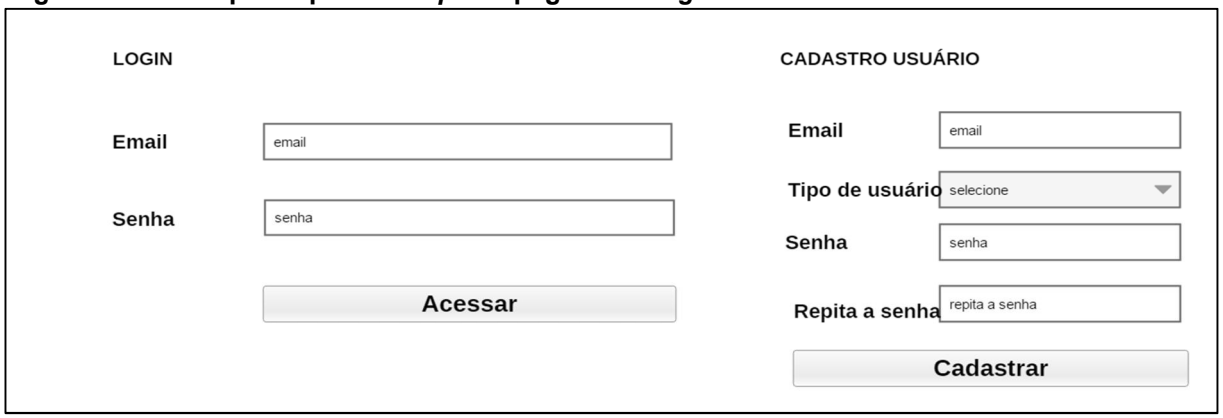
A Figura 6 apresenta um diagrama de caso de uso que pode ser utilizado para auxiliar na compreensão do sistema. O diagrama de caso de uso é utilizado também para auxiliar no levantamento de requisitos e para a geração de um protótipo de baixa fidelidade.

Figura 6 – Diagrama de caso de uso do primeiro *Sprint*



Fonte: Autoria própria

Figura 7 – Protótipo do primeiro *sprint*: página de *Login*/Cadastro



Fonte: Autoria Própria

A Figura 7 representa o protótipo de baixa fidelidade para o primeiro Sprint. Este protótipo sugere quais campos a página deve conter e como devem ficar distribuídos no layout.

Após a análise dessas figuras, é possível definir de maneira mais assertiva o conteúdo das tarefas do *Sprint Backlog* para o primeiro *Sprint.* Dessa forma as tarefas que devem ser concluídas ao final dessa etapa são representadas no Quadro 5.

Quadro 5 – *Sprint Backlog* do primeiro *Sprint*

Item backlog	Tarefas
Cadastro Usuários	Criar formulário de cadastro (*e-mail*, tipo, senha, repetir senha)
	Criar tabela usuário (*e-mail*, tipo, senha, data criação)
	Validação de campos e inserção de dados no banco de dados
***Login* Usuários**	Criação de formulário de *login* (*e-mail*, senha)
	Validação e redirecionamento conforme tipo do usuário

Fonte: Autoria própria

A estimativa foi realizada utilizando como base na Figura 6, Figura 7 e o Quadro 5. O Quadro 6 apresenta o levantamento de estimativas do primeiro Sprint:

Quadro 6 – Estimativa primeiro *Sprint*

Tipo de Função	Complexidade Funcional	Total do Tipo de função
ALI	01(Baixa) X 7	07
EE	01(Baixa) X 3	03
CE	01(Media) X 4	04
TOTAL		**14**

Fonte: Autoria própria

5.2.1 Resultado

O resultado do primeiro *Sprint* define a configuração do ambiente de desenvolvimento, a construção do *back-end* e *front-end e* o primeiro *upload* do projeto para o servidor do Heroku. O formulário de cadastro de usuário e o formulário de *login* ficaram definidos em uma única página, o formulário de cadastro que é ilustrado na Figura 8.

O formulário de cadastro (Figura 8 (B)) possui os seguintes campos: e-mail, função no time, senha e repita senha. Após o clique do usuário no botão cadastrar os dados são enviados ao servidor. O formulário de *login* (Figura 8 (A)) possui os campos:

e-mail e senha. Quando o *login* for realizado, o servidor redireciona o usuário para a página respectiva da função da equipe.

Figura 8 – Resultado primeiro *Sprint: página de Login/Cadastro*

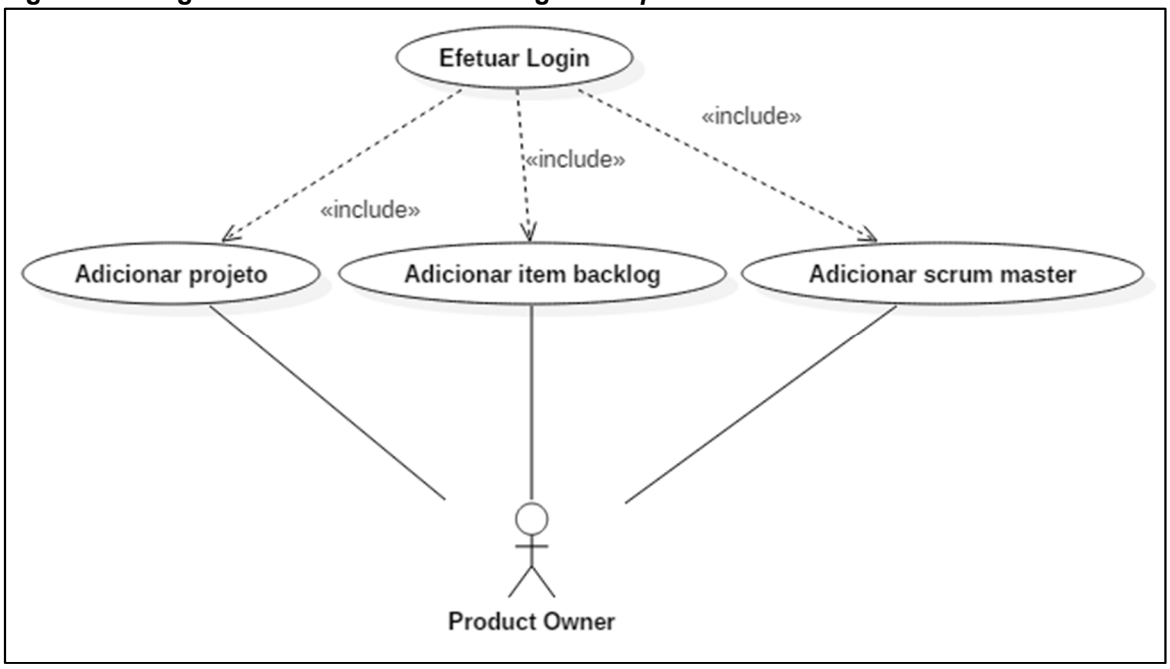
Fonte: Autoria própria

Essa tarefa foi estimada em 14 PF e foi realizada em 15 horas. Entretanto essa proporção de PF/horas deve reduzir nas próximas *Sprints* uma vez que a configuração de ambiente de desenvolvimento foi realizada. Como o resultado foi similar ao que se esperava na definição do *Sprint Backlog,* não foi realizado a revisão da estimativa.

5.3 SEGUNDO *SPRINT*

Nesta etapa é abordado o desenvolvimento das funcionalidades para o *Product Owner* como cadastro, edição e arquivamento de projetos e a adição de um *Scrum Master* ao projeto. Foram selecionados os itens do *backlog*: "*Product Owner* criar/editar/listar/excluir projeto", "*Product Owner* criar/editar/lista/excluir item ao backlog" e *Produtc Owner* associar Scrum Master ao projeto. A Figura 9 apresenta o diagrama de caso de uso do segundo Sprint.

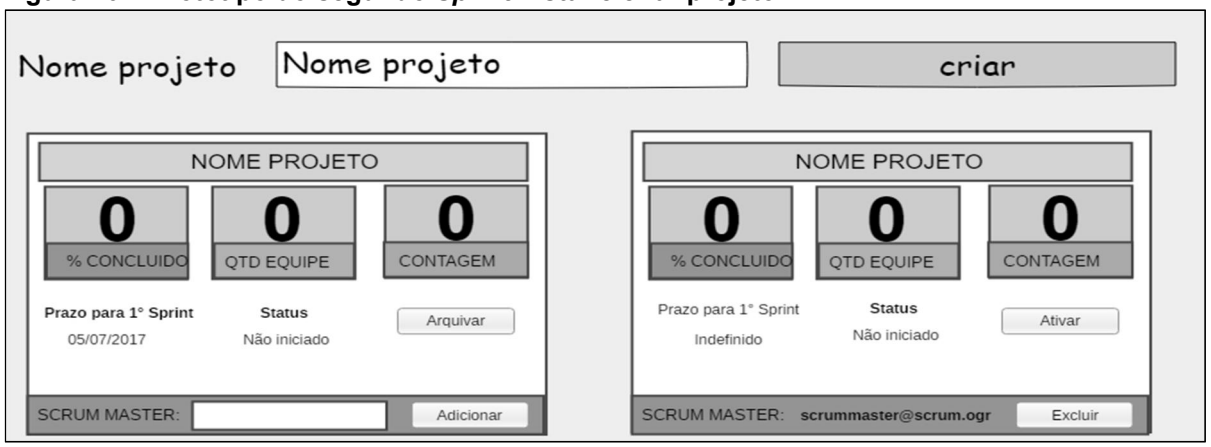
Figura 9 – Diagrama de caso de uso do segundo *Sprint*



Fonte: Autoria Própria

A Figura 9 representa algumas das funcionalidades do *Product Owner* pode acessar na ferramenta, como adicionar projeto, adicionar item backlog e adicionar *Scrum Master.*

Figura 10 – Protótipo do segundo *Sprint*: listar e criar projeto



Fonte: Autoria Própria

Com o diagrama de caso de uso e as informações sobre o projeto é possível gerar um protótipo de baixa fidelidade da tela de projeto, apresentado na Figura 10, e da tela de cadastro dos itens do Backlog na Figura 11.

Figura 11 – Protótipo do segundo *Sprint*: listar e adicionar itens ao *Backlog* do projeto

Fonte: Autoria Própria

Com estes dois protótipos e o diagrama de caso de uso é possível construir o *Sprint Backlog* para a segunda etapa, que ficou conforme o Quadro 7:

Quadro 7 – *Sprint Backlog* do segundo Sprint

Item backlog	Tarefas
Product Owner **criar/listar projeto**	Criar a página de cadastro e lista de projeto (nome)
	Criar a tabela projeto (nome, data entrega, data criação, usuário)
	Validar os campos e inserção de dados no banco (*back-end*)
Product Owner **criar/editar/listar /arquivar item backlog**	Criar página de cadastro e lista de *backlog* (nome, prioridade, descrição)
	Criar tabela tarefa (nome, data criação, prioridade, descrição, projeto)
	Validar os campos e a inserção de dados no banco de dados
Produtc Owner **adicionar *Scrum Master* ao projeto**	Criar campo para pesquisa e um botão para adicionar *Scrum Master* ao projeto.

Fonte: Autoria Própria

Analisando a Figura 9, Figura 10,Figura 11 e o Quadro 7 é possível fazer a estimativa em pontos de função para o segundo *Sprint*, conforme apresenta o Quadro 8:

Quadro 8 – Estimativa segundo Sprint

Tipo de Função	Complexidade Funcional	Total do Tipo de função
ALI	02(Baixa) X 7	14
EE	03(Baixa) X 3	09
CE	03(Baixa) X 3	09
TOTAL		**32**

Fonte: Autoria própria

5.3.1 Resultado

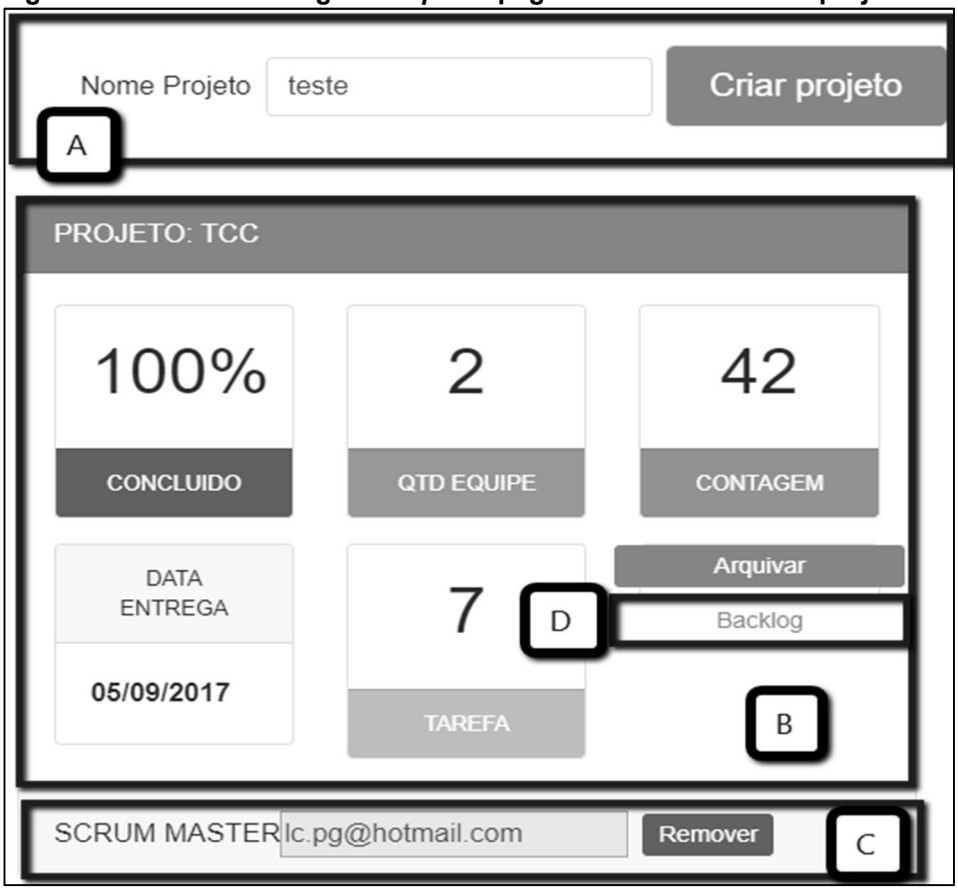
A página inicial do usuário *Product Owner* mostra *card*[10] com projetos cadastros pelo usuário (Figura 12 (B)) com os dados referentes ao projeto (porcentagem concluído, número de membros na equipe, estimativas de pontos de função, tarefa, *link* para arquivar, *link* para o *Backlog* e data de entrega). Desses dados apenas o *link* do *backlog* foi implementado, o restante foi construído aguardando implementação futura.

