UNIVERSIDADE FEDERAL DE SANTA CATARINA

JACYARA BOSSE

IMPLEMENTAÇÃO DE MÉTRICAS ÁGEIS INTEGRADAS À PLATAFORMA GITHUB

Florianópolis - SC
2019/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

JACYARA BOSSE

IMPLEMENTAÇÃO DE MÉTRICAS ÁGEIS INTEGRADAS À PLATAFORMA GITHUB

Trabalho de Conclusão de Curso como requisito para a obtenção do grau de Bacharel em Ciências da Computação pela Universidade Federal de Santa Catarina - UFSC.
Orientador: Prof. Dr. Jean Carlo Rossa Hauck

Florianópolis - SC
2019/2

JACYARA BOSSE

IMPLEMENTAÇÃO DE MÉTRICAS ÁGEIS INTEGRADAS À PLATAFORMA GITHUB

Trabalho de Conclusão de Curso como requisito para a obtenção do grau de Bacharel em Ciências da Computação pela Universidade Federal de Santa Catarina - UFSC.
Orientador: Prof. Dr. Jean Carlo Rossa Hauck

Florianópolis - SC, 1 de novembro de 2019

BANCA EXAMINADORA

Prof. Dr. Jean Carlo Rossa Hauck
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Prof. Dr. Raul Sidnei Wazlawick
UNIVERSIDADE FEDERAL DE SANTA CATARINA - LABORATÓRIO BRIDGE

M.Sc Thaísa Cardoso Lacerda
UNIVERSIDADE FEDERAL DE SANTA CATARINA - LABORATÓRIO BRIDGE

AGRADECIMENTOS

Ao meu orientador Jean Hauck, por estar sempre presente durante todo o desenvolvimento deste trabalho, sempre disponível para dúvidas e me guiando para seguir o melhor caminho.

Aos membros da banca, Professor Raul Wazlawick, e à colega de trabalho Thaísa Lacerda, por aceitarem fazer parte da banca avaliadora.

À meus pais, que sempre me deram todo apoio, amor e carinho durante a jornada do curso e principalmente durante a concepção deste trabalho. E também às minhas irmãs sempre apoiando minhas decisões e dando todo suporte necessário para que eu pudesse concluir o curso.

À minha equipe Supernova, que sempre me apoiou no desenvolvimento deste trabalho, em especial a meus colegas Lucas Petroski e Nathan Molinari por toda a ajuda e conhecimento que me passaram.

Ao meu colega, amigo e mentor Antônio Taha por todo o tempo investido me repassando conhecimento e me auxiliando durante o desenvolvimento da ferramenta.

A todos meus amigos amados e às minhas amigas da vida musical por sempre acreditarem em mim e no meu potencial.

A todos meus colegas do curso por sempre me ajudarem e sempre compartilharem conhecimento. E em especial ao meu colega e amigo André de Carvalho Ribeiro (*in memoriam),* primeira pessoa que me mostrou o curso de Ciência de Computação. Que seu legado fique para sempre em nossos corações.

RESUMO

Com o crescimento do mercado de software nos últimos anos, as organizações buscam cada vez mais um alto padrão de qualidade para seus produtos, permitindo que se consolidem no mercado. Isso implica constantemente em buscar a melhoria dos processos de software dentro da organização, muitas vezes no contexto de equipes que utilizam abordagens ágeis. A gestão desses processos de forma manual pode ser muitas vezes cara, demorada e ineficaz. A coleta de métricas de forma automatizada pode agilizar na gestão. Equipes ágeis costumam utilizar plataformas para auxiliar o gerenciamento de suas tarefas, e neste trabalho é abordada a plataforma GitHub, que além de ser conhecida pelo armazenamento e versionamento de código, também possui ferramentas para realizar o gerenciamento de tarefas. Portanto, neste trabalho é desenvolvida uma ferramenta Web que coleta métricas ágeis, integrada à plataforma GitHub, com objetivo de identificação de melhorias no fluxo em equipes em ambientes ágeis. A ferramenta é avaliada no contexto de equipes ágeis, e com os resultados obtidos levantam indícios de que a ferramenta fornece indicadores suficientes para a avaliação de software em ambientes de desenvolvimento ágil, e também fornecendo uma automatização de um processo feito muitas vezes de forma manual.

Palavras-chave: Avaliação de processos de Software, Automatização de avaliação de processos, Qualidade de produto, Equipes ágeis, *GitHub*.

ABSTRACT

With the great growth of the software market in recent years, organizations increasingly seek a high quality standard for their products, allowing the organization to grow and consolidate in the market. This constantly involves pursuing the improvement of software processes within the organization, often in the context of teams that use agile approaches. Managing these manual processes can often be costly, time consuming and ineffective. Automated metric collection can streamline management. The devices often use platforms to assist task management, and this work is covered in the GitHub platform, which is not only known for storing and versioning code, but also has tools to perform or manage tasks. Therefore, this work develops a web tool that collects agile metrics, integrated with the GitHub platform, aiming to identify improvements in the flow of teams in agile environments. The tool is evaluated in the context of agile teams, and with the results obtained, evidence suggests that a tool uses recommended indicators for software evaluation in agile development environments, and also providing automation of a process often done manually.

Keywords: *Process assessment, Software process, Automated process assessment, product quality, GitHub, Agile teams.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Graves problemas com a qualidade de software foram inicialmente percebidos na década de 1960 com o desenvolvimento dos primeiros grandes sistemas de software e continuaram a desafiar a engenharia de software ao longo do século XX. Na chamada crise do software, o software entregue era lento e pouco confiável, difícil de manter e de reusar (Sommerville, 2011). Com motivação nesses problemas, a Engenharia de Software surge no final dos anos 60 com o objetivo de contornar esses diversos problemas, buscando um tratamento mais controlado ao desenvolvimento de sistemas complexos e propondo soluções como a criação de processos de software, com a intenção de gerar produtos finais com qualidade.

A avaliação de processos de software é um procedimento de medição subjetivo que envolve o julgamento de pessoas qualificadas para identificação quantitativa de pontos fortes e fracos nos processos (Anacleto, 2005). Essa avaliação geralmente ocorre com o objetivo de se conhecer a situação atual dos processos executados pela organização (Anacleto, 2005).

A qualidade de um processo, como uma entidade intangível, é tipicamente avaliada com base nos produtos gerados pela sua execução. Essa avaliação normalmente parte da análise das características dos artefatos gerados em relação a um conjunto de resultados esperados de um modelo de referência (ACUÑA, 2000).

A partir dos anos 90, surgiram as chamadas metodologias ágeis para desenvolvimento de software (Manifesto Ágil, 2001). Além de uma nova visão sobre como desenvolver software, ser ágil está associado a uma mudança cultural, uma nova forma de pensar, e não apenas mais uma abordagem, modelo, método ou processo de desenvolvimento. Ser ágil é assumir um conjunto de valores, princípios e práticas que flexibilizam o desenvolvimento de software de forma a responder às constantes mudanças do mercado, enfocando o que é importante (Larman, 2003). Este trabalho é aplicado em equipes que promovem essa cultura ágil, ou seja, as equipes ágeis.

A medição tem como principal foco apoiar a tomada de decisão em relação aos projetos, processos e atendimento aos objetivos organizacionais. Segundo a ISO/IEC 12207 [ISO/IEC, 2008] o propósito da Medição é coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar o efetivo gerenciamento dos processos e demonstrar objetivamente a qualidade dos produtos.

Segundo McGarry (2002), a medição é o nível mais importante em um projeto. A medição de software ajuda o gerente de projetos a fazer um trabalho melhor. Ajuda a definir e implementar planos mais realistas, alocar adequadamente recursos escassos para implementar esses planos e monitorar com precisão o progresso e o desempenho em relação a esses planos.

Segundo a ISO/IEE 15939 (2017), Uma medida é definida em termos de um atributo e o método para quantificá-lo. Uma medida base é funcionalmente independente de outras medidas, que captura informações sobre um único atributo. E a medida derivada é uma medida definida como uma função de dois ou mais valores de medidas básicas. Derivado as medidas capturam informações sobre mais de um atributo ou o mesmo atributo de várias entidades.

Segundo Hartman e Dymond (2006), a métrica ágil deve reforçar princípios ágeis, medir resultados e não saídas, seguir tendências e não números, responder uma pergunta específica para uma pessoa real e pertencer a um conjunto pequeno de métricas e diagnósticos.

Diversas equipes em ambientes de desenvolvimento ágil utilizam o método Kanban. Este método é uma abordagem enxuta para o desenvolvimento de software ágil. Na realidade, o Kanban possui uma série de significados. Literalmente, é uma palavra japonesa que significa "cartão visual". Portanto, no contexto de desenvolvimento de software ele é relativamente novo, enquanto no processo produtivo o termo já é utilizado há mais de meio século (Albino, 2017).

Conforme Boeg (2010), a maioria dos especialistas concorda que o Kanban é um método de gestão de mudanças, dando ênfase em alguns princípios, como visualizar o trabalho em andamento, limitar o trabalho em progresso e tornar explícitas as políticas sendo seguidas.

Algumas organizações vêm implantando a OKR para atingir objetivos organizacionais. OKR (do inglês, *Objective and Key-Results*), é um framework de definição e gerenciamento de objetivos. O framework tem dois componentes básicos: o objetivo (o que queremos alcançar) e um conjunto de resultados-chaves (como saber se estamos chegando lá) (Castro, 2015). O propósito do OKR é criar alinhamento na organização. Assim, objetivos ambiciosos e resultados-chaves mensuráveis são estabelecidos e precisam ser visíveis para todos os níveis da organização (Niven e Lamorte, 2016).

Muitas equipes utilizam plataformas para gerenciamento de código e de tarefas. Uma ferramenta que permite não só o gerenciamento de código mas também o de tarefas é a plataforma GitHub[1], onde os usuários são desenvolvedores que podem criar/contribuir/compartilhar/buscar por repositórios de projetos de acordo com assuntos e linguagens de programação em que foram desenvolvidos (Rocha, 2016).

A plataforma GitHub (*GitHub*, 2019) começou a operar em 2007, quando foi realizado seu primeiro “commit”. Hoje em dia são mais de 40 milhões de usuários e mais de 100 milhões de repositórios, conforme estatísticas de outubro de 2019 no “sobre” do site do GitHub (*GitHub*, 2019).

[1] https://github.com/about

Mesmo com as facilidades de gerenciamento de tarefas oferecidas pelas ferramentas e técnicas disponíveis, muitas equipes ainda possuem dificuldades em estimar tarefas, tempo de entregas e acompanhar o esforço aplicado.

Assim, este trabalho visa a automação parcial do gerenciamento de processos utilizando a API do GitHub para realizar a integração de uma ferramenta com a plataforma. A partir da coleta de dados de tarefas, versões, “pull requests”, e outras funções do GitHub, essa otimização do processo de avaliação traz benefícios para as equipes ágeis que utilizam a plataforma GitHub.

Com isso, espera-se que a partir da coleta de tais dados aplicados a necessidade de cada equipe ágil, seja possível gerar indicadores e resultados que auxiliarão as equipes em vários quesitos de melhoria de processos e estimativa de desenvolvimento de tarefas.

1.1 Objetivos

A partir da contextualização do problema, o objetivo geral e objetivos específicos deste trabalho são assim definidos:

1.1.1 Objetivo geral

O objetivo deste trabalho de conclusão de curso é a análise e implementação de métricas ágeis utilizadas no gerenciamento de projetos por equipes ágeis, coletadas de forma automatizada e integrada à plataforma GitHub.

O trabalho realizado é aplicado em estudo de caso aplicado em equipes ágeis do laboratório Bridge. Essas equipes utilizam o método Kanban nos seus processos internos de desenvolvimento, incluindo a estimativa de entregas e particionamento de tarefas.