Ainda na página inicial do P.O. temos o campo para inserir o nome e o botão para enviar o formulário para o servidor (Figura 12 (A)). O formulário fica no alto da página e realiza a validação do campo nome do projeto antes de submeter os dados para o servidor.

No rodapé do *card* de cada projeto, foi construído um formulário com o campo de e-mail e o botão para adicionar um Scrum Master ao projeto, caso não exista. Quando há um Scrum Master vinculado, seu *e-mail* e a opção de remove-lo, são exibidos. Dessa forma, o P.O. pode adicionar ou remover *Scrum Master* ao projeto (Figura 12(C)).

[10]Contêiner de conteúdo flexível e extensível. Inclui opções de rodapé, cabeçalho, cores de fundo e outras opções de exibição (ZEMEL, 2015).

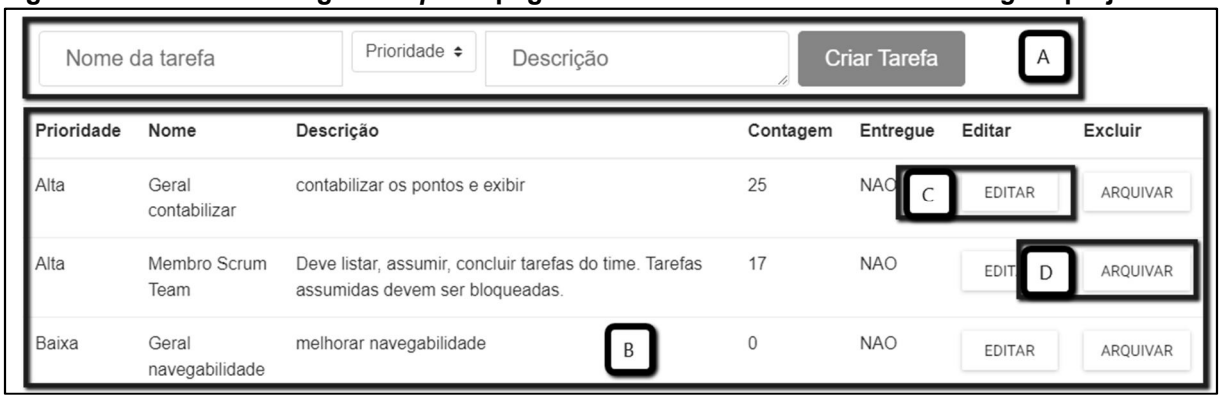
Figura 12 – Resultado segundo S*print*: página de cadastro/listar projetos



Fonte: Autoria própria

Ao clicar no *link* do *Backlog* (Figura 12 (D)), o usuário é direcionado para a página de cadastro/lista de itens do *Backlog* do projeto (Figura 13). No topo dessa página foi construído um formulário com os campos nome da tarefa, prioridade e descrição. O botão criar tarefa envia os dados validados para o servidor (Figura 13 (A)). Os dados validados pelo servidor e inseridos no banco são exibidos na tabela abaixo do formulário (Figura 13 (B)). O P.O. pode editar (Figura 13 (C)) ou arquivar (Figura 13 (D)) o item do *backlog.*

Figura 13 – Resultado segundo *sprint*: página cadastro/lista de itens do *backlog* do projeto



Fonte: Autoria própria

Ao final deste *Sprint* é possível usar as funcionalidades criadas para o *Product Owner* para cadastrar um projeto com nome de “TCC” e criar seu *backlog* para ser utilizado nos próximos *Sprints.* A lista de tarefas cadastradas se limitou a itens no *backlog* que não estavam no início do projeto. E adicionar e arquivar tarefas posteriormente ao longo do projeto.

Os 32 PF estimados no início do segundo S*print* foram realizados em 30 horas. Novamente não houve uma grande diferença entre o que foi delimitado no *Sprint Backlog* com relação ao resultado apresentado. Para o próximo *Sprint* será usado o *Backlog* da ferramenta para definição das tarefas que serão realizadas.

5.4 TERCEIRO *SPRINT*

Para o terceiro *Sprint,* os itens do backlog a serem realizados (Figura 14) foram extraídos do recém-concluído cadastro de item do *backlog* do projeto

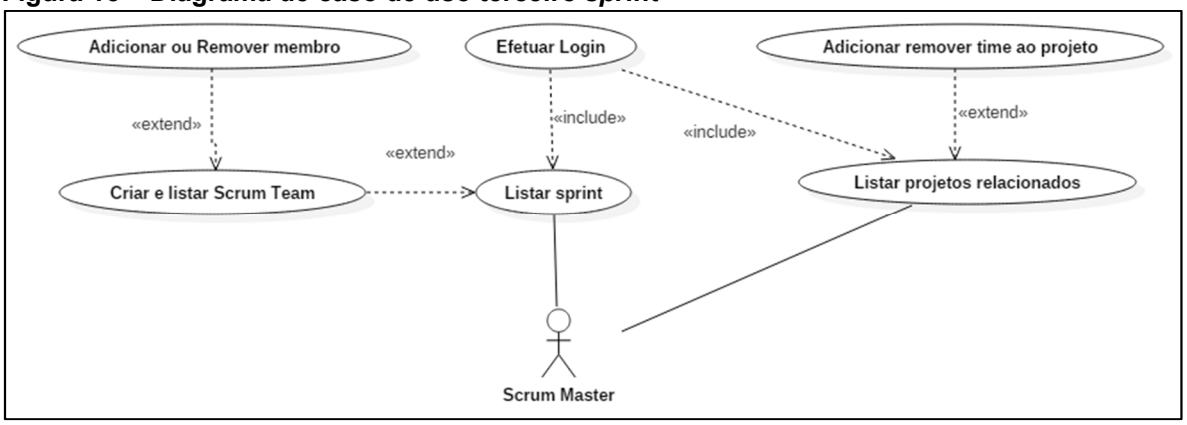
Figura 14 – Itens do b*acklog* selecionados na ferramenta para o terceiro *Sprint*

Alta	Scrum Master - Scrum Team	Scrum master deve criar time, adicionar e remover membro ao time.
Alta	Scrum Master - projeto	listar projeto e associar time a projeto

Fonte: Autoria própria

Foi criado um diagrama de casos de uso, apresentado na Figura 15, e um protótipo de baixa fidelidade, apresentado na Figura 16. Isso é recomendado porque melhora o entendimento das funcionalidades do *Scrum Master* e uso da ferramenta.

Figura 15 – Diagrama de caso de uso terceiro *sprint*



Fonte: Autoria própria

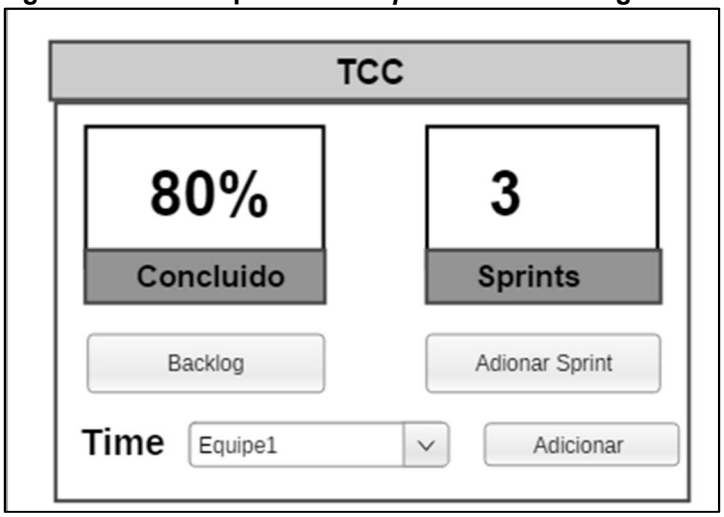
A Figura 15 exibe as funcionalidades do *Scrum Master.* Após efetuar *login* no sistema o usuário lista os projetos relacionados a sua conta. Tem as opões de criar e listar *Scrum Team,* listar *Sprint,* adicionar ou remover time ao projeto e adicionar ou remover membro ao *Scrum Team.*

Figura 16 – Protótipo do terceiro *sprint*: para funcionalidade *Scrum Master*

Fonte: Autoria própria

O protótipo da Figura 16 sugere que cada time seja listado em *cards*, cada *card* exibe os membros ao *Scrum Team* e possibilita a inserção de um novo membro. No topo da página que lista os *Scrum Team o protótipo* que deve ser construído um formulário para criar os *Scrum Team.* O protótipo da Figura 17 sugere a listagem de dados do projeto em um *card* e possui a as opções de listar e associar um time ao projeto em um campo no rodapé do *card*.

Figura 17 – Protótipo terceiro *sprint*: item backlog: Scrum Master - projeto



Fonte: Autoria própria

A partir da Figura 16 e da Figura 17 é possível definir a lista de tarefas do *Sprint* que cada item do backlog vai possuir, como apresentada no Quadro 9:

Quadro 9 – *Sprint Backlog* do terceiro *Sprint*

Item backlog	Tarefas
Scrum Master	Criar página de cadastro e listar de time (nome, time, *e-mail*)
Scrum Team	Criar tabela time (nome, data criação)
	Criar página para inserir e remover membro no time (via *e-mail*)
	Criar opção de remover time da lista do Scrum Master
	Validar campos e inserir os dados no banco de dados (*back-end*)
Srcum Master	Criar página para listar o projeto (contendo nome, link para *backlog*, equipe)
Projeto	Criar a opção de inserir ou remover time no projeto
	Criar tabela equipe (*scrum master*, *product owner*, time, projeto)
	Validar campos e inserir os dados no banco de dados

Fonte: Autoria própria

Uma vez definidas as características de cada tarefa, é possível usar a Figura 15, a Figura 16, a Figura 17 e o Quadro 9 para realizar a estimativa de complexidade do *Sprint* (Quadro 10).

Quadro 10 – Estimativa terceiro *Sprint*

Tipo de Função	Complexidade Funcional	Total do Tipo de função
ALI	02(Baixa) X 7	14
EE	02(Baixa) X 3	06
CE	02(Baixa) X 3	06
TOTAL		**26**

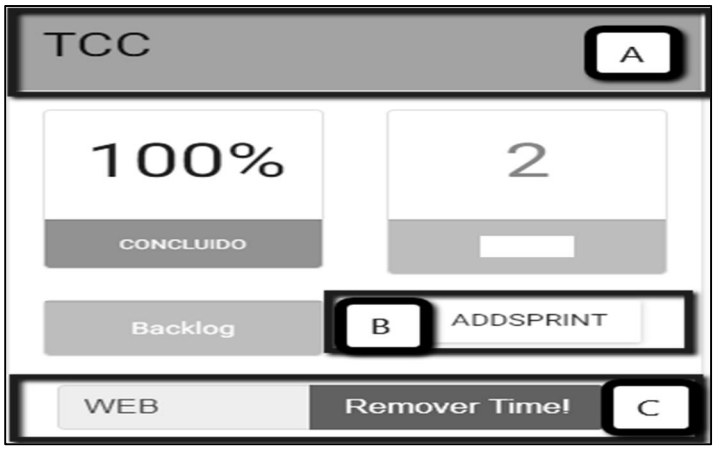
Fonte: Autoria própria

5.4.1 Resultado

Após realizar o desenvolvimento do *Sprint* tem-se os seguintes resultados:

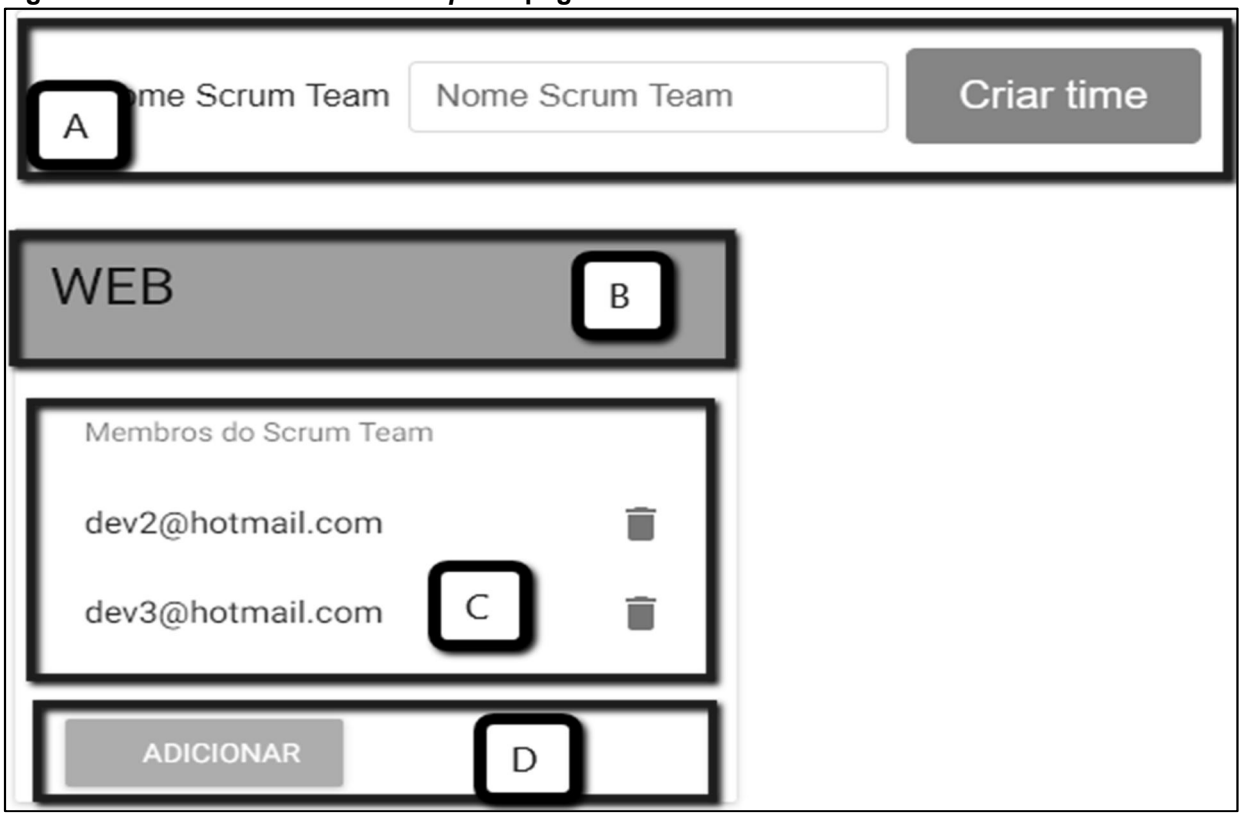
- A Figura 18 com a lista dos projetos que estão vinculados ao *Scrum Master* que acessou a ferramenta;
- Cada *card* é composto por um cabeçalho com o nome do projeto (Figura 18 (A));
- O corpo do *card* contém dados do projeto (somente exibição, não implementados);
- O formulário no rodapé do *card* (Figura 18 (C)) permite listar e vincular *Scrum Team* ao projeto. Caso o projeto possua *Scrum Team* vinculado é permitido exclui-lo.

Figura 18 – Resultado terceiro *sprint*: página listar projetos Scrum Master



Fonte: Autoria própria

Figura 19 – Resultado do terceiro S*print*: página listar/criar Scrum Team



Fonte: Autoria própria

A página para criar e listar o *Scrum Team* (Figura 18) possui um *layout* parecido com a página de projeto do Product Owner. O formulário para criar *Scrum Team* está localizado no topo da página (Figura 19 (A)) com o campo “Nome Scrum Team” e o botão “Criar Time”. Os *Scrum Team* são listados em *card* que possuem um cabeçalho com o nome do projeto (Figura 19 (B)), corpo do *card,* lista dos membros do *Scrum Team* (Figura 19 (C)) e no rodapé do *card* o botão “Adicionar“ (Figura 19 (D)) que ao ser clicado abre uma janela (Figura 20) que permite busca ou inserção de um membro.

Figura 20 – Resultado do terceiro S*print*: janela adicionar membro ao *Scrum Team*

Fonte: Autoria própria

Esse *Sprint* foi concluído em 30 horas, para uma estimativa de 26 pontos de função. Não foi realizada uma revisão nas estimativas de pontos de função, pois o resultado do *Sprint* está de acordo com o proposto no início. Após os testes básicos para verificação de inconsistências, foi realizado o *deploy* desse projeto no Heroku.

5.5 QUARTO SPRINT

Para o quarto *Sprint* será realizado o desenvolvimento dos itens do *Backlog* selecionados na ferramenta conforme a Figura 21. Estes itens visam funcionalidades ligadas ao usuário *Scrum Master*.

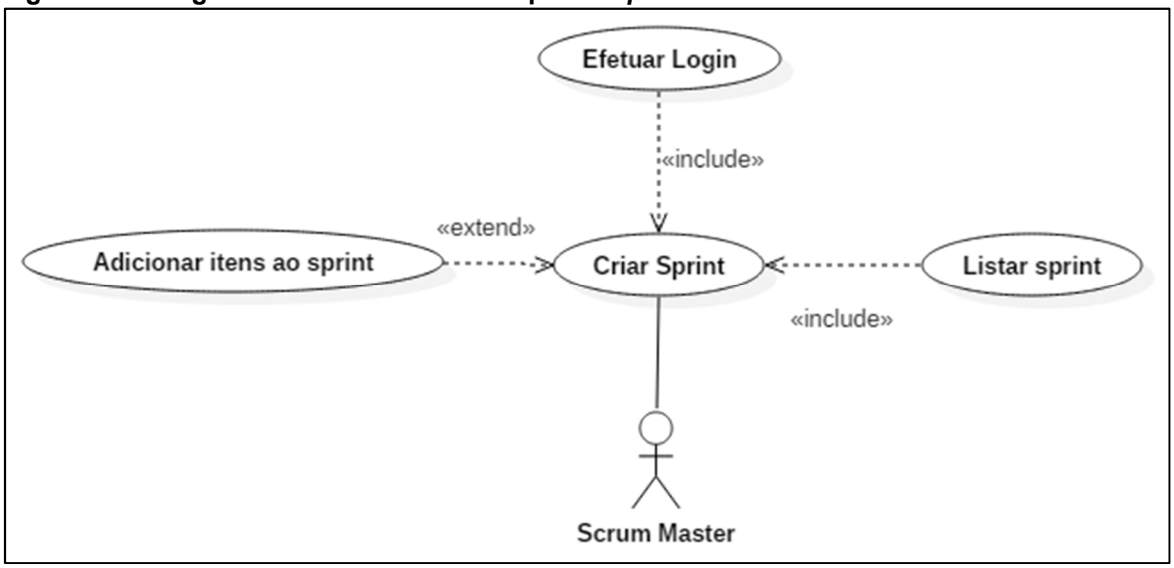
Figura 21 – Itens do *backlog* selecionados na ferramenta para o quarto *Sprint*

Alta	Scrum Master - reuniao backlog	Scrum master deve listar o backlog de um projeto e reunido com o time decidir o Sprint backlog
Alta	Scrum Master - criar tarefa	Scrum master deve, junto com a equipe, criar tarefa para cada item do backlog
Alta	Scrum Master - sprint	Scrum master deve listar o sprint e ter acesso a suas tarefas

Fonte: Autoria própria

A Figura 22 apresenta o caso de uso com as funcionalidades que o usuário *Scrum Master* irá ter acesso a partir do quarto *Sprint.* Os protótipos apresentados na Figura 23 e na Figura 24 sugerem um modelo para o desenvolvimento do *Sprint.* Outras tarefas não precisaram de um protótipo devido a sua baixa complexidade, como *Scrum Master*–Sprint.

Figura 22 – Diagrama de caso de uso do quarto *Sprint*



Fonte: Autoria própria

A Figura 22 demonstra algumas funcionalidades do *Scrum Master.* Após efetuar *login* este pode adicionar itens ao *Sprint,* criar *Sprint* ou listar *Sprint.*

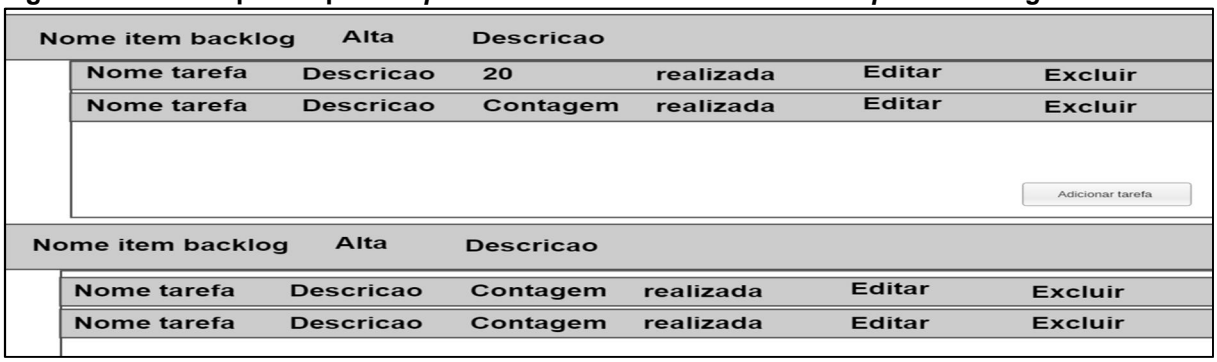
A Figura 23 apresenta o protótipo de baixa fidelidade para a ação do *Scrum Master* de adicionar o *Sprint* e selecionar os itens do *Backlog.*

Figura 23 – Protótipo do quarto *Sprint: Scrum Master* selecionar itens do backlog e criar *Sprint*

Fonte: Autoria própria

A Figura 24 apresenta o protótipo de baixa fidelidade que lista os itens do backlog selecionados para o *Sprint* e suas respectivas tarefas.

Figura 24 – Protótipo do quarto *Sprint*: *Scrum Master* lista itens do *Sprint Backlog*



Fonte: Autoria própria

Quadro 11 – *Sprint* Backlog do quarto *Sprint*

Item backlog	**Tarefas**
Scrum Master **Reunião B*acklog***	Criar página para listar itens do *Backlog* e permitir selecionar os itens
	Criar tabela sprint (nome, data criação, data entrega, tarefa-backlog)
	Validação de campos e inserção de dados no banco (*back-end*)
Srcum Master **Listar item selecionado para o *Sprint Backlog***	Criar página de listar itens *sprint backlog*
	Criar a opção de inserir (somente a opção) e listar tarefas referente ao item selecionado
	Criar tabela sprint-tarefa (nome, descrição, data criação, data entrega, pontos, sprint)
	Validação de campos e inserção de dados no banco
Scrum Master **Listar *Sprint***	Criar página para listar sprints do projeto e permitir acessar suas tarefas

Fonte: Autoria própria

Baseando nas figuras: Figura 22, Figura 23 e Figura 24 é possível construir o Quadro 11, apresentando os itens do *backlog* e suas respectivas tarefas.

Quadro 12 – Estimativa quarto *Sprint*

Tipo de Função	**Complexidade Funcional**	**Total do Tipo de função**
ALI	02(Baixa) X 7	14
EE	01(Baixa) X 3	03
SE	03(Baixa) X 4	12
TOTAL		**29**

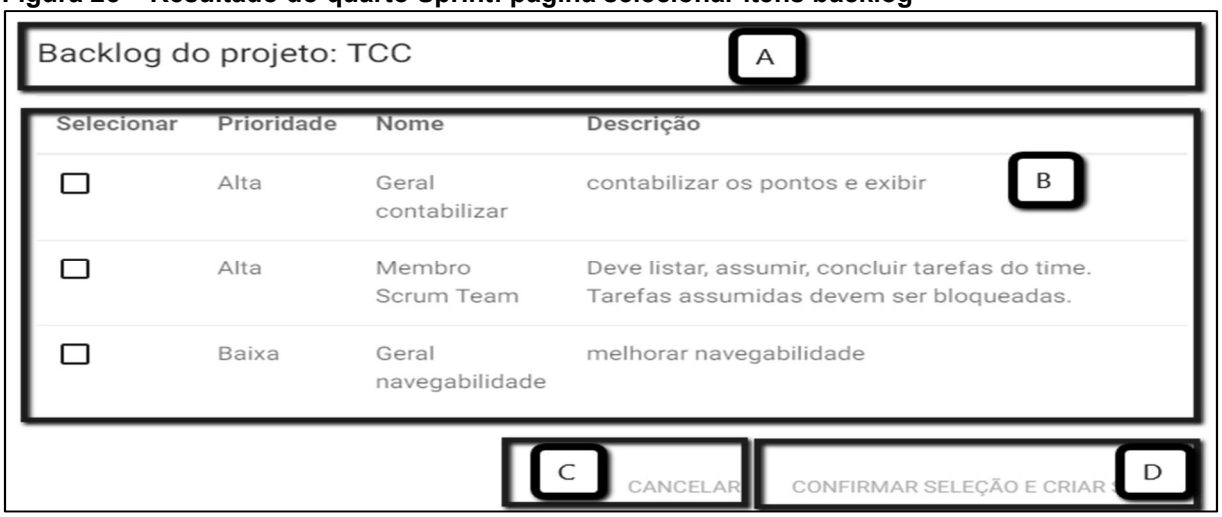
Fonte: Autoria própria

O Quadro 12 apresenta as estimativas do quarto *Sprint*. Essas estimativas foram geradas a partir da análise do caso de uso, os protótipos e o Sprint Backlog, citados anteriormente.