O preenchimento dos dados em uma planilha do Kanban utilizada como referência nas equipes, é preenchida de forma manual. É observando o estados das tarefas no quadro da equipe na plataforma GitHub, e então passado para a planilha, semanalmente. A ferramenta a ser implementada tem como objetivo automatizar esse processo, obtendo um histórico consultando as informações obtidas na plataforma GitHub, de acordo com cada equipe.

A implementação de uma ferramenta que busque dados da plataforma GitHub de forma automatizada, irá permitir então uma melhor avaliação dos processos de software nos ambientes ágeis do laboratório. Esta pesquisa busca por meio do desenvolvimento de um sistema web, viabilizar a melhoria na visualização dos processos e fluxos da equipe.

Por meio da integração com o *GitHub,* pretende-se gerar indicadores que servirão como uma entrada para estruturas de obtenção de resultados. O laboratório além do Kanban, utiliza a OKR como estrutura para o alcance de objetivos organizacionais. Portanto, além da inclusão de dados para geração de métricas do Kanban, de forma interna da equipe, busca-se também alimentar o

acompanhamento de *key results* que servem como análise de entrada dos objetivos organizacionais do laboratório.

.

1.1.2 Objetivos específicos

- Análise das métricas ágeis utilizadas por equipes em ambientes de desenvolvimento ágil.
- Análise e modelagem de uma ferramenta com base na análise de métricas significativas para equipes ágeis.
- Desenvolvimento da Ferramenta Web com integração à plataforma GitHub.
- Avaliação da ferramenta em um estudo de caso com equipes ágeis na organização alvo.

1.2 Método de pesquisa

Visando alcançar os objetivos do trabalho, foi adotada uma metodologia de pesquisa dividida em quatro etapas:

Etapa 1: Fundamentação teórica - Análise da literatura sobre os principais conceitos relacionados ao tema deste trabalho. Essa etapa é composta pelas seguintes atividades:

Atividade 1.1: Análise da literatura sobre conceitos de engenharia de software relevantes ao trabalho.

Atividade 1.2: Análise da literatura sobre o método Kanban e Objective and Key Results.

Atividade 1.3: Análise da literatura sobre os conceitos utilizados na plataforma GitHub.

Etapa 2: Levantamento e análise do estado da arte referente a ferramentas de avaliação automatizada de processos de software, utilizando a técnica de mapeamento sistemático da literatura (Kitchenham, 2007). As atividades que compõem esta etapa são as seguintes:

Atividade 2.1: Definição dos termos e estrutura das buscas.

Atividade 2.2: Execução das buscas e seleção de trabalhos relevantes.

Atividade 2.3: Análise dos trabalhos selecionados.

Etapa 3: Modelagem e desenvolvimento da ferramenta, levando em consideração os estudos e entrevistas realizados com o propósito de levantar requisitos que abrangem as métricas ágeis utilizadas pelas equipes. Os seguintes passos são seguidos para conclusão desta etapa:

Atividade 3.1: Análise de requisitos.

Atividade 3.2: Modelagem da ferramenta proposta.
Atividade 3.3: Desenvolvimento da ferramenta proposta.

Etapa 4: Aplicação e avaliação da ferramenta desenvolvida. Nesta etapa, avalia-se a utilidade e a funcionalidade da ferramenta desenvolvida. Esta etapa é composta das seguintes atividades:

Atividade 4.1: Definir objetivo da avaliação.
Atividade 4.2: Planejar a avaliação.
Atividade 4.3: Executar a avaliação.
Atividade 4.4: Análise dos resultados.

1.3 Estrutura do Trabalho

O trabalho está estruturado da seguinte forma. O capítulo 2 apresenta os conceitos e informações que fundamentam este trabalho. O capítulo 3 apresenta o estado da arte, apresentando o estudo sobre as ferramentas e métodos existentes de avaliação automatizada de processos de software. No capítulo 4 é feito o levantamento de requisitos e concepção da modelagem da ferramenta. O capítulo 5 apresenta toda descrição do desenvolvimento da ferramenta. No capítulo 6 é realizada a avaliação da ferramenta e segue-se com a conclusão e trabalhos futuros no capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos e abordagens utilizadas ao longo deste trabalho. Nesta etapa é realizada a revisão bibliográfica com o enfoque nos conceitos de Engenharia de Software aplicados na avaliação de processos de software.

Além dos assuntos envoltos pela Engenharia de Software, é realizado também um estudo também sobre as estruturas e plataformas relacionadas ao trabalho, como a estrutura Objective and Keys Result (OKR) e a plataforma GitHub.

2.1 Processo de Software

A ISO/IEC TR 29110-1:2011 (ISO/IEC, 2011) introduz o conceito de processo como conjunto de atividades inter-relacionadas ou interativas que transformam entradas em saídas. Desta forma, de acordo com Pressman (2010) o processo de software pode ser definido como uma coleção de padrões que definem um conjunto de atividades, ações, tarefas de trabalho, produtos de trabalho e/ou comportamentos relacionados necessários ao desenvolvimento de softwares de computador.

Muitas empresas adotam modelos de processos com o intuito de definir regras de forma geral sobre seus processos. Um modelo de processo apresenta uma filosofia, uma forma geral de comportamento, baseada na qual processos específicos podem ser definidos. Um modelo de processo para as atividades de projeto e desenvolvimento de software também pode ser chamado de ciclo de vida. Assim, quando uma empresa decide adotar um processo, ela deve inicialmente buscar um modelo de processo e adaptar a filosofia e práticas recomendadas para criar seu próprio processo. A partir daí, todos os projetos da empresa deverão seguir este processo definido (Wazlawick, 2013).

A definição e atuação dos processos de software dentro de uma empresa tem um papel fundamental para conseguir a garantia de qualidade do produto desenvolvido. Segundo Pressman (2010, p.25), “Pela combinação de padrões, uma equipe de software pode construir um processo que melhor satisfaça às necessidades de um projeto”. Então o desenvolvimento da melhoria de processos é um fator importante para obter esses objetivos alinhados à uma maior produtividade e controle.

Inclusive, de acordo com Wazlawick (2013), há várias vantagens em se definir desenvolvimento de software como um processo, entre as principais estão: O tempo de treinamento pode ser reduzido, os produtos podem ser mais uniformizados e a possibilidade de capitalizar experiências.

E como este trabalho irá abordar processos de software estabelecidos de forma independente por equipes ágeis, é importante ressaltar que, assim como aponta Sommerville (2011, p.495), “Não existe algo como um processo de software ‘ideal’ ou ‘padrão’ que seja aplicável a todas as organizações ou para todos os produtos de software de um tipo particular”.

2.2 Abordagens ágeis

Nos dias de hoje, as empresas operam em um ambiente global, com mudanças rápidas. Assim, precisam responder a novas oportunidades e novos mercados, a mudanças nas condições econômicas e ao surgimento de produtos e serviços concorrentes. Softwares fazem parte de quase todas as operações de negócios, assim, novos softwares são desenvolvidos rapidamente para obterem proveito de novas oportunidades e responder às pressões competitivas (Sommerville, 2011).

Ainda neste âmbito, de acordo com Sommerville (2011), processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida, projetar, construir e testar o sistema não estão adaptados ao desenvolvimento rápido de software. Com as mudanças nos requisitos ou a descoberta de problemas de requisitos, o projeto do sistema ou sua implementação precisa ser refeito ou retestado. Como consequência, um processo convencional em cascata ou baseado em especificações costuma ser demorado, e o software final é entregue ao cliente bem depois do prazo acordado.

Os modelos ágeis de desenvolvimento de software seguem uma filosofia diferente dos modelos prescritivos. Apesar de serem usualmente mais leves, é errado entender os métodos ágeis como modelos de processos meramente menos complexos ou simplistas. Não se trata de apenas de simplicidade, mas de focar mais nos resultados do que no processo (Wazlawick, 2013).

Os princípios de desenvolvimento de software ágil que são seguidos e defendidos surgiram dos princípios tradicionais de desenvolvimento de software e de várias experiências baseadas nos sucessos e fracassos dos projetos de software (Rao, 2011).

Conforme o Manifesto Ágil (Agile, 2001), O movimento da metodologia ágil não é anti-metodologia; Na verdade, é a restauração da credibilidade da palavra. Também é a restauração de um equilíbrio: adotar a modelagem, mas não apenas para arquivar alguns diagramas em um repositório corporativo empoeirado.

Aprofundando mais sobre o assunto, o propósito do Manifesto Ágil (2001) é o seguinte:

"Estamos descobrindo maneiras melhores de desenvolver software fazendo isso e ajudando os outros a fazer isso. Valorizamos:

- Indivíduos e interações sobre processos e ferramentas.

- Software funcionando sobre documentação completa.
- Colaboração do cliente sobre negociação de contrato.
- Responder às mudanças sobre seguir um plano.”

Portanto, este trabalho irá focar em equipes ágeis, ou seja, equipes de desenvolvimento de software que fazem uso das abordagens ágeis no dia a dia. Essas equipes utilizam princípios da metodologia ágil *Scrum*.

2.2.1 Scrum

Segundo Schwaber (2002, p.2), a metodologia *Scrum* tem como objetivo definir um processo para gerência de projetos de software, que seja focado nas pessoas e que seja indicado para ambientes em que os requisitos surgem e mudam rapidamente.

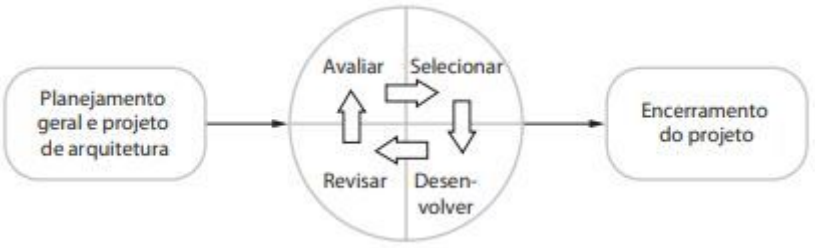
Antes de entrar em detalhes sobre a metodologia Scrum, é necessário definir alguns conceitos como os de *sprint*, *product backlog* e *Scrum teams* (equipes).

Portanto, conforme Schwaber (2002), *sprints* são divisões do desenvolvimento em intervalos de tempos de, no máximo 30 dias. As equipes são pequenas de, no máximo, 7 integrantes. E *product backlog* é uma lista de atividades que provavelmente serão desenvolvidas durante o projeto.

A ideia por trás do Scrum é que toda a equipe deve ter poderes para tomar decisões, de modo que o termo ‘gerente de projeto’ tem sido deliberadamente evitado. Toda a equipe participa das reuniões diárias para a manutenção do foco da equipe. Durante a reunião, todos os membros da equipe compartilham informações, descrevem seu progresso desde a última reunião, os problemas que têm surgido e o que está planejado para o dia seguinte. Isso garante que todos na equipe saibam o que está acontecendo e, se surgirem problemas, poderão replanejar o trabalho de curto prazo para lidar com eles (Sommerville, 2011).

Sobre as fases que compõem o ciclo de vida do Scrum, Sommerville (2011) aponta a existência de três fases. A primeira é uma fase de planejamento geral, em que se estabelecem os objetivos gerais do projeto e da arquitetura do software. Em seguida, ocorre uma série de ciclos de sprint, sendo que cada ciclo desenvolve um incremento do sistema. Finalmente, a última fase do projeto encerra o projeto, completa a documentação exigida, como quadros de ajuda do sistema e manuais do usuário, e avalia as lições aprendidas com o projeto.

Figura 1: Ciclo *Sprint*



Fonte: Sommerville, 2011

. Rising e Janoff (2000), que aplicaram a metodologia ágil Scrum em pequenas equipes ágeis, relatam uma série de resultados positivos após algumas iterações. Sendo esses alguns dos pontos destacados:

- O produto se torna uma série de partes gerenciáveis.
- O progresso é feito, mesmo quando os requisitos não são estáveis.
- Tudo é visível para todos.
- A comunicação da equipe melhora e compartilham sucessos ao longo do caminho
- Os clientes vêem a entrega pontual de incrementos e obtêm feedback frequente sobre como o produto realmente funciona,
- Um relacionamento com o cliente se desenvolve, a confiança e o conhecimento cresce, e é criada uma cultura onde todos esperam que o projeto seja bem-sucedido.