5.5.1 Resultado

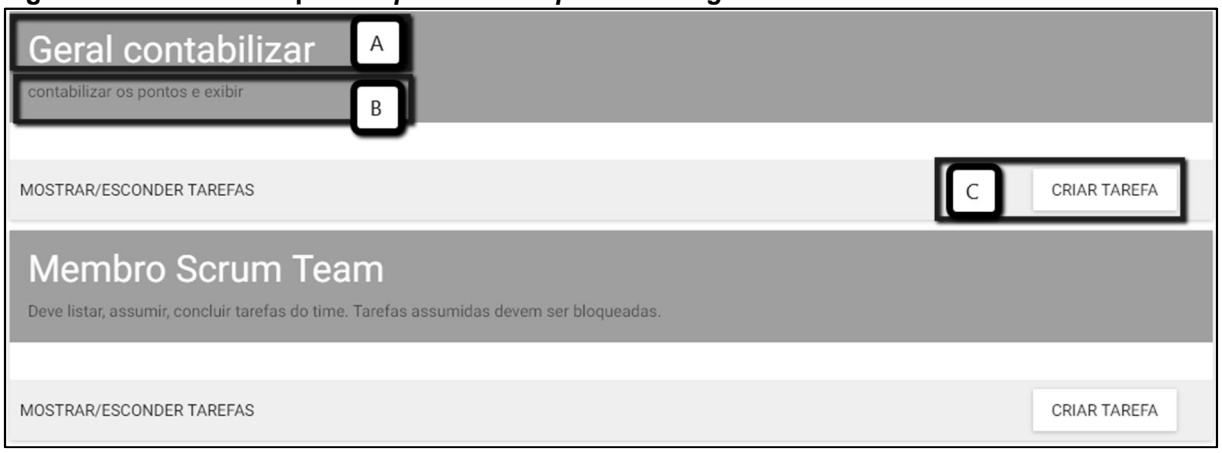
Ao final do quarto *Sprint*, o botão “ADDSPRINT” (Figura 18 (B)) recebe a funcionalidade para abrir uma janela com a lista de itens do *backlog* (Figura 25) e as opções de “CANCELAR” (Figura 25 (C)) ou “CONFIRMAR SELEÇÃO E CRIAR SPRINT” (Figura 25 (D)). Essa janela possui um título com o nome do projeto (Figura 25 (A)) e a lista de itens do *backlog* do projeto (Figura 25 (B)). Após a seleção dos itens e a confirmação do usuário, o sistema direciona para a página apresentada na Figura 26. Essa página lista os itens do *Sprint backlog* em *cards,* o título de cada *card* contém nome (Figura 26 (A)) e a descrição (Figura 26 (B)) do item do *backlog.* No rodapé do *card* o botão “CRIAR TAREFA” (Figura 26 (C)) receberá funcionalidade nos próximos passos.

Figura 25 – Resultado do quarto Sprint: página selecionar itens backlog



Fonte: Autoria própria

Figura 26 – Resultado quarto *Sprint:* listar *sprint backlog*



Fonte: Autoria própria

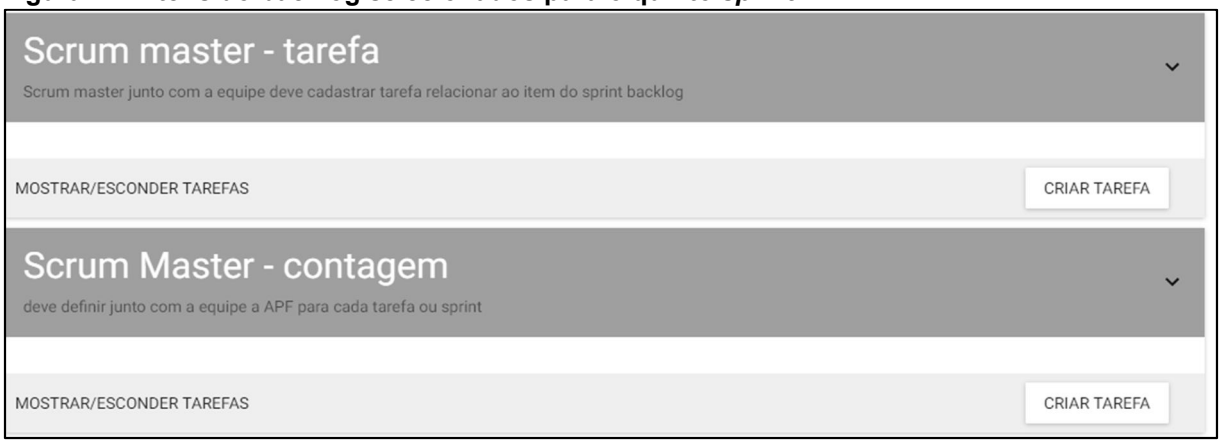
Para este *Sprint* foram estimados 29 PF e foi realizado em 25 horas. Após os testes de consistência foi realizado um *deploy* do projeto no *Heroku.* A partir desta etapa é possível construir o *backlog* dentro da própria ferramenta.

5.6 QUINTO SPRINT

Para o quinto *Sprint* foram selecionados, na própria ferramenta, os itens do *backlog* cadastros conforme apresenta a

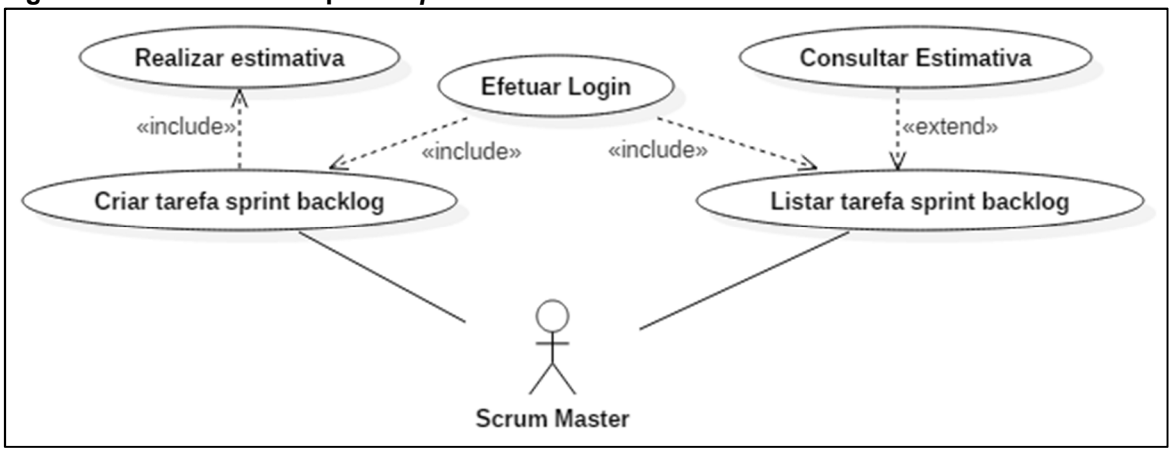
Figura 27: "*Scrum master* – tarefa*"* e *"Scrum master* - contagem*".* Essas funcionalidades estão ligadas ao usuário Scrum Master.

Figura 27 – Itens do backlog selecionados para o quinto *sprint*



Fonte: Autoria própria

Figura 28 – Caso de uso quinto *sprint*

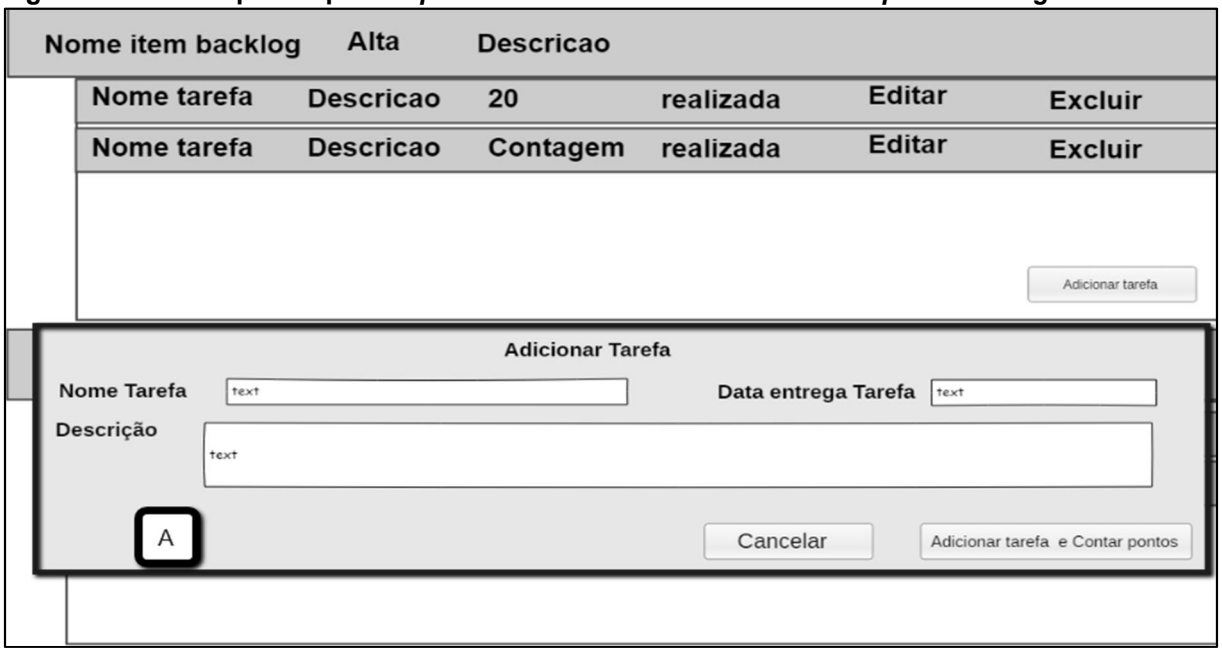


Fonte: Autoria própria

O diagrama de caso de uso (Figura 28) representa as funcionalidades que serão desenvolvidas para o usuário Scrum Master nesta etapa. Após efetuar *login* o usuário pode criar tarefa para o *Sprint Backlog* e realizar estimativas. Usuário pode listar tarefas do *Sprint Backlog* e consultar estimativas.

O protótipo apresentado na Figura 29 é utilizado para esclarecer funcionalidades. Ele sugere exibir um formulário para adicionar uma tarefa conforme apresentado na Figura 29(A)). Após o preenchimento deste formulário o usuário é encaminhado para a página que contém as estimativas.

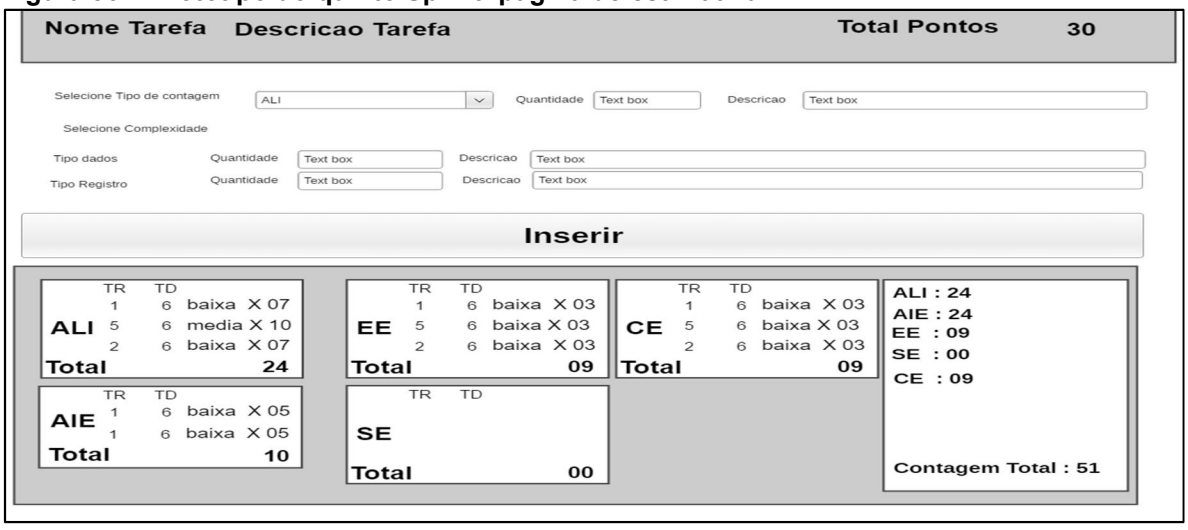
Figura 29 – Protótipo do quinto *sprint*: cadastrar a tarefa no item do *sprint backlog*



Fonte: Autoria própria

A página para inserir a estimativa é sugerida para ser construída por um formulário conforme apresenta a Figura 31. Esse formulário contém os campos referentes à APF.