2.3 Avaliação de Processos

De acordo com a ISO/IEC/IEEE 24765 (ISO/IEC, 2011) a avaliação é uma ação de aplicar critérios documentados específicos a um módulo de software, pacote ou produto específico com a finalidade de determinar a aceitação ou a liberação do módulo de software, pacote ou produto. Dessa forma a ISO (ISO/IEC, 2011) também determina processo de software como uma avaliação disciplinada dos processos de uma unidade organizacional em relação a um modelo de avaliação de processos. E modelo de avaliação de processos é um modelo adequado para avaliar a capacidade do processo, com base em um ou mais modelos de referência de processo.

No desenvolvimento deste trabalho não será utilizado um modelo de referência para a avaliação de processo. Porém, os resultados coletados da avaliação servirá como a entrada de dados para popular a OKR (Objective and Key Results).

A avaliação geralmente ocorre com o objetivo de se conhecer a situação atual dos processos executados pela organização. Assim, são determinados pontos de maior prioridade a serem melhorados considerando, principalmente, as metas de melhoria da organização e os benefícios que cada melhoria pode introduzir

(Anacleto, 2005). Portanto realizar a avaliação de processos de acordo com o que cada organização necessita é um dos passos deste trabalho.

2.3.1 Qualidade de Processos de Software

Alcançar competitividade pela qualidade, para as empresas de software, implica tanto na melhoria da qualidade dos produtos de software e serviços correlatos, como dos processos de produção e distribuição de software. Desta forma, assim como para outros setores, qualidade é fator crítico de sucesso para a indústria de software (MPS. BR, 2011).

Os termos ‘garantia de qualidade’ e ‘controle de qualidade’ são amplamente usados na indústria manufatureira. A garantia de qualidade (QA, do inglês quality assurance) é a definição de processos e padrões que devem conduzir a produtos de alta qualidade e a introdução de processos de qualidade na fabricação. O controle de qualidade é a aplicação desses processos de qualidade visando eliminar os produtos que não atingiram o nível de qualidade exigido. (Sommerville, 2011). Portanto ao realizar a avaliação automatizada dos processos, será possível obter resultados como o de quanto a equipe se aproxima da garantia de qualidade proposta pelos padrões de processo, e que também podem impactar em decisões do controle de qualidade.

A meta da garantia de qualidade é fornecer à gerência os dados necessários para que fique informada sobre a qualidade do produto, ganhando assim compreensão e confiança de que a qualidade do produto está satisfazendo suas metas (Pressman, 2010).

2.3.2 Avaliação automatizada de Processos de Software

O processo de avaliação automatizada pode ser de forma parcial ou de forma total. Uma ou mais etapas do processo de avaliação podem ser automatizadas por meio de alguma tecnologia. Para isso aprofundaremos nos conceitos de avaliação totalmente automatizada e avaliação parcialmente automatizada.

A ideia de avaliação totalmente automatizada é de que os indicadores de avaliação são integrados no processo de software e a avaliação é feita usando critérios para interpretar os dados de medição. Os critérios podem ser incorporados em uma ferramenta, como um sistema especialista. Atualmente, isso pode funcionar apenas em condições muito limitadas e especiais. Por exemplo, se uma organização tiver um processo estatisticamente estável, a automação de avaliação pode ser considerada usando os limites de controle para o processo como critérios de avaliação (Jarvinen, 2000).

Visto algumas das limitações da avaliação totalmente automatizada, Jervinen (2000) destaca algumas vantagens na avaliação parcialmente automatizada: “A

avaliação parcialmente automatizada parece mais promissora do que tentar automatizar totalmente a avaliação. O uso de medições de software como evidência de apoio para a avaliação da capacidade de software abre novas possibilidades de avaliação. Em primeiro lugar, as interrupções para o pessoal e seu trabalho podem ser reduzidas à medida que mais informações são adquiridas on-line automaticamente. Em segundo lugar, as avaliações podem ser realizadas com mais frequência, pois os dados de medição são coletados continuamente".

Ainda em relação à avaliação parcial, fornecer uma trilha de auditoria é mais fácil, por mais julgamentos são baseados em dados de medição. E o custo da avaliação é reduzido, uma vez que menos pessoas são necessárias para realizar uma avaliação e as próprias avaliações se tornam parte integrante do trabalho. Existem também dificuldades associadas à avaliação parcialmente automatizada. Talvez os maiores problemas sejam que coletar dados de medição é caro e construir a coleta de medições como parte do processo de software é lento (Jervinen, 2000).

De acordo com o que foi visto, a avaliação parcial além de obter melhores efeitos é também um dos pontos chaves deste estudo. Portanto realizar a automatização da avaliação de processos através de alguma técnica, neste caso sendo a ferramenta Web um insumo para obter os indicativos necessários, é o objetivo principal a ser desenvolvido por este trabalho.

2.4 Kanban

Conforme descrito por Anderson (2016), o Kanban é um método de organização e gerenciamento de serviços profissionais. Ele usa conceitos de *Lean*, como limitar o trabalho em andamento para melhorar os resultados. Um sistema Kanban é um meio de equilibrar a demanda de trabalho a ser realizado com a capacidade disponível para iniciar um novo trabalho.

A palavra "Kanban" vem do japonês e significa "Cartão Visual". Uma pesquisa pela palavra no Google retorna mais de 5 milhões de resultados; isto porque a palavra também é utilizada para descrever o sistema que vem sendo utilizado há décadas pela Toyota para visualmente controlar e equilibrar a linha de produção. O termo tem se tornado quase sinônimo da implementação dos princípios *Lean*. Então, embora sistemas kanban sejam conceito relativamente novo em TI, vêm sendo utilizados por mais de 50 anos no sistema de produção Lean na Toyota. (Boeg, 2010).

Segundo Anderson, (2016) o Kanban é sustentado em seis prioridades, sendo elas:

- Visualize o fluxo de trabalho
- Limite o trabalho em progresso
- Meça e gerencie o fluxo de trabalho

- Torne as políticas explícitas
- Desenvolva loops de feedback
- Melhore de forma colaborativa

Boeg (2010) aponta que existem diversas abordagens para o Kanban, mas a maioria dos especialistas concorda que o Kanban é um método de gestão de mudanças, que dá ênfase aos seguintes princípios:

● Visualizar o trabalho em andamento;

● Visualizar cada passo em sua cadeia de valor, do conceito geral até software que se possa lançar;

● Limitar o Trabalho em Progresso (WIP – *Work in Progress*), restringindo o total de trabalho permitido para cada estágio;

● Tornar explícitas as políticas sendo seguidas;

● Medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessas decisões;

● Identificar oportunidades de melhorias, criando uma cultura Kaizen, na qual a melhoria contínua é responsabilidade de todos.

Outros benefícios podem ser observados ao adotar o kanban em uma equipe ágil. Os gargalos do processo ficarão visíveis em tempo real, é fornecido uma maneira de criar software de forma ágil, é útil para situações nas quais há uma alta taxa de incerteza e alta variabilidade, e também aumenta a visibilidade de tudo que está acontecendo (Albino, 2017).

2.4.1 Quadro Kanban

O quadro Kanban é composto de colunas com os estados onde o cartão Kanban está situado. Segundo Peinado (2007), o cartão kanban é responsável pela comunicação e pelo funcionamento de todo o sistema, nele devem estar contidos as informações mínimas para o bom funcionamento da linha de produção. Sendo necessário, ele poderá conter um número maior de informações, desde que sejam importantes para a área específica, onde se pretende implementar o sistema kanban.

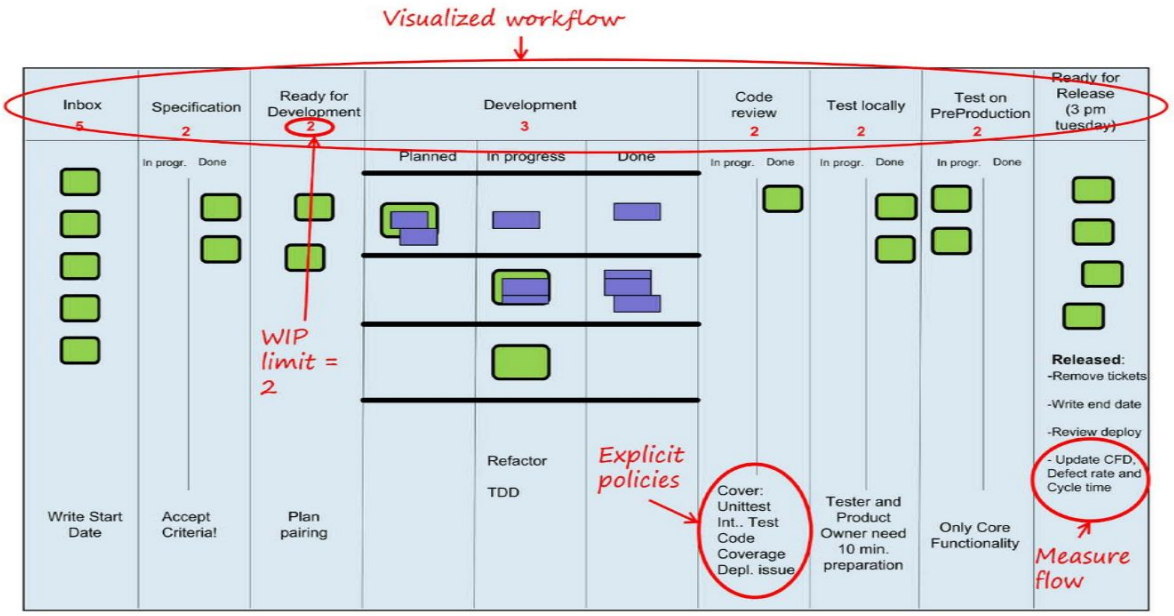
Segundo Boeg (2010), Visualizar seu fluxo de trabalho gera uma série de benefícios, sendo os mais importantes:

● Foco no “todo”: Fica visível exatamente como o seu trabalho afeta as outras pessoas e vice-versa;

● Transparência: Todos sabem exatamente o que está acontecendo e nenhuma informação é escondida;

● Identificação de desperdícios: Surgirá naturalmente um questionamento da razão pela qual as coisas são feitas;

Figura 2: Quadro Kanban



Fonte: Boeg, 2010

2.5 Métricas de software

Segundo Pressman (2010) a mensuração é aplicada no processo de desenvolvimento de software ou atributos de um produto com o objetivo de melhorá-lo de forma contínua que utilizando ao longo do projeto auxilia na estimativa, no controle de qualidade, na avaliação da produtividade e no controle do projeto.

Com as métricas os gerentes conseguem informações quantitativas que auxiliam na tomada de decisões e para obter uma visão melhor do trabalho que está sendo realizado, podendo assim desenvolver um software mais confiável (Inthurn, 2001).

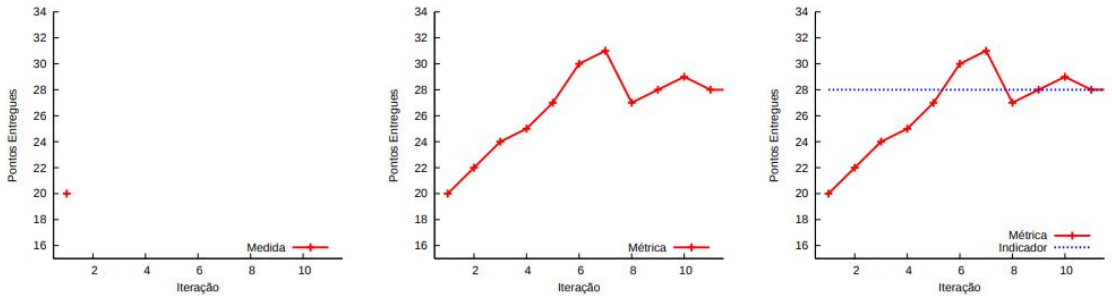
2.5.1 Métricas, medidas e indicadores

Para discutir o papel das métricas no acompanhamento de projetos ágeis, é importante conhecer as diferenças entre o conceitos de medidas, métricas e indicadores.