Figura 30 – Protótipo do quinto Sprint: página de estimativa



Fonte: Autoria própria

Utilizando os protótipos é possível elaborar o *sprint backlog* para esta etapa conforme apresenta o Quadro 11Quadro 13:

Quadro 13 – *Sprint Backlog* do quinto *Sprint*

Item backlog	Tarefas
Scrum Master **Tarefa**	Criar página para inserir tarefa (nome, data prevista entrega, descrição), direcionar para página de estimativa
	Criar tabela sprint-tarefa (nome, data criação, data entrega, desenvolvedor tarefa-backlog, pontos)
	Validar os campos e a inserção de dados no banco de dados (*back-end*)
Srcum Master **Estimativa**	Criar página de estimativa de pontos (selecionar: ALI, AIE, EE, SE ou CE, TD e TR) quantidade e descrição, preencher tabela com total pontos
	Criar a opção de inserir (somente a opção) e listar tarefas referente ao item selecionado
	Criar a tabela de pontos (tarefa, estimativa)
	Criar a tabela de estimativa (tarefa, ALI, AIE, EE, CE, SE, TD e TR)
	Validação de campos e inserção de dados no banco

Fonte: Autoria própria

Para realizar a estimativa utiliza-se o *Sprint backlog, os* protótipos e o caso de uso. Dessa forma é possível ter mais sucesso na elaboração da estimativa conforme apresenta o Quadro 14.

Quadro 14 – Estimativa quinto *sprint*

Tipo de Função	Complexidade Funcional	Total do Tipo de função
ALI	03(Baixa) X 7	21
EE	02(Média) X 4	08
SE	02(Alta) X 7	14
TOTAL		**43**

Fonte: Autoria própria

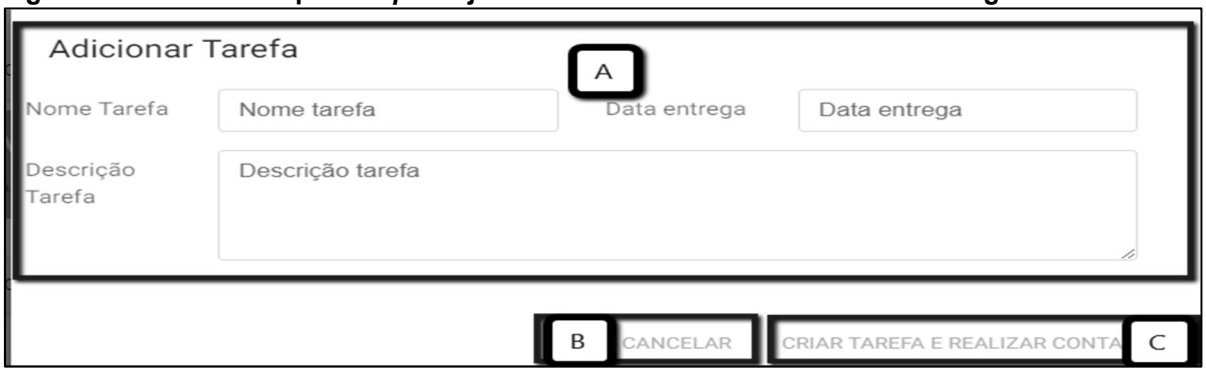
Para este *Sprint* foram totalizados 43 pontos de função distribuídos entres os itens presentes no *Backlog* do produto.

5.6.1 Resultado

É possível analisar o resultado do quinto *Sprint* na Figura 31, Figura 32 e Figura 33. Na Figura 31 é apresentada uma janela que é exibida após o clique em “CRIAR TAREFA” da Figura 26 (C). Esta janela possui um formulário (Figura 31 (A)), o botão “CANCELAR” (Figura 31 (B)) e o botão “CRIAR TAREFA E REALIZAR

CONTAGEM” (Figura 31 (C)). Estas funcionalidades permitem criar uma tarefa e ser direcionado para a página de contagem (Figura 32).

Figura 31 – Resultado quinto *Sprint*: janela adicionar tarefa ao item do backlog



Fonte: Autoria própria

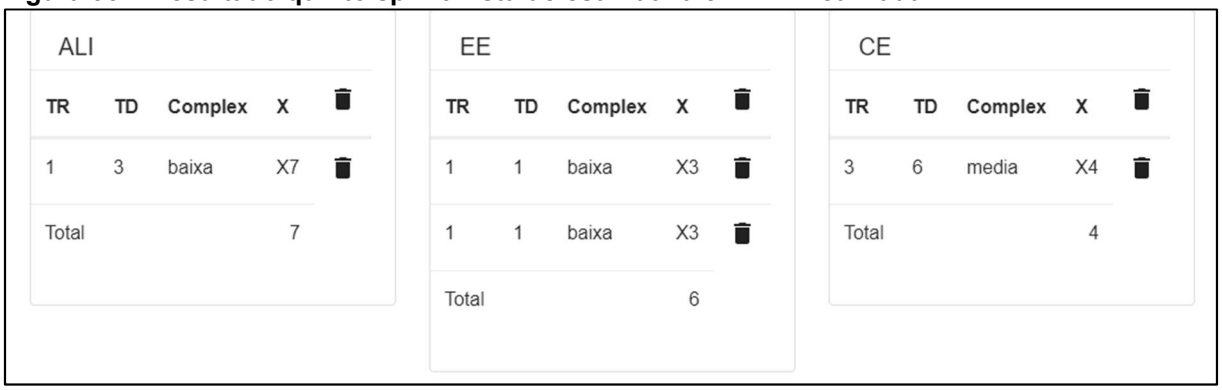
Figura 32 – Resultado quinto sprint: formulário de estimativa APF

Fonte: Autoria própria

Para realizar a estimativa foi desenvolvida uma página apresentada na Figura 32 que é composta por: cabeçalho, formulário e *cards.* O cabeçalho (Figura 32 (A)) exibe o nome e a descrição da tarefa. O formulário é dividido em conjunto de dados (Figura 32 (B)), que possui os seguintes campos: “Tipo de contagem” que possui as seguintes opções: ALI, AIE, EE, SE e EE, esses campos são referentes às características da APF e complexidade (Figura 32 (C)) com os seguintes campos: tipo de dados e tipo registro para inserção da quantidade e descrição.

A estimativa (Figura 33) é exibida abaixo do formulário após inserir os dados e enviar para validação no servidor. A estimativas são agrupadas de acordo com as caraterísticas de APF(ALI, AIE, EE, SE e CE).

Figura 33 – Resultado quinto sprint: lista de estimativa em APF realizada



Fonte: Autoria própria

Este *Sprint* foi realizado em 20 horas para 43 pontos de função. As funcionalidades desenvolvidas neste *Sprint* podem ser utilizadas para os próximos passos de desenvolvimento da ferramenta, como a estimativa de complexidade da tarefa.

5.7 SEXTO SPRINT

No sexto *Sprint* foram desenvolvidas as funcionalidades para o membro do *Scrum Team* (Figura 34 – Itens do backlog selecionados na ferramenta para o sexto sprint). O item do *backlog "*Geral Contabilizar" não possui protótipo ou diagrama de caso de uso, pois se trata de um item que pode ser estimado utilizando os dados de Sprints anteriores.

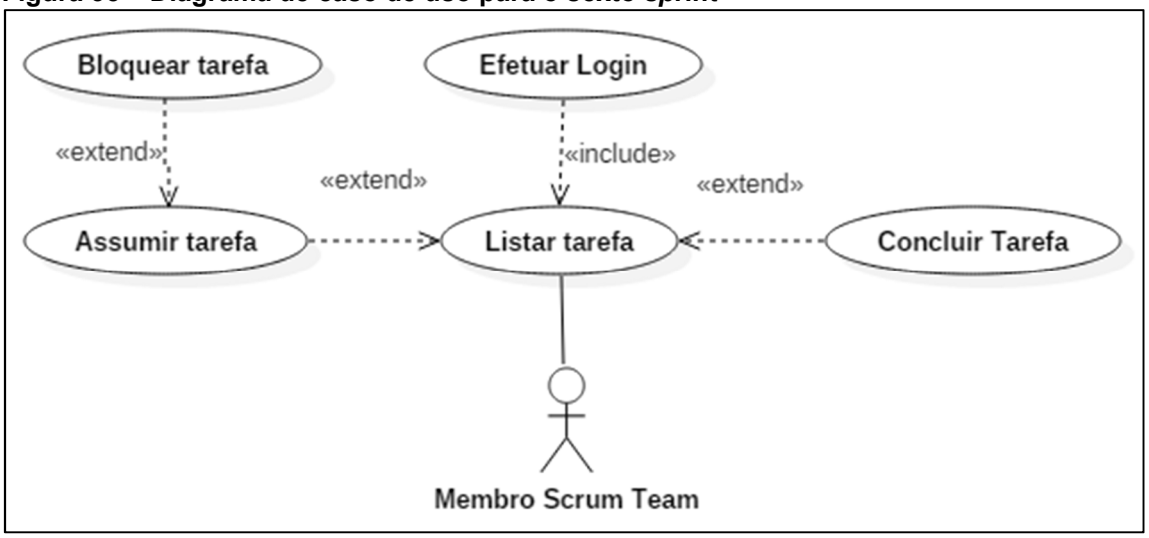
Figura 34 – Itens do backlog selecionados na ferramenta para o sexto sprint

Alta	Geral contabilizar	contabilizar os pontos e exibir
Alta	Membro Scrum Team	Deve listar, assumir, concluir tarefas do time. Tarefas assumidas devem ser bloqueadas.

Fonte: Autoria própria

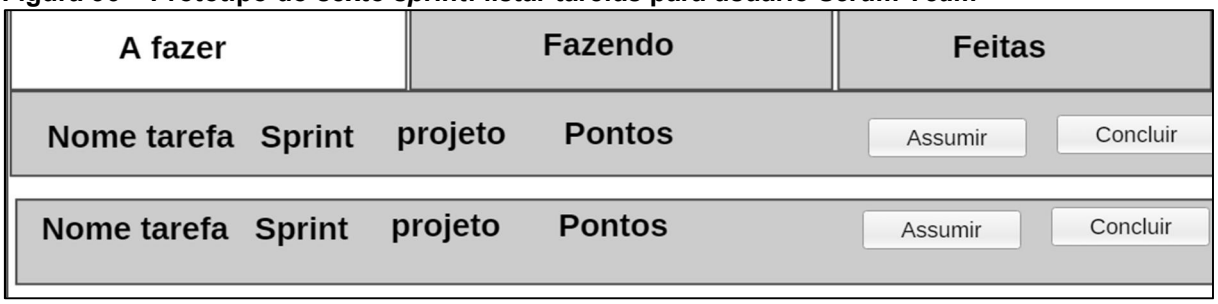
O caso de uso da Figura 35 e o protótipo da Figura 36 auxiliam a definição das tarefas. Para o item “membro *Scrum Team”* foi criada a tarefa “Listar tarefa” que foi estimada em 17 pontos de função (Figura 38).

Figura 35 – Diagrama de caso de uso para o sexto *sprint*



Fonte: Autoria própria

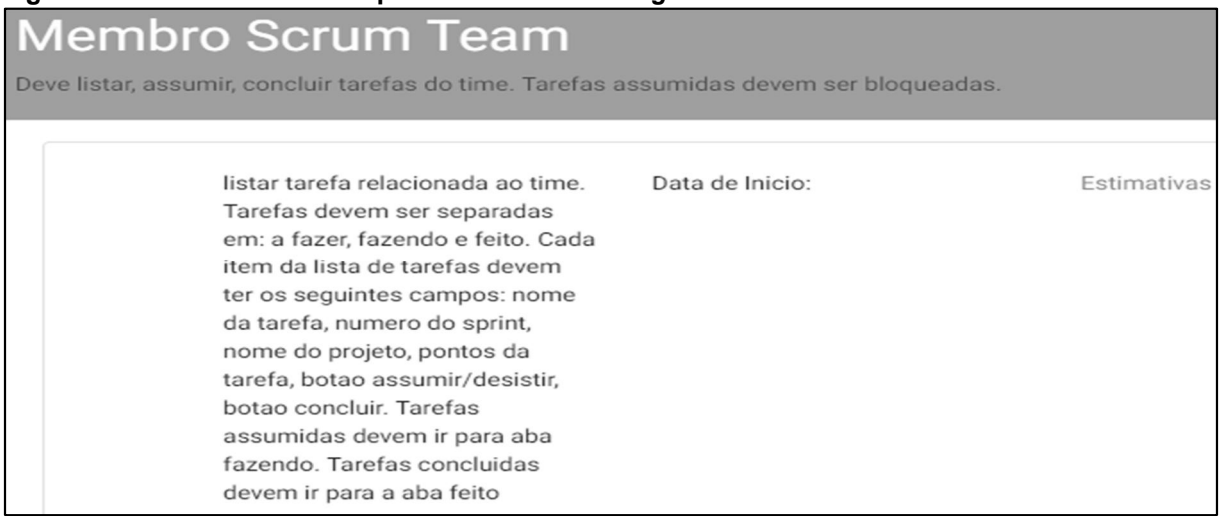
Figura 36 – Protótipo do sexto *sprint*: listar tarefas para usuário S*crum Team*



Fonte: Autoria própria

A Figura 37 apresenta as tarefas definidas para o item backlog do membro *Scrum Team*. A Figura 38 exibe as estimativas realizadas para uma tarefa do *Sprint Backlog.*

Figura 37 – Tarefas definidas para o item do *backlog*: Membro *Scrum Team*



Fonte: Autoria própria

Figura 38 – Estimativa detalhada para a tarefa: Listar tarefa do item Membro *Scrum Team*

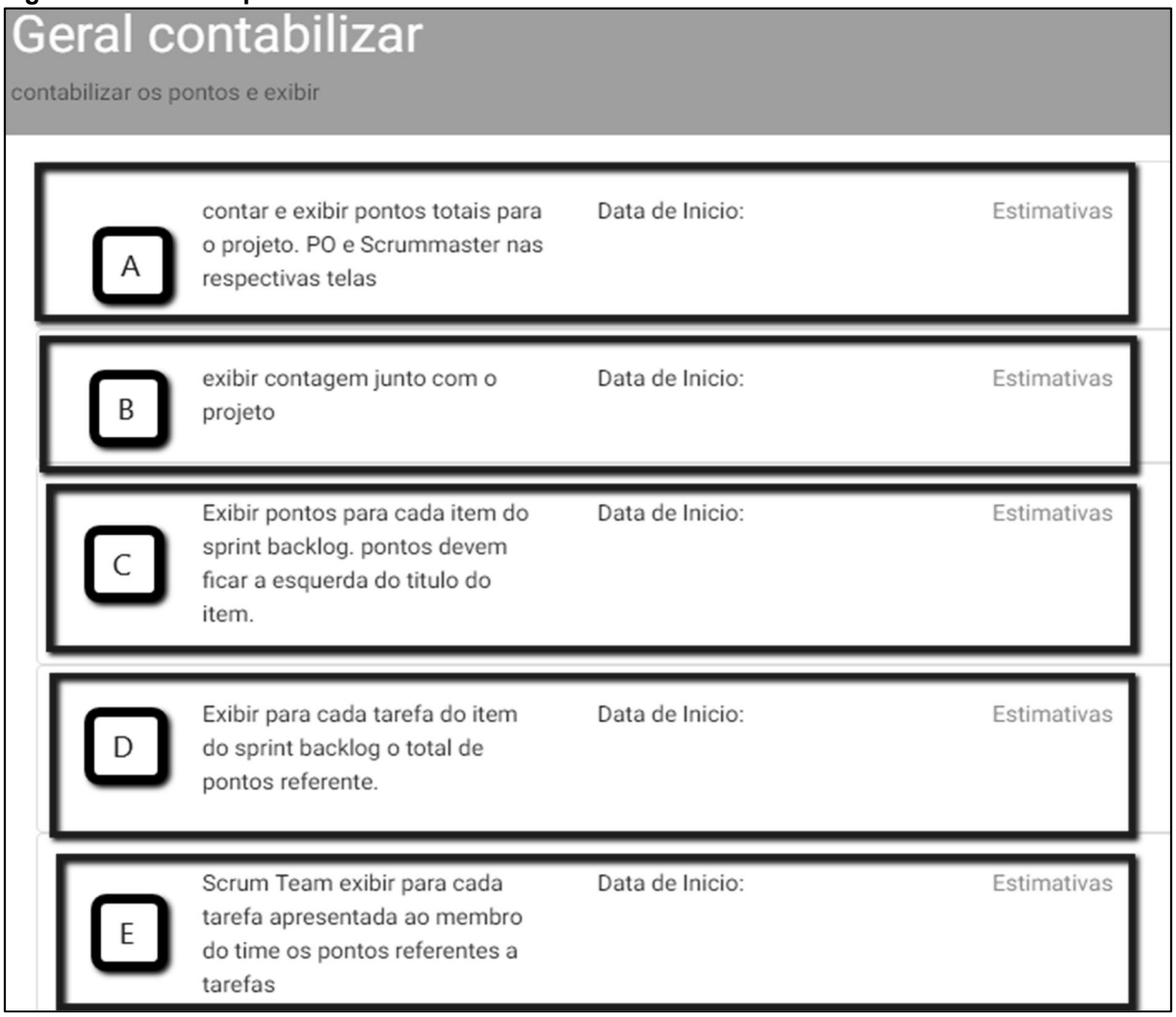
Fonte: Autoria própria 1

Para o item “Geral contabilizar” foram definidas as seguintes tarefas:

- Pontos projeto (Figura 39 (A))
- Pontos projeto *Scrum Master*(Figura 39 (B))
- Pontos item backlog (Figura 39 (C))
- Pontos para cada tarefa do item do sprint Backlog (Figura 39 (D))
- Pontos para cada tarefa do membro do *Scrum Master* (Figura 39 (E))

A primeira tarefa (Figura 39 (A)) foi estimada em 6 pontos de função e as demais em 3 pontos de função, totalizando 18 PF para este item do backlog. A segunda tarefa (Figura 37 – Tarefas definidas para o item do *backlog*: Membro *Scrum Team*) foi estima em 17 PF, exemplificados na Figura 38 – Estimativa detalhada para a tarefa: Listar tarefa do item Membro *Scrum Team*. Dessa forma esse Sprint ficou com 35 pontos de função.

Figura 39 – Tarefas para o item: Geral Contabilizar



Fonte: Autoria própria

5.7.1 Resultado

Neste Sprint foram definidas as funcionalidades do membro *Scrum Team.* Estas funcionalidades são: listar tarefas referentes ao time, assumir tarefa, desistir da tarefa e concluir tarefa.

- A tarefa pode estar na guia “À FAZER” (Figura 40(A)) onde o usuário pode assumir a tarefa clicando no botão “ASSUMIR” (Figura 40 (B)).
- As tarefas em execução pela equipe estarão na aba “FAZENDO” (Figura 41 (A)), nesta aba o usuário pode utilizar o botão “DESISTIR” (Figura 41 (B)) para que a tarefa retorne a aba “À FAZER” e o botão “CONCLUIR” (Figura 41 (C)) para que a tarefa seja movida para aba “FEITO” (Figura 42 (A)).

- Na aba “FEITO” ficam as tarefas concluídas pelo *Scrum Team*, essas tarefas ficam aguardando a aprovação do Product Owner.

Figura 40 – Resultado sexto *sprint*: tarefas aba à fazer membro *Scrum Team*

Fonte: Autoria própria

Figura 41 – Resultado sexto sprint: tarefas aba fazendo membro Scrum Team

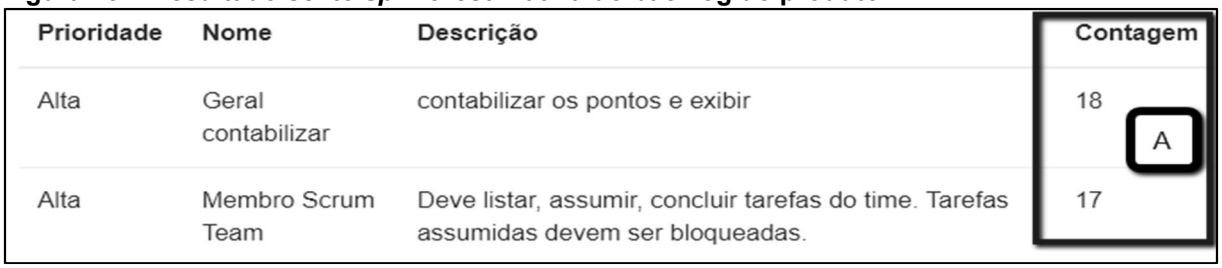
Fonte: Autoria própria

As tarefas do item “Contagem geral” foram realizadas em várias páginas e retornam o número de pontos de função (Figura 42 (A)). Esses pontos estão agrupados por itens do backlog ou tarefa do Sprint backlog.

Figura 42 – Resultado sexto *sprint*: tarefas aba feito membro Scrum Team

Fonte: Autoria própria

Figura 43 – Resultado sexto *sprint*: estimativa do backlog do produto



Prioridade	Nome	Descrição	Contagem
Alta	Geral contabilizar	contabilizar os pontos e exibir	18
Alta	Membro Scrum Team	Deve listar, assumir, concluir tarefas do time. Tarefas assumidas devem ser bloqueadas.	17

A

Fonte: Autoria própria

O sexto Sprint foi realizado em 15 horas, estimado em 35 PF e adicionou na ferramenta as funcionalidades do membro do Scrum Team e a exibição das estimativas para cada usuário. Como a estimativa condiz com o resultado não será realizada a revisão da estimativa.

5.8 COMPARAÇÃO DE MÉTODOS DE ESTIMATIVA

Após concluir a ferramenta é possível comparar a técnica proposta com o Scrum que utiliza estimativas em *Planning Poker*, conforme o

Quadro 15 foi escolhido para comparação por se tratar da estimativa mais indicada para uso com o Scrum.

Quadro 15 – Comparação de estimativas com *Planning Poker* e APF em *Scrum*

Características	Scrum -APF	Scrum – Planning Poker
Permite revisar as estimativas?	Sim	Não
Exige experiência do *Scrum Team,* em relação a tecnologia, na tomada das estimativas?	Não	Sim
Mantém o *Scrum Team* independente?	Sim	Sim
Pode ser empregada a qualquer momento no projeto?	Sim	Não

Fonte: Autoria própria

A técnica proposta pode ser utilizada por um *Scrum Team* sem experiência com a tecnologia adotada no projeto. Pode ser revisada pelo *Scrum Team* ou uma parte interessada no projeto a qualquer momento da etapa de desenvolvimento. Diferentemente do *Planning Poker,* pode ser empregada a qualquer momento no projeto. As duas técnicas de estimativas mantêm a independência do *Scrum Team*, compartilhando esse ponto em comum.

Analisando a comparação é possível verificar que a técnica proposta pode vir a substituir o *Planning Poker* sem a perda das principais características do Scrum. Dessa forma, uma equipe que já utilize o *framework* reduz a curva de aprendizado para aplicar a técnica.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este Capítulo apresenta a conclusão e os trabalhos futuros que podem ser desenvolvidos a partir desse trabalho. A Seção 6.1 exibe a conclusão. A Seção 6.2 apresenta os trabalhos futuros.

6.1 CONCLUSÃO

Este trabalho propôs uma técnica de estimativa com APF em projetos *Scrum*. Para exemplificar seu uso foi desenvolvida uma ferramenta que foi útil para a aplicação da técnica e também foi utilizada para exemplificar a própria aplicabilidade da técnica. Essa ferramenta está disponível para uso *online* sob a licença GPL, para que a comunidade de desenvolvimento tenha acesso ao código fonte e possa introduzir melhorias.

Utilizando os dados disponibilizados no decorrer do desenvolvimento da ferramenta é possível construir um histórico resumido do projeto.Utilizando o histórico o *Scrum Team, Product Owner* e partes interessadas podem ser mais assertivas quanto aos prazos e custos de construção de uma aplicação similar.

A técnica se mostrou efetiva para a construção da ferramenta e não alterou a dinâmica de desenvolvimento. É possível destacar que as estimativas utilizadas podem sofrer uma revisão por terceiros, tornando a estimativa de complexidade transparente para gerentes e partes interessadas.

A ferramenta pode ser utilizada para construção de outras aplicações, facilitando dessa forma a aplicação da técnica e verificação de histórico de estimativas.

6.2 TRABALHOS FUTUROS

A ferramenta foi construída de forma que suas funcionalidades possam ser expandidas em trabalho futuros. Foi criado um repositório público no *Github*[11] e todas

[11] Link do repositório: < https://github.com/luizcarlospedrosogomes/APF-SCRUM>

as tecnologias utilizadas são *open-source* e contam com ampla documentação. Ainda é necessário validar a técnica em trabalhos futuros.

Existem necessidades de aprimorar a ferramenta em diversos aspectos que podem ser desenvolvidos em trabalhos futuros como:

- *Scrum Master* pode criar reuniões (*Sprint* Backlog, Revisão do *Sprint*)
- *Product Owner* pode aceitar ou rejeitar tarefas
- *Scrum Master* pode enviar convite para reunião
- Integrar *login* da ferramenta com outros aplicativos (*Google, Facebook, GitHub*)
- Melhorar layout e usabilidade
- Usar o *GitHub* como repositório de cada projeto cadastrado
- Aprimorar o formulário de estimativas
- Gerar um histórico de projetos

A técnica aplicada para o desenvolvimento da ferramenta carece de uma avaliação mais ampla, assim como a própria ferramenta. Para executar tais avaliações se sugere os seguintes trabalhos futuros:

- Aplicar a técnica a equipes *Scrum*
- Validar a técnica
- Aplicar a técnica com pontos de ajustes

REFERÊNCIAS

AGUIAR, M.; SYMONS, C.; VLIET, E. V**.** Gerenciando Ágil em Escala**:** Um resumo para Executivos de Software e Chief Information Officers. 2017. Disponível em: <http://www.ifpug.org/wp-content/uploads/2017/08/Agile-and-FSM-for-CIOs-pt-br.pdf>. Acesso em: 08 set. 2017.