Uma medida é a avaliação de um atributo segundo um método de medição específico, funcionalmente independente de todas as outras medidas e capturando informação sobre um único atributo (McGarry, 2002).

Conforme a IEEE (IEEE Std 610.12, 1990), uma métrica é um método para determinar se um sistema, componente ou processo possui um certo atributo. Um indicador é um dispositivo ou variável que pode ser configurado para um determinado estado com base no resultado de um processo ou ocorrência de uma determinada condição.

Figura 3: Diferença entre métrica, medida e indicadores



(a) Medida (b) Métrica (c) Indicador

Fonte: Sato, 2007

2.5.2 Métricas ágeis

Com relação às métricas aplicadas com objetivo de medir desempenho em equipes ágeis, Sato (2008) e Cohn (2008) citam uma série de critérios que uma boa métrica ágil deve ter, tais como: reforçar princípios ágeis, envolvendo toda a equipe, seguir tendências e não números; pertencer a um conjunto pequeno de métricas e diagnósticos; ser facilmente coletada, deixar claro os fatores que a influenciam para evitar manipulações, facilitar a melhoria do processo e fornecer *feedbacks*.

Conforme Albino (2017), outra característica importante sobre métricas de negócio é que elas devem ser compreensíveis, isto é, elas não podem ser complicadas e todos devem compreender o que elas representam. A métrica de negócio deve também ser uma relação ou uma taxa. Números absolutos não devem ser usados. A quantidade de usuários pode ser pouco útil, porém, a porcentagem de usuários ativos diários já poderá trazer tendências de crescimento. Razões e taxas são comparativas, o que ajuda a tomar melhores decisões.

Algumas das métricas ágeis também destacadas pelo autor (Albino, 2017), especialmente quando se trata de métricas utilizadas dentro do desenvolvimento do método Kanban, são:

- *Work in Progress*
- Cumulative Flow Diagram
- *Lead time*
- *Throughput*

O *WIP (Work in progress), CFD (Cumulative Flow Diagram), lead time* e *Throughput* são descritos da seção 4.5, junto com outras métricas utilizadas em equipes ágeis.

2.6 Objectives and Key Results (OKR)

Uma das partes finais deste trabalho é a captura dos indicativos obtidos na avaliação do processo. Esses indicativos servirão como entrada para o OKR, pois é uma forma de medir resultados de processos e é utilizado na organização-alvo do estudo de caso deste trabalho.

A *Objectives and Key Results* - OKR, é uma estrutura de pensamento crítico e disciplina contínua que busca garantir que os funcionários trabalhem juntos, concentrando seus esforços para fazer contribuições mensuráveis que impulsionam a empresa para frente (Niven & Lamorte, 2016).

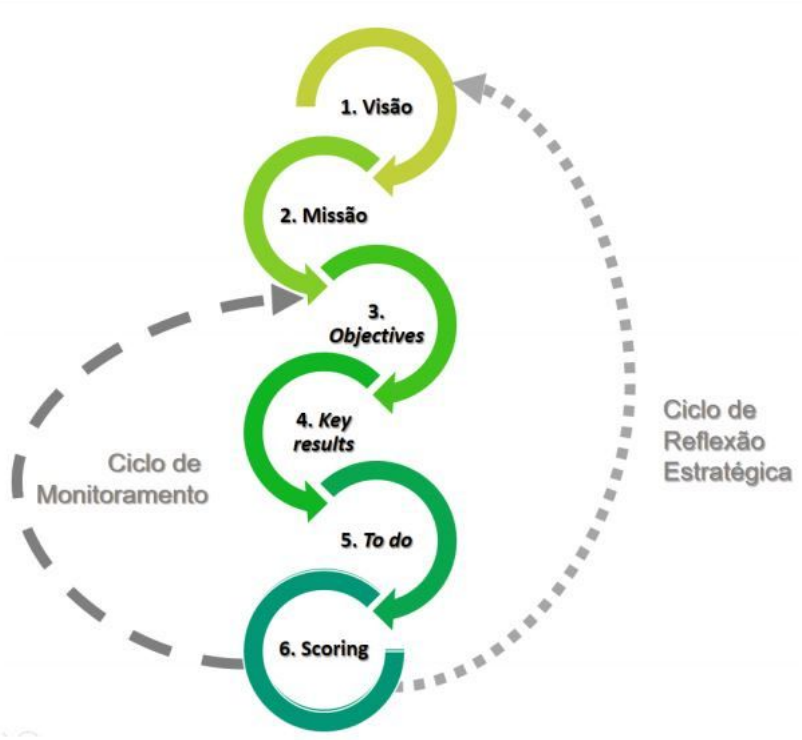
OKR é uma estrutura de definição de metas criada pela Intel e adotada por várias empresas do Vale do Silício. O Google é o caso mais famoso, tendo adotado o OKR em seu primeiro ano. Twitter, LinkedIn, Dropbox e Oracle estão entre outros adotantes (Castro, 2015).

Também segundo Castro (2015), essas são algumas das principais características que tornam a OKR única:

- Simplicidade: para permitir ciclos frequentes de estabelecimento de metas , o processo é extremamente simples. Os próprios OKRs devem ser simples e fáceis de entender.
- Cadência mais curta: Em vez de usar um processo anual de planejamento estático, o OKR usa ciclos mais curtos de definição de metas , permitindo um planejamento dinâmico e uma adaptação mais rápida às mudanças.
- Open Source: OKR é um framework de código aberto para que as empresas o adaptem a cada cultura e contexto, criando diferentes versões dele.
- Objetivos a longo prazo: objetivos que tiram o time da zona de conforto e fazem com que repensem o modo como trabalham para atingir o máximo desempenho.

Conforme aponta Luna (2017), a OKR possui cinco iterações e dois ciclos. A primeira parte da análise em como uma organização “quer ser reconhecida” (visão institucional), permitindo a identificação o “seu propósito” (missão institucional). A partir destas duas informações essenciais, o time procura identificar ao menos três objetivos (objectives), que uma vez alcançados auxiliarão a organização a cumprir sua missão. Para cada objetivo, então, são identificadas de uma a três métricas (key results) que permitirão medir o progresso daquele objetivo. E só então, são elaboradas as ações (to do) para alcançar os resultados descritos pelas métricas estipuladas. A partir daí, os OKRs são distribuídos no tempo e monitorados periodicamente através de reuniões de avaliação (Scorings).

Figura 4: Ciclos da OKR



Fonte: Luna, et al. 2017

Os ciclos de monitoramento ocorrem dentro das iterações de avaliação, visando avaliar como os objetivos estão sendo alcançados e realizando ajustes sempre que necessário. Já os ciclos de reflexão estratégica ocorrem sempre que a estratégia organizacional precisa ser adequada à uma nova realidade (Luna, 2017).

2.7 Plataforma GitHub

Como este trabalho faz o estudo voltado para a integração com a plataforma GitHub, faz-se necessário o apontamento dos principais conceitos envolvidos por esse sistema que aqui serão utilizados.

O GitHub é um site de codificação social que usa o controle de revisão distribuído e sistema de gerenciamento de código fonte. Ele implementa uma rede social na qual os desenvolvedores podem transmitir suas atividades para outros interessados e se inscreverem nela (Thung, 2013).

2.7.1 Repositório

Segundo a especificação no site do GitHub (2018), um repositório é como uma pasta para o seu projeto. O repositório de um projeto contém todos os arquivos do seu projeto e armazena o histórico de revisões de cada arquivo.

Cada usuário pode possuir um repositório individualmente e conceder a outras pessoas o acesso do colaborador ao seu repositório para que elas possam colaborar em seu projeto. E também pode compartilhar a propriedade de um repositório com outras pessoas em uma organização e conceder aos membros da organização permissões de acesso para colaborar em seu repositório.

Repositórios podem ser públicos ou privados. Repositórios públicos são visíveis para todos. Somente o proprietário e os colaboradores podem visualizar ou contribuir para um repositório privado.

2.7.2 Issues

Conforme a documentação do GitHub (2014), issues são uma ótima maneira de acompanhar as tarefas, os aprimoramentos e os bugs dos seus projetos. Eles são como e-mails, exceto que podem ser compartilhados e discutidos com o resto de sua equipe. E o acompanhamento de issues do GitHub tem foco na colaboração, referências e formatação de texto.

Em nossa modelagem, as *issues* referem-se às tarefas que cada equipe irá executar, seja uma pequena parte do módulo a ser desenvolvida, um bug ou uma alteração na análise de requisitos

2.7.3 Milestones, Labels e Assignatures

Depois de coletar muitas *issues* (tarefas), talvez seja difícil encontrar as que mais lhe interessam. *Milestones*, *labels* e *assignatures* são ótimos recursos para filtrar e categorizar as *issues* (GitHub, 2014).

Também segundo a documentação do GitHub (2014) é designado as seguintes definições:

> ***Milestones*** (marcos): são grupos de problemas que correspondem a um projeto, recurso ou período de tempo. As pessoas as usam de muitas maneiras diferentes no desenvolvimento de software.
>
> ***Labels*** (etiquetas): são uma maneira de organizar diferentes tipos de problemas. Os problemas podem ter quantas etiquetas você quiser e você pode filtrar um ou vários rótulos de uma só vez.
>
> ***Assignatures*** (atribuição): Cada *issue* pode ter um responsável - uma pessoa responsável por encaminhar a tarefa.

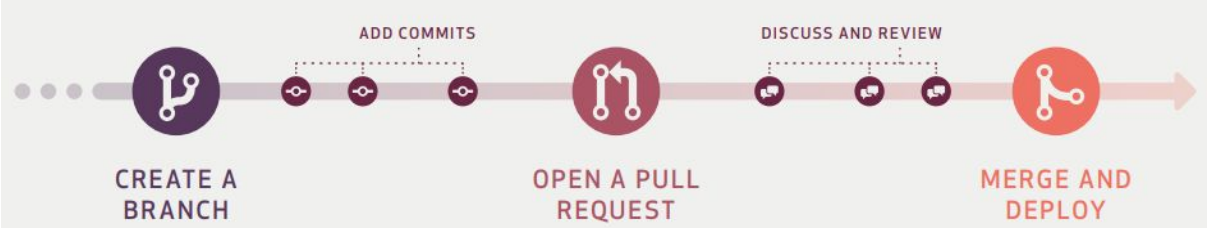
2.7.4 Fluxo

É necessário conhecer os seguintes conceitos explicitados a seguir para entender o fluxo do GitHub. Esses conceitos também serão muito abordados durante todo o caso de estudo desenvolvido neste trabalho.

De acordo com o Glossário do GitHub (2018), *branches*, *pull requests* e *merge* são conceituados conforme o seguinte:

> ***Branch***: Uma *branch* (ramo) é uma versão paralela de um repositório. Ele está contido no repositório, mas não afeta a ramificação principal, permitindo que o usuário trabalhe livremente sem interromper a versão "ao vivo". Depois de fazer as alterações desejadas, o usuário pode mesclar sua ramificação na ramificação principal para publicar suas alterações.
>
> ***Commit***: Um commit, ou "revisão", é uma mudança individual em um arquivo (ou conjunto de arquivos). Permite manter registro de quais alterações foram feitas, quando e por quem. As confirmações geralmente contêm uma mensagem de confirmação, que é uma breve descrição de quais alterações foram feitas.
>
> ***Pull request***: Os *pull request* são alterações propostas para um repositório submetido por um usuário e aceito ou rejeitado pelos colaboradores de um repositório. Assim como as *issues*, *pull requests* têm seu próprio fórum de discussão.
>
> ***Merge***: Significa levar as alterações de um branch e aplicá-las em outro. Isso geralmente acontece através de um pull request (que pode ser considerada como uma solicitação de merge) ou por meio da linha de comando. Um merge (mesclagem) pode ser feito automaticamente através de um *pull request* por meio da interface da web do GitHub, se não houver alterações conflitantes, ou pode ser feita por meio da linha de comando.