ALBINO, J. P. et al. Design de interfaces para web baseados no sistema de grade do boostrap 3. **II World Congress On Systems Engineering And Information Technology,** Vigo, v. 5, n. 8, p.150-154, 19 nov. 2015.

ALECRIM, E. O que é ERP e para que serve? 2017. Disponível em: <http://sistemaserp.org/o-que-e-erp/>. Acesso em: 29 set. 2017.

BAZILIO, E. Framework: o que é e como ele pode ajudar? 2015. Disponível em: <http://blog.infolink.com.br/framework-o-que-e-e-como-ele-pode-ajudar/>. Acesso em: 29 set. 2017.

BEEDLE, M. et al**.** Manifesto Ágil. [2001]. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 12 ago. 2017.

BOMFIM, M R G; ANDRADE, J R. **Guia de Contagem de Pontos de Função do MP**. Brasília: Ministério do Planejamento, 2015. 45 p. (1).

CAMPOS, A. C. Open Source é: Você sabe mesmo o que é Open Source? 2009. Disponível em: <http://www.linuxnewmedia.com.br/images/uploads/pdf_aberto/LM_53_14_15_01_col_augusto.pdf>. Acesso em: 29 set. 2017.

CARNEIRO, A. P.; MOREIRA, J. L.; FREITAS, A. L. C. **Técnicas de Otimização de Banco de Dados Um Estudo Comparativo:** Mysql e Postgresql. 2011. 10 f. Monografia (Especialização) - Curso de Ciência da Computação, Furg, Rio Grande, 2011.

CARVALHO, V. **Postgres:** Banco de dados para aplicações web modernas. São Paulo: Casa do Codigo, 2017.

CAVALCANTE, R. CSS**.** 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 29 ago. 2017.

COCKBURN, A. **Escrevendo Casos de Usos Eficazes:** Um guia prático para desenvolvedores de software. São Paulo: Bookman, 2007.

COHN, M. **Desenvolvimento de Software com Scrum:** Aplicando métodos ágeis com sucesso. Porto Alegre: Bookman, 2011.

COMER, D. E. **Redes de Computadores e Internet.** 6. ed. Porto Alegre: Bookman, 2016.

CRUZ, F. **O guia completo do Scrum e da agilidade em projetos:** Conquiste sua certificação e aprenda a usar métodos ágeis no seu dia a dia. Rio de Janeiro: Brasport, 2015.

CRUZ, F. Scrum e o papel do Scrum Master. 2012. Disponível em: <http://www.devmedia.com.br/scrum-e-o-papel-do-scrum-master/26626>. Acesso em: 12 ago. 2017.

DIMES, T. **Scrum Essencial.** [s.l]: Babelcube, 2014.

EIS, D.; FERREIRA, E. **HTML5 e CSS3:** com pimenta e farinha. São Paulo: Lulu, 2012.

ESTROZI, L. F. et al. **Programação para Internet com PHP.** Rio de Janeiro: Brasport, 2010.

FACEBOOK. React. 2017. Disponível em: <https://facebook.github.io/react/>. Acesso em: 30 ago. 2017.

FEITOSA, M. P. **Fundamentos de Banco de Dados.** São Paulo: Clube dos Autores, 2008.

FONSECA, D. Conceitos básicos sobre Metodologias Ágeis para Desenvolvimento

de Software (Metodologias Clássicas x Extreme Programming). 2008. Disponível em: <http://www.devmedia.com.br/conceitos-basicos-sobre-metodologias-ageis-para-desenvolvimento-de-software-metodologias-classicas-x-extreme-programming/10596>. Acesso em: 16 ago. 2017.

FRASSON, R. **Turbine seu CSS:** Folhas de estilo inteligentes com Sass. São Paulo: Casa do Código, 2016.

GARZIA, A. A. Javascript. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 29 ago. 2017.

GUEDES, G. **UML2:** Uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.

HEROKU. Cloud Application Plataform. 2017. Disponível em: <https://www.heroku.com/home>. Acesso em: 28 ago. 2017.

IFPUG. Como pontos de função ajudar projetos de metodologia ágil. 2017. Disponível em: <http://www.ifpug.org/how-function-points-help-agile-methodology-projects/?lang=pt>. Acesso em: 24 set. 2017.

IFPUG. Manual de Práticas de Contagem de Pontos de Função. **International Function Pont Users Group(IFPUG)**, v. 4.3.1, 2010.

JUNIOR, A. R. Scrum–Mais um caso de sucesso. 2013.

LEITE, F. F. Ranking das linguagens de programação. 2016. Disponível em: <http://www.furtadoleite.com.br/ranking-das-linguagens-de-programacao/>. Acesso em: 28 ago. 2017.

LICHIRGU, A. G. F. B. Contagem de pontos de função. 2016. Disponível em: <http://www.devmedia.com.br/contagem-de-pontos-de-funcao/34390>. Acesso em: 28 ago. 2017.

LOVI, R**.** Qual a diferença entre ux e ui? Mas não é tudo a mesma coisa? Talvez seja, ou será que não? 2017. Disponível em: <http://www.raffcom.com.br/blog/qual-a-diferenca-entre-ux-e-ui/>. Acesso em: 29

set. 2017.

MACORATTI, J. C. Análise de Pontos por Função**:** O Processo de contagem. [2010]. Disponível em: <http://www.macoratti.net/apf_pcta.htm>. Acesso em: 29 ago. 2017.

MANSUR, R. **Governança de TI:** Metodologias, Frameworks e Melhores Práticas. Rio de Janeiro: Brasport, 2007.

MARTINS, J. C. C. **Técnicas Para Gerenciamento de Projetos de Software.** Rio de Janeiro: Brasport, 2007.

MARTINS, L. Client-side e Server-side. 2015. Disponível em: <https://www.gigasystems.com.br/artigo/60/client-side-e-server-side>. Acesso em: 27 set. 2017.

MATOS, F V. Pontos de função e metodologias ágeis conseguem conviver juntas? 2010. Disponível em: <https://imasters.com.br/artigo/18137/agile/pontos-de-funcao-e-metodologias-ageis-conseguem-conviver-juntas/?trace=1519021197&source=single>. Acesso em: 04 out. 2017.

MEDEIROS, H. Introdução a Engenharia de Requisitos. 2013. Disponível em: <http://www.devmedia.com.br/introducao-a-engenharia-de-requisitos/29454>. Acesso em: 18 ago. 2017.

MICHAELIS. Dicionário Brasileiro da Língua Portuguesa**.** 2017. Disponível em: <http://michaelis.uol.com.br/busca?r=0&f=&t=&palavra=metodologia>. Acesso em: 29 set. 2017.

MICROSOFT. O que é PaaS? Plataforma como serviço. 2017. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-paas/>. Acesso em: 28 ago. 2017.

MILETTO, E. M.; BERTAGNOLLI, S. C. Desenvolvimento de Software II: Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP - Eixo: Informação e Comunicação. Porto Alegre: Bookman, 2014.

MINETTO, E. L. **Framework para desenvolvimento em PHP.** São Paulo:

Novatec, 2007.

MUZETTI, I. Requisitos em projetos ágeis. 2014. Disponível em: <http://www.devmedia.com.br/engenharia-de-requisitos-agil/31871>. Acesso em: 08 ago. 2017.

NADEAU, T. et al. **Projeto e Modelagem de Banco de Dados.** 2. ed. Rio de Janeiro: Elsevier Brasil, 2013.

NESMA. Análise de P ontos de Função Inicial. 2015. Disponível em: <https://nesma.org/wp-content/uploads/2015/11/Early-Function-Point-Analysis-2015-07-15-PT.pdf>. Acesso em: 19 ago. 2017.

NUNES. R. D. A implantação das metodologias ágeis de desenvolvimento de software Scrum e Extreme Programming(XP): Um alternativa para pequenas empresas do setor de tecnologia da informação. **Forscience,** v. 1, n. 1, 5-25, 2013.

ODOO. Cresça o seu negócio: Software de gerenciamento tudo-em-um. Bonito. Fácil de usar. 2017. Disponível em: <https://www.odoo.com/pt_BR/>. Acesso em: 29 set. 2017.

OLIVEIRA, M. S. HTML5. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML/HTML5>. Acesso em: 29 ago. 2017.

ONVLEE, J. Scrum and Function Points: friends or foe. 2015. Disponível em: <https://nesma.org/2015/03/scrum-function-points-friends-foe/>. Acesso em: 04 out. 2017.

PHP. O que é PHP? 2017. Disponível em: <https://secure.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 20 ago. 2017.

PINHEIRO, A. F. **Análise de Ponto de Função (APF).** [s.l]: Amazon, 2017.

POSTGRES. Introdução e Histórico: PostgresSQL Wiki. 2017. Disponível em: <https://wiki.postgresql.org/wiki/Introdução_e_Histórico>. Acesso em: 28 ago.

2017.

POTENCIER, F. PHP é muito melhor do que você pensa. 2014. Disponível em: <https://imasters.com.br/linguagens/php/php-e-muito-melhor-do-que-voce-pensa/?trace=1519021197&source=single>. Acesso em: 27 ago. 2017.

PRESCOTT, P. **Programando em JavaScript.** [S.l]Rio de Janeiro: Babelcube, 2016.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software.** 8. ed. [S.l]: Mcgraw Hill Brasil, 2016.

PRIKLADNICKI, R.; ORTH, A. I. **Planejamento e gerência de projetos.** Porto Alegre: Edipucrs - Puc Rs, 2009.

PYTHON. Python**.** 2017. Disponível em: <https://www.python.org/>. Acesso em: 29 set. 2017.

RIBEIRO, L. O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML. 2012. Disponível em: <http://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 12 ago. 2017.

RITTER, R. Scrum e Planning Poker: Análise de estimativa de software. 2014. Disponível em: <http://www.devmedia.com.br/scrum-e-planning-poker-analise-de-estimativa-de-software/31019>. Acesso em: 12 ago. 2017.

SADALAGE, P. J.; FOWLER, M. **NoSQL Essencial:** Um Guia Conciso para o Mundo Emergente da Persistência Poliglota. São Paulo: Novatec, 2013.

SAMY, M. **Fundamentos de HTML5 e CSS3.** São Paulo: Novatec, 2015.

SATURI. F. Responsabilidades de um Scrum Master. 2013. Disponível em: <http://www.devmedia.com.br/responsabilidades-de-um-scrum-master/27358>. Acesso em: 08 ago. 2017.

SCHMITZ, D. **Criando Sistemas RESTful com PHP e jQuery:** Uma abordagem prática na criação de um sistema de vendas. São Paulo: Novatec, 2013.

SCRUM. The Home of Scrum: Professional Scrum Training, Resources and Certifications to learn and prove your knowledge. 2017. Disponível em: <https://www.scrum.org/>. Acesso em: 27 set. 2017.

SLIM. Documetation: Slim Framework. 2017. Disponível em: <https://www.slimframework.com/docs/>. Acesso em: 29 ago. 2017.

SOMMERVILLE, I. et al. **Engenharia de software.** [s.l]: Addison Wesley Bra, 2008.

SOMMERVILLE, I. et al. **Engenharia de software.** São Paulo:Pearson Education, 2011

SUTHERLAND, J. **Scrum: a arte de fazer o dobro do trabalho na metade do tempo.** São Paulo: Leya, 2014.

SUTHERLAND, J.; SCHWABER, K**.** Um guia definitivo para o Scrum: As regras do jogo. 2013. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 09 ago. 2017.

TAURION, C. **Cloud Computing - Computação em Nuvem:** Transformando o mundo da tecnologia da informação. Rio de Janeiro: Brasport, 2009.

TAVARES, G. Uso de metodologias ágeis em uma organização baseada em linha de produto. 2011. Disponível em: <http://www.devmedia.com.br/uso-de-metodologias-ageis-em-uma-organizacao-baseada-em-linha-de-produto-artigo-revista-engenharia-de-software-magazine-38/21662>. Acesso em: 11 ago. 2017.

VAZQUEZ, C. E.; SIMÕES, G. S. **Engenharia de Requisitos:** software orientado ao negócio. Porto Alegre: Brasport, 2016.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de função:** medição, estimativa e gerenciamento de projeto de software. [s.l]: Erica, 2008.

VERAS, M. **Arquitetura de Nuvem:** Amazon Web Services(AWS). Rio de Janeiro: Brasport, 2013.

VERAS, M. **Computação em Nuvem:** Nova Arquitetura de TI. Rio de Janeiro: Brasport, 2015.

WALTENBERG, R. AngularJS**:** O que é e porquê utilizar. 2016. Disponível em: <http://blog.algaworks.com/o-que-e-angularjs/>. Acesso em: 29 set. 2017.

WAZLAWICK, R. **Engenharia de software:** Conceitos e práticas. [s.l]: Elsevier Brasil, 2013.

ZARDETTO, P. Definição de requisitos em metodologias ágeis. 2016. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/mp258?lang=en. Acesso em: 11 ago. 2016.