Figura 5: *The GitHub flow*



Fonte: GitHub, 2017 (https://guides.github.com/introduction/flow)

3 ESTADO DA ARTE

Esta seção tem o objetivo de levantar informações sobre a avaliação automatizada de processos de Software em organizações que usam metodologias ágeis. Para alcançar este objetivo, é efetuado um mapeamento sistemático da literatura (MSL) em bibliotecas digitais, pois esta técnica fornece uma maneira de encontrar, avaliar e interpretar todas as pesquisas relevantes sobre uma determinada questão de pesquisa (KITCHENHAM, 2007). A técnica de MSL é adaptada às limitações de tempo de um trabalho de conclusão de curso.

Esta etapa do trabalho foi desenvolvida com conjunto com o acadêmico Augusto (Zwirtes, 2018), que também realiza seu trabalho final de curso na área de automatização de avaliação de processos de software.

Esse levantamento tem como foco extrair informações sobre como modelos automatizados vem sendo utilizado na prática para fazer avaliações de processos de software, assim como as conclusões obtidas em sua utilização e experiências adquiridas. A relação dos trabalhos encontrados fornece uma visão do estado da arte nesse assunto.

3.1 Mapeamento Sistemático da Literatura

Um MSL visa buscar respostas a perguntas de pesquisa para basear suas conclusões, por meio de uma busca na literatura realizada por meio de procedimentos sistemáticos. Para a área da Engenharia de Software, um dos principais guias para realização de MSLs é proposto por Kitchenham (2007). Nesse guia é proposta uma série de atividades para a realização de uma MSL, organizadas em três principais fases: planejar a revisão; conduzir a revisão e relatar a revisão. Cada fase é constituída por um conjunto de etapas as quais foram adotadas neste trabalho e são apresentadas de forma agrupada nas seções a seguir: i) identificação das pesquisas relevantes; ii) seleção dos estudos primários; iii) avaliação da qualidade do estudos; iv) extração dos dados; v) síntese dos dados;

3.2 Definição do mapeamento

A partir da necessidade de pesquisa identificada, a questão geral de pesquisa definida foi: “Quais são as abordagens para avaliação automatizada de processos?” . Este trabalho procurou responder a essa questão geral com o objetivo de esclarecer informações quanto ao uso de modelos automatizados utilizados por equipes ágeis. Assim os critérios utilizados para definir a pesquisa são:

- População: processos modelados.
- Intervenção: avaliação automatizada de processos.
- Comparação: não se aplica, pois se trata de um MSL, onde não será realizado meta-analysis.
- Resultado: melhoria de processos.

- Contexto: empresas, organizações, equipes de desenvolvimento de software.

Após a definição da questão geral de pesquisa foram derivados os termos de pesquisa, desenvolvidas as strings de busca e selecionadas as fontes de dados.

3.2.1 Base de dados

As buscas são realizadas em bibliotecas digitais relevantes para a área de Engenharia de Software (KITCHENHAM, 2007), como: IEEEXplore, ScienceDirect, Springer e ACM Digital Library. Estas bases de dados foram escolhidas pois são as mais importantes bibliotecas e bases de dados digitais, onde pode-se encontrar trabalhos científicos relevantes para o desenvolvimento do trabalho de conclusão do curso.

3.2.2 Critérios de pesquisa

O processo de consolidação da busca envolveu o acesso às bibliotecas digitais pré- selecionadas entre julho/2018 e agosto/2018, tendo os resultados sido revisados em agosto/2018. Os critérios de inclusão, exclusão e qualidade dos estudos primários, baseados em Kitchenham (2007), encontram-se detalhados a seguir:

Critérios de inclusão:

- Artigos que apresentam métodos, técnicas ou ferramentas de avaliação automatizada de processos de software.
- Artigos que apresentam estudos sobre auto-avaliação de processos em equipes ágeis.
- Estudos disponíveis em bases de dados científicas.
- Artigos publicados entre o ano de 2010 e 2018.
- Artigos que apresentem estudos empíricos com resultados sobre avaliação automatizada de processos.
- Artigos que apresentem estudos empíricos de ferramentas de auto-avaliação de processos de software.
- Artigos que relatem experiências de avaliação automatizada de processos em ambientes de desenvolvimento ágil de software.

Critérios de exclusão:

- Artigos que não abordam processos de software.
- Artigos nos quais o material completo não está disponível.
- Artigos que não apresentem resultados de avaliação de processos seguindo normas e modelos de referência mais conhecidos.
- Artigos que abordam softwares automatizados e nao processos.
- Estudos que não estejam completamente disponíveis via rede Universidade Federal de Santa Catarina;

3.2.3 Termos de pesquisa

Os termos utilizados para a realização da busca incluem *semi-automate*, *automated software, automated, automation, automatics* por se tratar de modelos automatizados de software, *process evaluation, process assessment, process self assessment, process model, process appraisal, process standard, process audit* por conter informações referentes às avaliações de processos, *agile*, *scrum*, *BPM*, *agile teams*, *software entities*, *VSEs*, *very small entities* a fim de definir os ambientes a serem aplicadas as formas automatizadas de processos.

Abaixo na "Tabela 1" estão especificados os termos selecionados, sinônimos e suas traduções:

Tabela 1: Termos de busca

Critérios	Termos	Sinônimos	Tradução para inglês
População	Processos de software	Auto-avaliação de processos, Ferramenta de autoavaliação, auditoria de processos	Process evaluation, process assessment, process auditing, process assessment tool
Intervenção	Avaliação automatizada, auto avaliação de processos	Ferramenta para avaliação automatizada de processos, Tecnologias de avaliação automatizada de processos, avaliação semi-automatizada de processos	automated process assessment, self-assessment, automated process evaluation, automated process appraisal, semi-automated process
Resultado	Padrão de processo de software	Modelo de processo de software, Padrão de processos, BPM	software process model, process standard, BPM, software process standard
Contexto	Empresas de Software, Equipes ágeis	Micro e pequenas empresas de software, ambientes ágeis, SCRUM	Software entities, VSEs, very small entities, Agile teams, agile environment, SCRUM

Fonte: Elaborado pelos autores

3.2.4 Strings de busca

Abaixo estão dispostos os termos utilizados agrupados, constituindo a *string* genérica de busca a ser adaptada para as pesquisas em bibliotecas digitais:

("automation" OR "automatics" OR "tool" OR "semi-automated" OR "automated software" OR "automated") AND ("process evaluation" OR "process assessment" OR "process audit" OR "process appraisal" OR "process self assessment" OR " process model" OR "process standard") AND ("agile" OR "scrum" OR "BPM" OR "agile teams" OR "software entities" OR "VSEs" OR "very small entities")

E na "Tabela 2" a seguir estão dispostos os termos anteriores, constituindo as *strings* de busca especificamente para cada base digital de pesquisa:

Tabela 2: String de busca adaptada

Base de dados	String Adaptada
IEEEXplore	((("automation" OR "automatics" OR "tool" OR "semi-automated" OR "semi automated" OR "software" OR "automated") AND ("process evaluation" OR "process assessment" OR "process audit" OR "process appraisal" OR "process self assessment" OR "process model" OR "process standard"))) [Specify Year Range: 2005 - 2018]
ACM Digital Library	"query": { ((("automation" OR "automatics" OR "tool" OR "semi-automated" OR "semi automated" OR "software" OR "automated") AND ("process evaluation" OR "process assessment" OR "process audit" OR "process appraisal" OR "process self assessment" OR "process model" OR "process standard"))) } "filter": {"publication Year":{ "gte":2005, "lte":2018 }}
Science Direct	pub-date > 2009 and (“automation” OR “automatics” OR “tool” OR “semi-automated” OR “automated software” OR “automated”) AND (“process evaluation” OR “process assessment” OR “process audit” OR “process appraisal” OR “process self assessment” OR “ process model” OR “process standard”) AND (“agile” OR “scrum” OR “BPM” OR “agile teams” OR “software entities” OR “VSEs” OR “very small entities”) [All Sources (Computer Science)]
Springer	(automation OR automatics OR tool OR semi-automated OR automated software OR automated AND process evaluation OR process assessment OR process audit OR process appraisal OR process self assessment OR process model OR process standard AND agile OR scrum OR BPM OR agile teams OR software entities OR VSEs OR very small entities) within computer science AND 2010-2018

Fonte: Elaborado pelos autores.

As strings variaram conforme a biblioteca digital utilizada e os termos definidos anteriormente foram adaptados a cada base com o objetivo de retornar os resultados de maior relevância.

3.3 Seleção dos estudos

A seleção dos artigos começou com a extração dos estudos encontrados nas buscas efetuadas nas bibliotecas digitais. Grande parte dos estudos retornados, tendo em vista que em algumas plataformas o número foi superior a mil resultados, foram avaliados com a leitura dos títulos e dos resumos primeiramente. Nas fases de leitura de títulos e resumos foram aplicados os critérios de inclusão e exclusão e os estudos considerados irrelevantes para a questão de pesquisa foram descartados.

Dessa forma, para a seleção dos estudos, foram realizados três ciclos: Busca Inicial, 1ª Iteração, 2ª Iteração e última iteração. A busca inicial extraiu os estudos das bibliotecas digitais pré-selecionadas, enquanto na 1ª, 2ª e última iteração constituíram a eliminação dos artigos que não atendem aos critérios de inclusão ou que entram nos critérios de exclusão e, portanto, não são considerados relevantes para a questão de pesquisa, respectivamente. As etapas serão descritas a seguir.

Na busca inicial do processo de coleta de dados foi submetido a string de busca nas bibliotecas digitais. As bibliotecas tiveram a seguinte quantidade de resultados:

- IEEExplore: 3750 resultados;
- Science Direct: 173 resultados;

- ACM Digital Library: 1435 resultados;
- Springer: 2328 resultados;

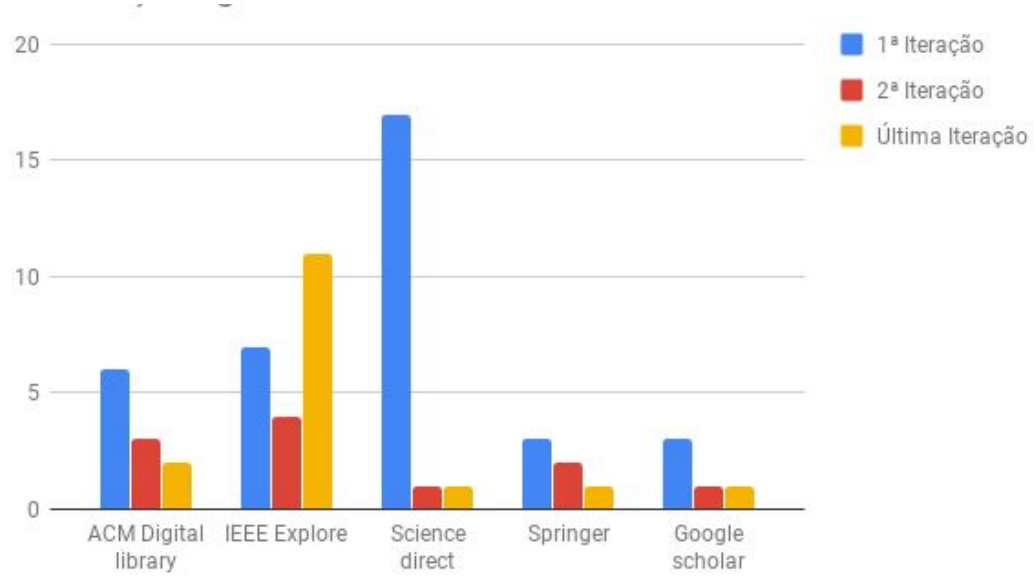
Para os casos em que houve inúmeros resultados, foi feita uma filtragem dos 125 primeiros resultados mais relevantes, conforme ordenação de cada base de dados pesquisada.

Na 1ª Iteração foram realizadas as leituras dos títulos e resumos, aplicando-se novamente os critérios de inclusão e exclusão e também os critérios de qualidade, eliminando outra vez os estudos irrelevantes.

Na 2ª Iteração todos os textos restantes foram lidos integralmente e confrontados com a questão da pesquisa e igualmente com os critérios de inclusão, exclusão e qualidade definidos. Por fim, dos textos considerados relevantes para o estudo, os dados para análises foram extraídos.

Após essas etapas, ainda houve alguns títulos base que não estavam sendo encontrados pelas strings de pesquisa submetidas. Então, houve o reformulamento e melhoria da string utilizada na biblioteca IEEExplore e aumentado o alcance para resultados a partir de 2005, e com esse refinamento, os resultados encontrados diminuíram para 1870. Foram verificados todos os 1870 títulos, realizando a leitura do resumo dos títulos pertinentes, e a posterior leitura integral e coleta dos dados dos estudos incluídos. Foram 7 trabalhos a mais incluídos nesta última iteração. As etapas e resultados podem ser observados na tabela a seguir:

Figura 6: Informações gerais da coleta de dados



Fonte: elaborado pelos autores.

3.4 Extração dos dados

Após a realização da leitura completa de todos os artigos selecionados, foram definidas as informações que possuíam maior relevância para responder à questão da pesquisa, assim com base na pergunta geral de pesquisa e na

necessidade identificada foram selecionados os dados a serem extraídos dos trabalhos.

3.4.1 Abordagens utilizadas

O quesito de abordagens e estratégia tentou extrair informações de maneira a responder sobre as abordagens utilizadas nas avaliações de software juntamente com a descrição da utilização de avaliação automatizada em abordagens ágeis. Cada uma delas é brevemente apresentada a seguir.

Calabró, Lonetti & Marchetti (2015) fazem uso de uma estrutura integrada que permite a modelagem, execução e análise de processos de negócios com base em uma infraestrutura de monitoramento flexível e adaptável, permitindo a definição e avaliação de medidas de KPI (*Key Performance Indexes*) especificadas pelo usuário.

Por sua vez Dai, Bai & Zhao (2007) fazem uma abordagem baseada na verificação de um *workflow* para especificar e verificar as composições de fluxo de trabalho de serviços da Web transformando um modelo BPEL em um modelo TPPN. Já a abordagem de Homchuenchom (2011) utiliza tarefas do Scrum selecionadas e modeladas com BPMN em uma plataforma BPMS. Porém o trabalho não aborda uma avaliação automatizada sobre o modelo feito, mas sim tem como objetivo um modelo proposto para gerar uma compreensão e visão detalhada do processo Scrum. Não tem como alvo automatizar o processo em si, mas tenta-se ajudar a equipe a realizar suas tarefas de maneira mais rápida e fácil.

A abordagem de Adali, Top & Demirors (2017) é *online* e tem a capacidade de orientar e automatizar os processos de avaliação de ambientes ágeis, incluindo fases de planejamento, condução e relatos no processo de avaliação de modo a fornecer uma maneira abrangente de avaliar a agilidade sem depender de um modelo *agile* específico. Agora, tratando-se de Sabato & FIlangieri (2002) a abordagem identifica os problemas operacionais reais, capturando o conhecimento e reduzindo drasticamente a análise de tempo, apoiando o proprietário do processo durante a sua fase de diagnóstico. Os trabalhadores da empresa respondem a um questionário quantitativo, baseado em respostas por números, que depois são passadas a um algoritmo desenvolvido para gerar um resultado qualitativo. Para cada indicador de qualidade, a metodologia propõe um questionário com um conjunto de perguntas, ordenadas por peso, orientadas para identificar a problemática operacional específica no sistema, organização, processo ou na qualidade operativa.

Astorga-Vargas (2014) integra as partes normativas e informativas de NMX-I-059 e NMX-I-15504 para fornecer ao avaliador maiores elementos que apóiam a tomada de decisões no momento de atribuir as qualificações às evidências apresentadas, tornando este processo eficiente, repetível e consistente. Para Pino (2010) a abordagem visa entender o processo de software dentro de uma organização e, usando esta base de conhecimento, se propõe a impulsionar o

implementação de mudanças para que metas específicas possam ser alcançadas, a estratégia é formular e executar as melhorias de corte de forma iterativas e incrementais, iniciadas por oportunidades de melhoria prioritárias que são descritas no plano de melhoria geral.

Quanto a abordagem de Patel & Ramachandran (2009) um formulário fará perguntas sobre as áreas críticas que cercam as práticas ágeis, levando em consideração aspectos críticos: tamanho da equipe, cliente no site, localização da equipe. Enquanto Choi, Kim & Park (2012) utilizam um processo modelo para que o desenvolvimento siga recomendações com base em quatro visões: visão de modelo de avaliação, visão de foco de negócio, visão organizacional e visão de ciclo de vida de software.

Garcia & Pacheco (2009) demonstram que uma pequena organização pode usar uma abordagem para o modelo CMMI-DEV como estrutura para fortalecer as práticas de gerenciamento de projetos ágeis, melhorar o desempenho do projeto e alcançar altos níveis de capacidade CMMI-DEV. E por último a abordagem de Khokhar, Mansoor, Rehman & Raufa (2010) aplica *guidelines* para três áreas: fase implementação, qualidade de processos e satisfação do consumidor visando as melhorias de processo. Enquanto Montenegro & Arévalo (2018), propõe o design e avaliação de um artefato modelo para desenvolvimento de software de governança em equipes de VSE.

Se tratando de Pino, Garcia & Piattini (2007) a abordagem é feita atribuindo valores sobre o cumprimento ou não das práticas específicas do modelo de processo selecionado. Permitindo verificar no nível de prática o estado dos processos de software da empresa em relação às áreas de processo do modelo CMMI. A abordagem de Varkoi (2010) apresenta um método de avaliação que consiste em processo de avaliação juntamente com um modelo de avaliação de processos incluindo indicadores de avaliação.

Suteeca & Ramingwong (2016) se concentram em explicar as relações entre práticas ágeis e um processo de implementação de software definido pela ISO / IEC 29110 versão 2011. Tendo como principal objetivo do estudo mostrar o potencial da aplicação desta norma ao desenvolvimento de software com SCRUM. Já Khonkhar, Mansoor, Rehman & Rauf (2010) a fim de fornecer uma base para o desenvolvimento de qualidade de software para pequenas organizações de software, propõe uma abordagem personalizada para a medição e melhoria de processos, levando em conta os limites explícitos dessas pequena organizações. O modelo proposto então fornece uma abordagem de monitoramento contínuo para processos de software. Essa abordagem foi implementada no setores de elicitação de requisitos, garantia de qualidade do software no ciclo de vida de desenvolvimento, e por último no processo de satisfação do cliente.

Para análise das atividades realizadas nas abordagens de avaliação, foi utilizada como referência a INE/IEC 15504-3 (atualizada para ISO/IEC 330020), de forma a generalizar as atividades e fases utilizadas nas abordagens.

Tabela 3: Tabela de referência de atividades ISO/IEC 330020

Abordagem	Planejamento	Coleta de dados	Validação de dados	Pontuação/ avaliação dos atributos de processo	Relatório dos resultados
(Calabró, Lonetti & Marchetti, 2015)	-	+	+/-	+	-
(Khokhar, Mansoor, Rehman & Raufa, 2010)	+	+	+	+	+
(Garcia & Pacheco, 2009)	+	-	-	+	+
(Choi, Kim & Park, 2012)	+	+	+	+	+
(Patel & Ramachandran, 2009)	+	+	+/-	+	+
(Pino, 2010)	+	+	+	+	+
(Astorga-Vargas, 2014)	+	+/-	+/-	+	-
(Sabato & FIlangieri, 2002)	+	-	-	+	-
(Adali, Top & Demirors, 2017)	+	+	+	+	+
(Homchuenchom, 2011)	+	-	-	+	-
(Dai, Bai & Zhao, 2007)	+	-	-	-	-
(Pino, Garcia & Piattini, 2007)	+	+/-	+/-	+	+
(Zaouali & GhannouchiI, 2016)	+	+/-	+/-	+/-	-
(Varkoi, 2010)	+	+	+	+	+
(Suteeca & Ramingwong, 2011)	+	-	-	-	-
(Montenegro & Arévalo, 2018)	+	-	-	+/-	+

Legenda: **+** Completamente | **+/-** Parcialmente | **-** Não realiza

Fonte: Elaborado pelos autores

Dentre as abordagens apresentadas na tabela acima, constatou-se que menos da metade (31,2%) apresentam todas as atividades, o que implica que muitas abordagens ainda não seguem claramente as etapas para avaliações de processos relacionadas a INE/IEC 15504-3. Entre as 5 atividades que compõem essa avaliação de processo, a parte de Relatório dos resultados foi a que mais teve

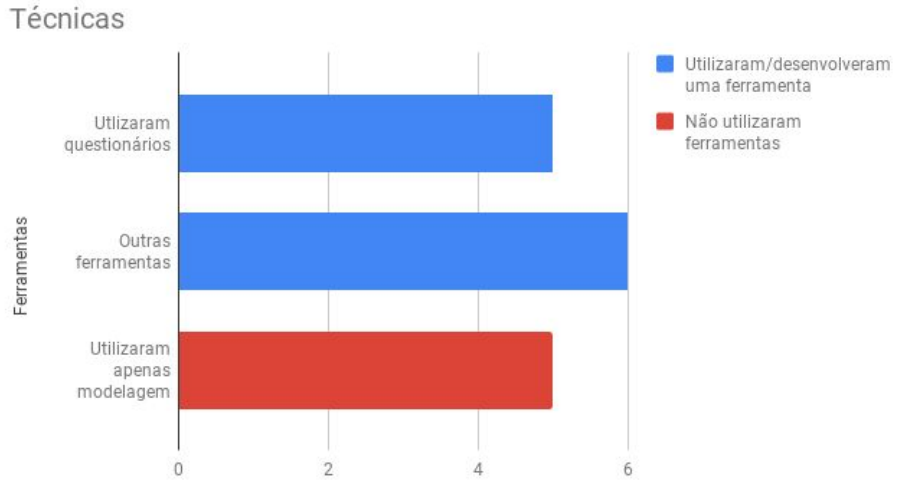
abstenção (43,7%) de uso provando que muitas abordagens acabam por não usar essa atividade como forma de gerar um desenvolvimento para um plano de melhoria ou determinar capacidade e riscos associados. Entretanto, as atividades referentes a Coleta de dados e Validação de dados apresentam ambas 37,5 % de ausência nas abordagens. Quando validação não puder ser alcançada, a circunstância deve ser claramente indicada na saída da avaliação do processo juntamente com o risco de perda dos resultados.

Em contrapartida, as atividades de Planejamento e Avaliação dos atributos de processo estão presentes respectivamente em 93,7% e 75% das abordagens o que mostra uma preocupação grande em relação a fase inicial de avaliação e fase onde espera-se que o avaliador consiga classificar o propósito da avaliação e seu contexto.

3.4.2 Ferramentas

Dentre os estudos coletados, a maior parte fez uso de alguma ferramenta para avaliação de processos de *software*. Outras, no entanto, propuseram em seus estudos apenas uma modelagem para avaliação e/ou melhoria nos processos. Algumas delas aplicaram essas ferramentas como será descrito na seção 3.4.5. Podemos destacar a quantidade de estudos que utilizaram ou não ferramentas no gráfico a seguir:

Figura 7: Técnicas e ferramentas



Fonte: elaborado pelos autores.