ZEMEL, T. **Web design responsivo:** páginas adaptáveis para todos os dispositivos. São Paulo: Casa do Código, 2015.

APÊNDICE A - Questionário de entrevista

QUESTIONÁRIO SOBRE ESTIMATIVA DE SOFTWARE PARA FINS DE APLICAÇÃO DE TÉCNICA DE ESTIMATIVAS COM SCRUM E APF

Esse Apêndice apresenta como foi elaborado o questionário aplicado aos profissionais de empresas que trabalham com metodologias ágeis.

O questionário foi desenvolvido para ser aplicado a empresas e profissionais que adotam metodologias ágeis. O contato com cada empresa ou profissional foi realizado através de rede social e/ou *e-mail*. Responderam as questões profissionais atuantes da cidade de Ponta Grossa / PR e da capital e região metropolitana de São Paulo dando mais confiabilidade a pesquisa.

Como este trabalho propôs uma técnica de estimativa com APF foi necessário elucidar algumas dúvidas. Essas dúvidas foram parcialmente elucidadas com a fundamentação teórica. Para complementar a elucidação dessas dúvidas foram elaboradas questões referentes a estimativas.

Foram elaboradas e aplicadas as seguintes questões:

A. Qual posição você ocupa na sua empresa/equipe?
B. Qual sua experiência na área de desenvolvimento de software?
C. Quem é o responsável por indicar os prazos de entrega das tarefas?
D. Qual metodologia/framework ágil você, empresa ou equipe utiliza?
E. Como é realizada a estimativa de complexidade de uma tarefa?
F. A entregas são realizadas pontualmente? Você está satisfeito com os prazos de entregas da sua empresa/equipe?
G. Você usa ou usaria APF com metodologias ágeis?
H. Qual seu conhecimento sobre APF? Tem certificação em APF? Já utilizou em algum projeto?

A seguir, são apresentadas respostas individuais e a compilação dos resultados gerais por meio de gráficos para cada questão.

A. Qual posição você ocupa na sua empresa/equipe?

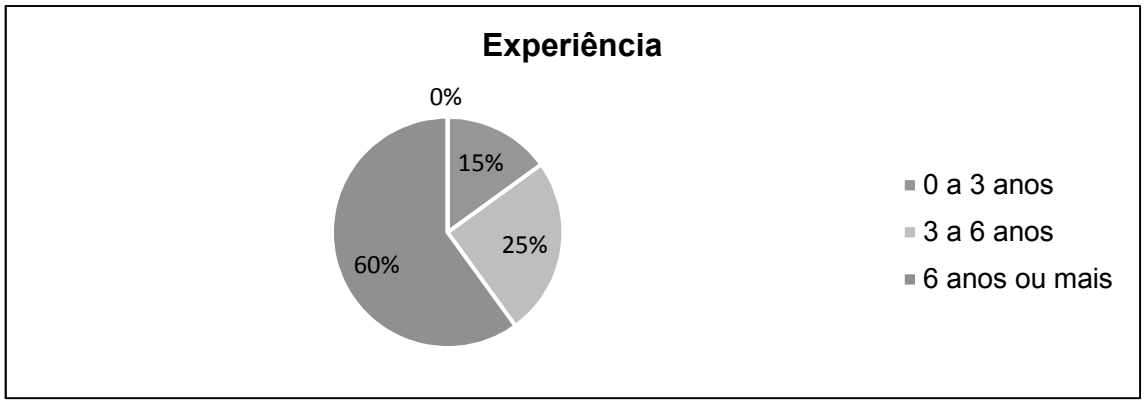
- Entrevistado 1: Analista, Desenvolvedor.
- Entrevistado 2: Analista de Sistemas Pleno.
- Entrevistado 3: Analista De Sistema/Scrum Master/Gerente Do Projeto.

- Entrevistado 4: Desenvolvedor pleno.
- Entrevistado 5: SEO.
- Entrevistado 6: Programador.
- Entrevistado 7: Instrutor, Scrum Master do time que mantém o ERP da empresa e atualmente começando trabalho de agile coach.
- Entrevistado 8: Desenvolvedor.
- Entrevistado 9: Gerente Técnico.
- Entrevistado 10: Gestor de Pessoas.

B. Qual sua experiência na área de desenvolvimento de software?

A Figura 44 apresenta o resumo da experiência dos entrevistados.

Figura 44 – Experiência dos Entrevistados

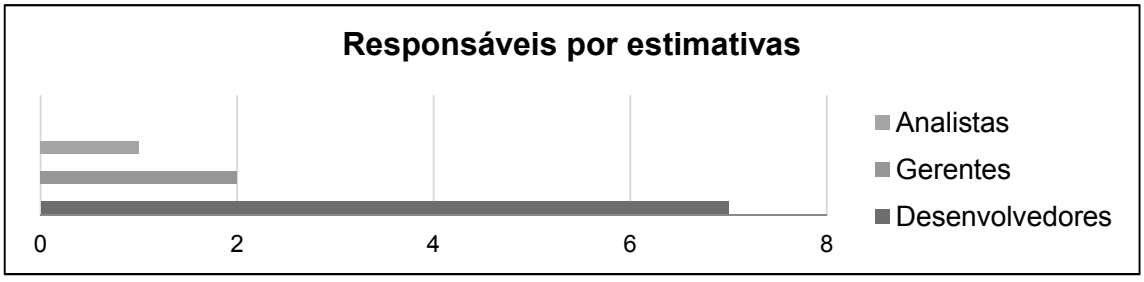


Fonte: Autoria própria

C. Quem é o responsável por indicar os prazos de entrega das tarefas?

A Figura 45 apresenta os responsáveis pelas estimativas em cada organização.

Figura 45 – Responsáveis por estimativas

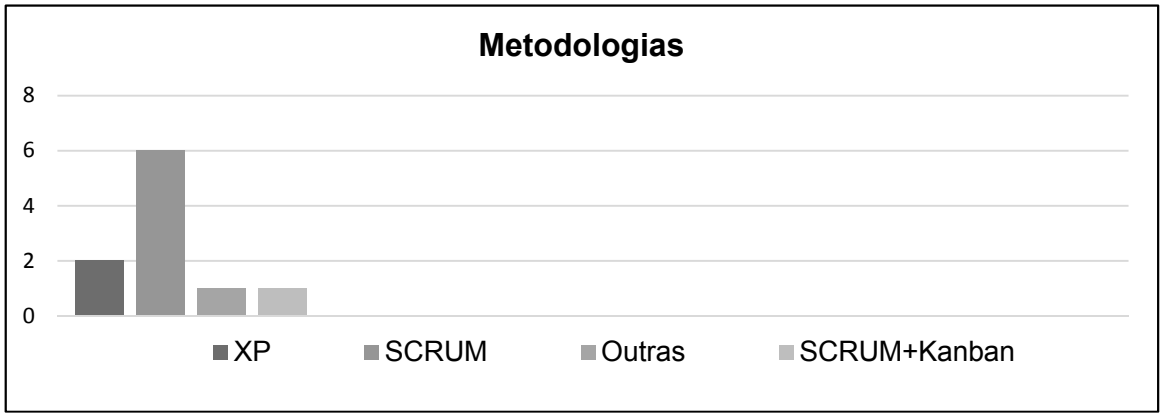


Fonte: Autoria própria

D. Qual metodologia/framework ágil você, empresa ou equipe utiliza?

A Figura 46 apresenta as metodologias utilizadas em cada organização

Figura 46 – Metodologias utilizadas



Fonte: Autoria própria

E. Como é realizada a estimativa de complexidade de uma tarefa?

- Entrevistado 1: Geralmente, o analista deve efetuar a análise de impacto da mudança no sistema. Após isso, ele deve verificar qual desenvolvedor possui o maior conhecimento em cima do assunto que será tratado (ou os desenvolvedores, caso a tarefa abranja mais de um assunto). Após isso, o analista deve verificar a habilidade de programação de cada desenvolvedor para chegar em um prazo concreto.
- Entrevistado 2: baseando-se em comparações. Comparando uma coisa nova com outra complexa que já fizemos.
- Entrevistado 3: Utilizamos planning poker e estimamos em horas.
- Entrevistado 4: No “achismo”.
- Entrevistado 5: Junto a equipe tecnica e gerente de projeto
- Entrevistado 6: Baseada na experiencia de cada um.
- Entrevistado 7: Usamos o *planning poker* para pontuar as histórias
- Entrevistado 8: Após separar em módulos os requisitos fornecidos pelo cliente, estima-se a complexidade de cada módulo individualmente.
- Entrevistado 9: É apresentado para o analista e um desenvolvedor sênior a solicitação feita pelo cliente, com base nisso são levantados quais pontos de alteração dentro do software serão alterados tendo em

vista entender a complexidade da alteração, baseado na experiência dos envolvidos são determinadas as horas necessárias para realizar a tarefa.

- Entrevistado 10: Após ter a análise de sistemas o desenvolvedor realiza uma leitura da demanda e de acordo com sua experiência.

F. A entregas são realizadas pontualmente? Você está satisfeito com os prazos de entregas da sua empresa/equipe?

- Entrevistado 1: Nem sempre são pontuais. Varia muito do humor e do estado mental de cada programador.
- Entrevistado 2: Às vezes os prazos são extrapolados.
- Entrevistado 3: As metas das Sprints não costumam ser atingidas. Quase sempre leva-se mais tempo que o estimado.
- Entrevistado 4: Nunca são entregues no prazo. As tarefas sempre são prometidas prata muito menos que precisam.
- Entrevistado 5: Sim, somos muito conservadores com as estimativas, dando boas margens.
- Entrevistado 6: Geralmente sim.
- Entrevistado 7: Sim, o time faz entregas quase que diárias de valor para o cliente e usuários do sistema, mesmo a Sprint sendo de 2 semanas.
- Entrevistado 8: Sim, o prazo altera somente quando o cliente solicita algo fora do planejado anteriormente.
- Entrevistado 9: As entregas não são realizadas pontualmente, pois é comum o processo sofrer interrupções por conta de agentes externos como alterações de escopo e paradas não programadas como correção de bugs ou sobreposição de tarefas.
- Entrevistado 10: Estou satisfeito, contudo as entregas não são pontuais visto que desenvolvimento de software é depende exclusivamente do capital humano, sendo que temos dias mais criativos, mais produtivos e outros nem tanto.

G. Você usa ou usaria APF com metodologias ágeis?

- Entrevistado 1: Não.

- Entrevistado 2: Acho que não combina muito, principalmente com Scrum. Onde estimamos a complexidade.
- Entrevistado 3: Não, me parece muito mais um método que se aplica para quando se planeja todo software antes de desenvolvê-lo.
- As experiências que tive utilizando em Sprints não foram boas, principalmente quando há alteração em uma história de usuário em alguma Sprint seguinte à que ela foi desenvolvida, contando novamente aquela história.
- Entrevistado 4: Usaria.
- Entrevistado 5: Nunca utilizei, desconheço os processos da metodologia.
- Entrevistado 6: Não uso.
- Entrevistado 7: Não.
- Entrevistado 8: Não.
- Entrevistado 9: Não
- Entrevistado 10: Hoje não usaria, contudo, como pode ver na próxima questão, nosso conhecimento é baixo sobre o assunto. Quem sabe se nos aprofundássemos no assunto poderia ser viável.

H. Qual seu conhecimento sobre APF? Tem certificação em APF? Já utilizou em algum projeto?

- Entrevistado 1: É bem superficial. Não tenho certificação e nunca utilizei em nenhum projeto.
- Entrevistado 2: Tenho conhecimento conceitual, nunca usei fora da área acadêmica.
- Entrevistado 3: Tenho conhecimento sobre o acompanhamento de contagens, em trabalhos de uma fábrica externa. Também conhecimentos da pós-graduação.
- Entrevistado 4: Conhecimento raso. Não. Não.
- Entrevistado 5: Nunca utilizei, desconheço os processos da metodologia
- Entrevistado 6: Nenhum. Não. Não.

- Entrevistado 7: Não sou certificado e já trabalhei um pouco com APF. Porém, não usaria mais por conta de ponto de função ser, no fundo, uma medida baseada em média matemática. O problema que eu vejo em trabalhar com fórmulas baseadas em média é, por exemplo, que a média de 7 e 9 é a mesma que 1 e 15. Ou seja, por conta das mudanças de contexto que ocorrem com frequência no mundo do software, eu noto que ponto de função é algo que não determina com precisão realmente a complexidade do trabalho que será realizado dado que ele tenta traduzir estes diferentes contextos para uma média comum. Além disso, outra questão que noto com ponto de função é que reparei em alguns times que ao usar esta técnica para estimar complexidade eles acabam perdendo o espírito de usar a estimativa durante reuniões de planejamento para promover a discussão e trazer um maior alinhamento entre os desenvolvedores e o cliente sobre o que precisa ser feito.
- Entrevistado 8: Pouco conhecimento, não possuo certificação e não lembro de ter usado.
- Entrevistado 9: Em uma das empresas que trabalhei utilizamos pontos de função, não sei se foi mal implantado, mas quando o utilizamos o custo para finalizar os levantamentos do projeto eram altos e mesmo assim ainda não eram assertivos.
- Entrevistado 10: Conhecimento apenas da Universidade e de leituras, sem nenhuma prática