Dentre um dos estudos que utiliza questionário, é interessante destacar o ASSO (A web-portal application for operational process assessment and organizational improvement) (SABATO, Sergio; FILANGIERI, Carlo, 2002). Nesta ferramenta os funcionários da empresa submetem-se à um questionário eletrônico, e os resultados identificam os problemas operacionais reais, capturando o

conhecimento e reduzindo o tempo de análise. No geral, a maior parte dos trabalhos que utilizam questionários se baseiam nos modelos de referência SCRUM e/ou CMMI. Então, a partir dos dados coletados nas perguntas, formulam indicativos de aceitação e qualidade de processos de software, como é o caso da ferramenta SPIALS (Software Process Improvement Adaptive Learning System)(HOMCHUENCHOM, Disorn, 2011).

Além de questionários, alguns estudos utilizaram ou desenvolveram ferramentas mais completas para auxiliar na avaliação de processos de software. Uma dessas ferramentas é o SysProVal (GARCIA, Ivan, 2009), que permite comparar as práticas atuais com práticas adaptadas ao CMMI, realizando a avaliação do processo selecionado e indicando uma melhoria adequada ao plano. Algumas são ferramentas web e foram desenvolvidas para a avaliação on-line, como a ferramenta de avaliação de agilidade AssessAgility (ADALI, Onat Ege, 2017). Alguns trabalhos não desenvolveram uma ferramenta específica, porém utilizaram uma ferramenta já existente, como é o caso da ferramenta WofBPEL([DAI, Guilan, 2007).

Apesar de nem todos os trabalhos apresentarem uma ferramenta para automatizar a avaliação de processos de softwares, todos apresentaram um modelo para seus casos de estudo. De um modo abrangente, todos os modelos apresentam as etapas planejamento e gerenciamento do projeto, análise dos dados e formulação de melhorias ou recomendações. Um dos modelos que destaca bem todas essas etapas é o Agile Maturity Model (AMM) (PATEL, Chetankumar, 2009).

Tabela 4: Relação de estudos coletados no mapeamento

Estudo	Ferramentas
(Calabró, Lonetti & Marchetti, 2015)	Key performance Indicators (KPIs)
(Dai, Bai & Zhao, 2007)	WofBPEL
(Homchuenchom, 2011)	SPIALS: Software Process Improvement Adaptive Learning System
(Adali, Top & Demirörs, 2017)	AssessAgility
(Pino, Garcia & Piattini, 2007)	SPQA.web
(Sabato & Filangieri, 2002)	ASSO: a Web-portal application for operational process assessment and organizational improvement
(Astorga-Vargas et al, 2014)	Autoevaluación de Requisitos de Procesos y Atributos de Procesos (AURAP)
(Pino, 2010)	Software process improvement – SPI
(Patel & Ramachandran, 2009)	Automated Tool Support
(Choi, Kim & Park, 2012)	ReMo
(Garcia & Pacheco, 2009)	SysProVal
(M. Nawazish et al, 2010)	MECA (Monitor, Evaluate Control, and Act)

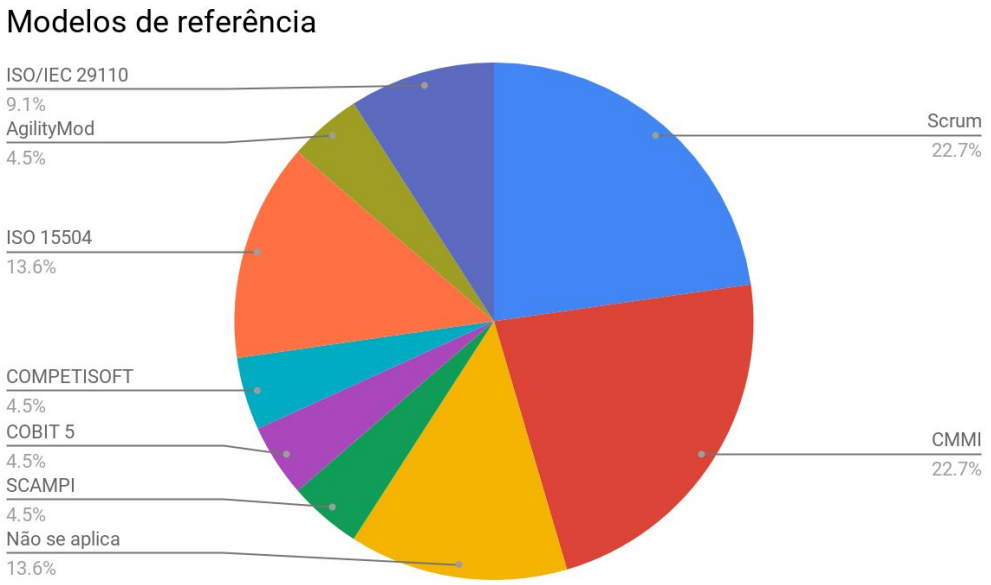
Fonte: Elaborado pelos autores

3.4.3 Modelos de referência

Foram analisados os modelos de referência utilizados na avaliação de processos. A maior parte dos estudos utilizaram SCRUM e CMMI como modelo de

referência. Alguns trabalhos apontaram mais de um modelo utilizado, e outros não apontaram nenhum ou não se aplicava ao estudo. O seguinte gráfico mostra a quantidade de estudos que utilizaram os seguintes modelos de referência:

Figura 8: Modelos de referência



Fonte: elaborado pelos autores

Dentre os modelos de referência apresentados no gráfico, destaca se pelo maior uso o CMMI, utilizado por 5 estudos (Homchuenchom, 2011; Pino, Garcia & Piattini, 2002; Patel & Ramachandran, 2009; Garcia & Pacheco, 2009; Khokhar et al., 2010). E o modelo Scrum também com 5 estudos (Montenegro & Arévalo, 2018; Suteecca & Ramingwong, 2016; Adali, Top & Demirors, 2017; Homchuenchom, 2011; Zaquali & Ghannouchi, 2016). Já a ISO 15504 apresentou 3 estudos (Pino, Garcia & Piattini, 2007) em comparação com a ISO 29110 utilizado em 2 estudos (Varkoi, 2010; Suteeca & Ramingwong). Tanto COBIT 5 (Montenegro & Arévalo, 2018) como AgilityMod (Adali, Top & Demirors, 2017) e COMPETISOFT (Pino, 2010) apresentaram apenas 1 estudo para os modelos de referência.

É interessante observar que mais de um quarto dos estudos (27,2%) realizaram avaliações de processos utilizando como base modelos ágeis (Scrum e AgilityMod) o que levanta indícios de uma preocupação com alinhamento do processo, mesmo para os modelos ágeis. Naturalmente, a pesquisa realizada buscou por ocorrências desse tipo para melhor entender como as VSE's adaptam modelos ágeis em seus processos. Todavia, mais de dois terços (13,6%) da pesquisa apresentaram estudos em que modelos de referência não foram identificados, o que mostra que nem todas as avaliações de processo seguem modelos ou normas.

3.4.4 Notações

Na coleta de dados, foram analisados também as notações utilizadas para descrever o processo de avaliação. Muitos dos trabalhos não apontaram uma notação específica, ou descreveram as atividades em uma linguagem própria. A notação BPMN ainda sendo uma das palavras chaves da pesquisa, foi apontada por apenas 2 estudos (CALABRÓ, Antonello; LONETTI, Francesca; MARCHETTI, Eda, 2015 , ZAOUALI, Sirine; GHANNOUCHI, AYACHI, Sonia, 2016).

Podemos observar com os dados obtidos que 56.3% dos estudos não apresentaram uma notação ou então não se aplicava ao caso de estudo. Todavia também teve estudos que utilizam notações informais para modelagem como por exemplo Patel & Ramachandran (2009) que apresentaram uma notação própria para representar seus processos.

Além do BPMN e a notação informal, ainda podemos destacar as seguintes notações utilizadas: a UML (Astorga-Vargas et al. 2014), o SPEM (Montenegro & Arévalo, 2018), o IDEF0 (Suteeca & Ramingwong, 2016) e o PmCOMPETISOFT (Pino, 2010). Abaixo, pode-se observar o gráfico que mostra as notações utilizadas através da coleta de dados:

Figura 9: Notações utilizadas

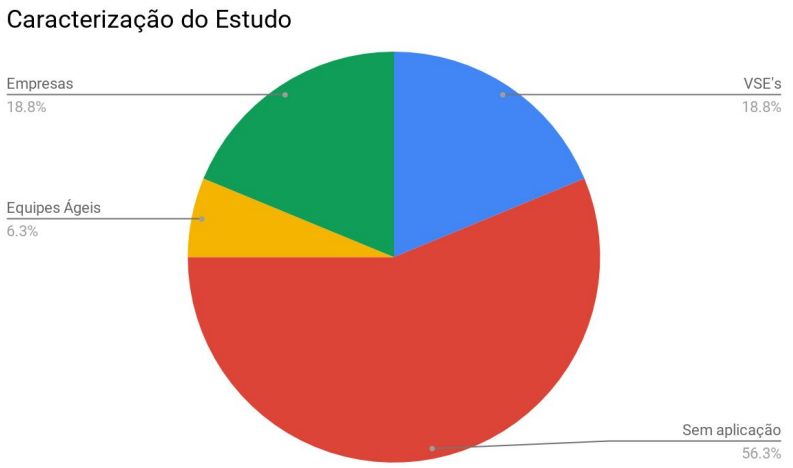
Fonte: elaborado pelos autores

3.4.5 Aplicação do estudo e resultados

A parte referente a caracterização do estudo que envolve identificar aspectos como contexto, a população e a amostra da aplicação dos trabalhos encontrados mostrou que muitas ferramentas criadas não foram colocadas em prática para

obtenção de resultados reais. Ainda assim, foi possível identificar que certas ferramentas são voltadas para as VSE's (Homchuenchon, 2011; Varkoi, 2010; Pino, 2010), outras para equipes ágeis (Zaouali & Ghannouchi, 2016) ou grandes organizações (Montenegro & Arévalo, 2018; Pino, Garcia & Piattini, 2007) como o caso da Organização do Setor Público (PSO), o Hospital Eugenio Espero (EEH) do Ministério da Saúde do Equador ou ainda a Unisoft Colômbia Ltda da Colômbia.

Figura 10: Caracterização do Estudo



Fonte: Elaborado pelos autores

Entretanto, mais da metade (56,3%) das publicações que tiveram uma caracterização de estudo feita, não apresentaram em sua maioria resultados aplicados na prática (Zaquali & Ghannouchi, 2016; Homchuenchon, 2011; Varkoi, 2010) o que impossibilitou analisar resultados observados da aplicação em estudo empírico. Já entre os estudos em que se obteve resultados (43,7%), Adali, Top & Demirors (2017) provou estar cobrindo todas as fases do modelo de avaliação, a eficiência provou reduzir o esforço despendido na avaliação, em média, de 30% em contraste para a avaliação manual. Pino, Garcia & Piattini (2007) em sua caracterização de estudo concluiu-se que a experiência foi enriquecedora tanto para a empresa como para o grupo de melhoria no comando da ferramenta de avaliação de processo utilizada. Por sua vez Pino (2010) teve resultados satisfatórios em suas aplicações, gerando bom resultado do ciclo de melhoria, permitindo às organizações terem uma melhor visão e controle do desenvolvimento de software e processos de gerenciamento de projetos. Já o estudo de Patel & Ramachandran (2009) foi aprovado por todos os gerentes e técnicos que testaram a ferramenta pois apoiaram muito a ideia de um framework de melhoria de processos para o desenvolvimento de software ágil.

3.5 Ameaças à validade

Quanto à abrangência e sua relevância em cima dos estudos retornados, levando em consideração que abordagens automatizadas para avaliações de processos, principalmente em VSEs ainda estão em fase de pesquisa e estudos, os relatos acerca dos resultados encontrados na sua implantação ainda são escassos (menos de 20 estudos encontrados). Assim, uma das principais ameaças à validade desta revisão refere-se à pequena abrangência dos estudos retornados. Outra ameaça se dá ao fato de que em algumas plataformas se obteve mais de mil resultados, então algum estudo relevante pode não ter sido encontrado. Para diminuir o risco de pesquisas incompletas, os termos de pesquisa foram selecionados para descrever os conhecimentos relacionados com a questão de pesquisa e sinônimos dos termos foram utilizados.

Além disso, a qualidade empírica dos estudos ainda iniciais de ferramentas para avaliações automatizadas de processos é considerada baixa, devido incompletude dos relatos de alguns estudos, por exemplo, foi encontrado um estudo para aplicação de BPMN em avaliações de processos, todavia o estudo não apresenta uma ferramenta automatizada para realizar as avaliações (ZAOUALI & Sirine, 2016; GHANNOUCHI, 2016). Já em outro estudo encontrado, é apresentado uma ferramenta chamada SPIALS para auto avaliação de processos (HOMCHUENCHOM et al, 2011) mas não foi encontrado resultados da implantação e eficácia da ferramenta na prática. Como um todo, dos 16 artigos coletados para análise de dados, 10 não possuem resultados observados em estudos empíricos e 9 não têm características de estudo como contexto, população, amostra, etc.

4 ANÁLISE E PROJETO

Este capítulo apresenta a análise dos requisitos, identificação dos processos utilizados nas equipes onde será aplicado o estudo e também uma descrição da arquitetura geral da ferramenta, juntamente com as tecnologias que serão utilizadas. São também analisadas as métricas que as equipes ágeis utilizam no dia a dia para gerenciamento de seus projetos, de forma a poder implementar a automatização da coleta de dados.

Ao final são apresentadas as propostas de protótipos de alta fidelidade de tela para o sistema, referentes às histórias de usuário sintetizadas depois das entrevistas com os usuários.

4.1 Análise do Contexto dos Processos

O Laboratório Bridge (Laboratório Bridge, 2019) é um laboratório integrado ao Centro Tecnológico da Universidade Federal de Santa Catarina voltado para a pesquisa e inovação em tecnologia da informação, entregando produtos para qualificar a gestão pública. O laboratório possui mais de 100 colaboradores e entre eles, a grande maioria são estudantes da Universidade Federal de Santa Catarina.

Um de seus projetos é o e-SUS AB (Laboratório Bridge, 2019), um sistema de prontuário eletrônico para ser utilizado nas unidades básicas de saúde do Brasil. Outro de seus projetos é o SISMOB (Laboratório Bridge, 2019), sistema que realiza o monitoramento de obras de engenharia e infraestrutura de diversos estabelecimentos de saúde financiados pelo Ministério da Saúde, e o RNI (Laboratório Bridge, 2019), que é o sistema de registro nacional de implantes da ANVISA.

Para a realização e aplicação dos estudos deste trabalho de conclusão de curso (conforme declaração de consentimento no Apêndice A), será realizado o enfoque em duas equipes ágeis que trabalham no projeto e-SUS.

A maior parte das equipes ágeis que integram o projeto e-SUS AB, utilizam o Kanban no gerenciamento de suas tarefas. Esse trabalho é feito através de uma planilha base comum para todas as equipes, em que cada uma popula com os seus respectivos dados, fornecendo as métricas específicas para cada equipe.

Uma das equipes tem como integrante a autora deste trabalho, e a outra equipe a autora tem contato direto. Por esse motivo foi feita a escolha dessas duas equipes, por critérios de proximidade e conveniência.

Para o levantamento dos requisitos foram realizadas entrevistas com membros das duas equipes selecionadas e também com o responsável pelo gerenciamento de OKRs dentro do laboratório. A análise do contexto foi realizada por meio da modelagem descritiva do processo realizado pelas duas equipes.

4.2 Modelagem Descritiva dos Processos

Esta etapa do trabalho foi traçada diretamente com as equipes do laboratório Bridge que vão servir de base para o estudo e aplicação da avaliação. As duas equipes ágeis utilizam a plataforma GitHub tanto para o gerenciamento de tarefas como para gerenciamento de código. Cada equipe tem seu próprio projeto dentro do repositório da organização. Em cada projeto chamado de *board* as equipes incluem suas tarefas e os seus respectivos estados, chamados de coluna.

A planilha do Kanban utilizada pelas equipes é populada semanalmente de acordo com o estado das tarefas no seu *board.* Nesse documento se encontram gráficos e informações importantes para o direcionamento do planejamento de entregas de equipe. Essas métricas são explicadas na seção “4.5.1 Análise dos indicadores” .

4.3 Entrevistas com os usuários

Foram realizadas duas reuniões e entrevistas com cada equipe para alinhar seus processos e posteriormente identificar pontos que sejam interessantes para a avaliação, obtendo indicativos relevantes para o estudo.

Em paralelo às conversas com as equipes, também foram discutidos os possíveis requisitos e levantado ideias com a colaboradora Luisa Lacerda, responsável pela implantação e acompanhamento do Kanban nas equipes ágeis. Nessa conversa, revisamos os requisitos e importância levantado pelas equipes, onde as equipes achavam necessidade principalmente nas métricas em que é necessário para população da planilha do Kanban. Entretanto, por mais que essa seja a necessidade atual, é importante olhar além dessas métricas e levantar dados que até mesmo as equipes ainda não sabem o quão vantajosos possam ser no futuro.

Pelo fato do método Kanban ser abordado constantemente no dia a dia das equipes, algo que foi constatado após as entrevistas, as principais métricas coletadas são as principais para a visualização desse método

Nas próximas seções é feito um estudo do processo detalhado de cada equipe ágil escolhida para o estudo de caso, as equipes Supernova e Royal Flush.

4.3.1 Equipe Supernova

A equipe supernova possui cinco integrantes, dentre eles, existe um analista de sistemas, responsável pela análise e documentação das tarefas, três

desenvolvedores *fullstack,* responsáveis pelas implementações, e dois testadores, responsáveis pela qualidade das tarefas.

A primeira reunião com a equipe foi realizada no dia 8 de outubro de 2018, e contou com a presença do gerente de projetos e dos componentes da equipe, incluído a autora que faz parte dessa equipe de trabalho. Neste momento foi relatado a ideia geral dos estudos e instigado os participantes a pensarem em possíveis indicadores consideráveis para poder obter-se uma análise de como seus processos fluem.

A segunda reunião foi realizada no dia 18 de outubro de 2018 e contou apenas com os membros da equipe. Foi efetuado um *brainstorming* e coletado diversas ideias que posteriormente se tornaram requisitos funcionais do sistema.

A terceira reunião aconteceu no início do segundo semestre de 2019, e foi importante para revisar todos os requisitos levantados anteriormente, ainda mais que nesse momento o Kanban está mais desenvolvido no dia a dia das equipes.

Mesmo a autora fazendo parte da equipe e estando ativamente envolvida nos processos, alguns pontos também foram levantados para a definição da modelagem do processo da equipe. A decisão foi de modelar o processo num nível maior, no nível de módulos.

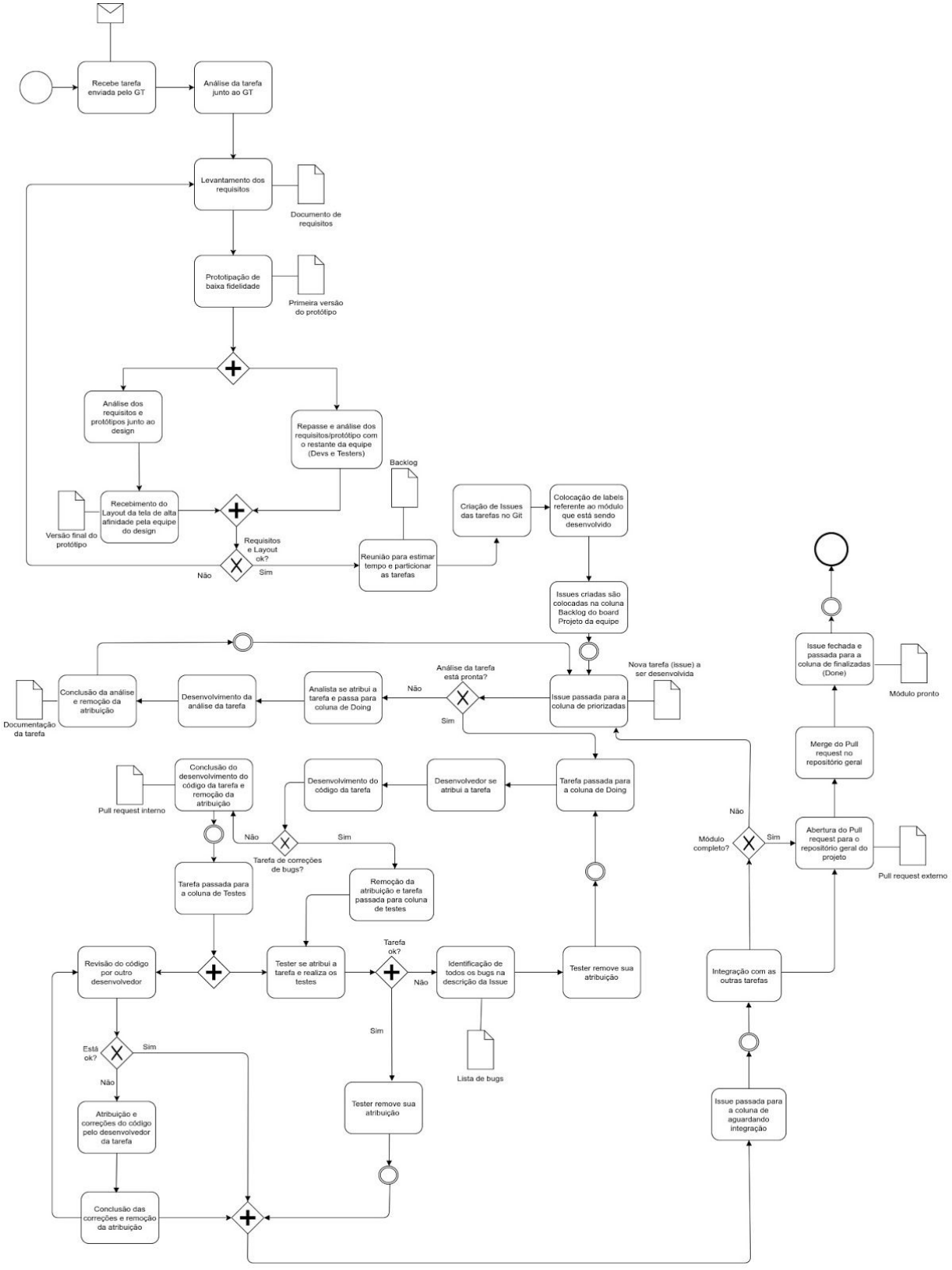
O módulo é um entregável com um nível alto de complexidade que no fim gera um produto dentro do projeto. Os ícones de eventos intermediários dispostos pelo fluxo (um círculo dentro de outro) representam as etapas onde vai ter coleta de indicativos.

Então, nesse sentido, segue na figura 9 a modelagem do processo, utilizando a notação BPMN 2, desde que um módulo chega como demanda para equipe até a momento que esse módulo é integrado com o restante do projeto:

Nomenclatura utilizada na modelagem dos processos:

1. **Git** - Plataforma GitHub
2. **GT** - Grupo de trabalho do Ministério da Saúde
3. **Devs/Testers** - Desenvolvedores fullstack/Testadores
4. **Board** - Quadro com várias colunas, que representam o estado de uma tarefa.
5. **Issue** - Pequena tarefa a ser desenvolvida cadastrada em uma issue do GitHub
6. **Pull request interno** - Pull request aberto no repositório interno, onde apenas membros da equipe e convidados tem acesso.
7. **Pull request extern**o - Pull request aberto no repositório geral do projeto.
8. **Merge** - Quando um pull request externo é integrado ao código geral do projeto.
9. **Label** - Etiqueta colocada na issue com informação

Figura 11: Fluxo do processo da equipe Supernova



Fonte: Diagrama elaborado pela autora.

4.3.2 Equipe Royal Flush

Da mesma forma e dinâmica descrita pela equipe Supernova, a equipe ágil Royal Flush possui uma analista, três desenvolvedores fullstack e dois testadores.

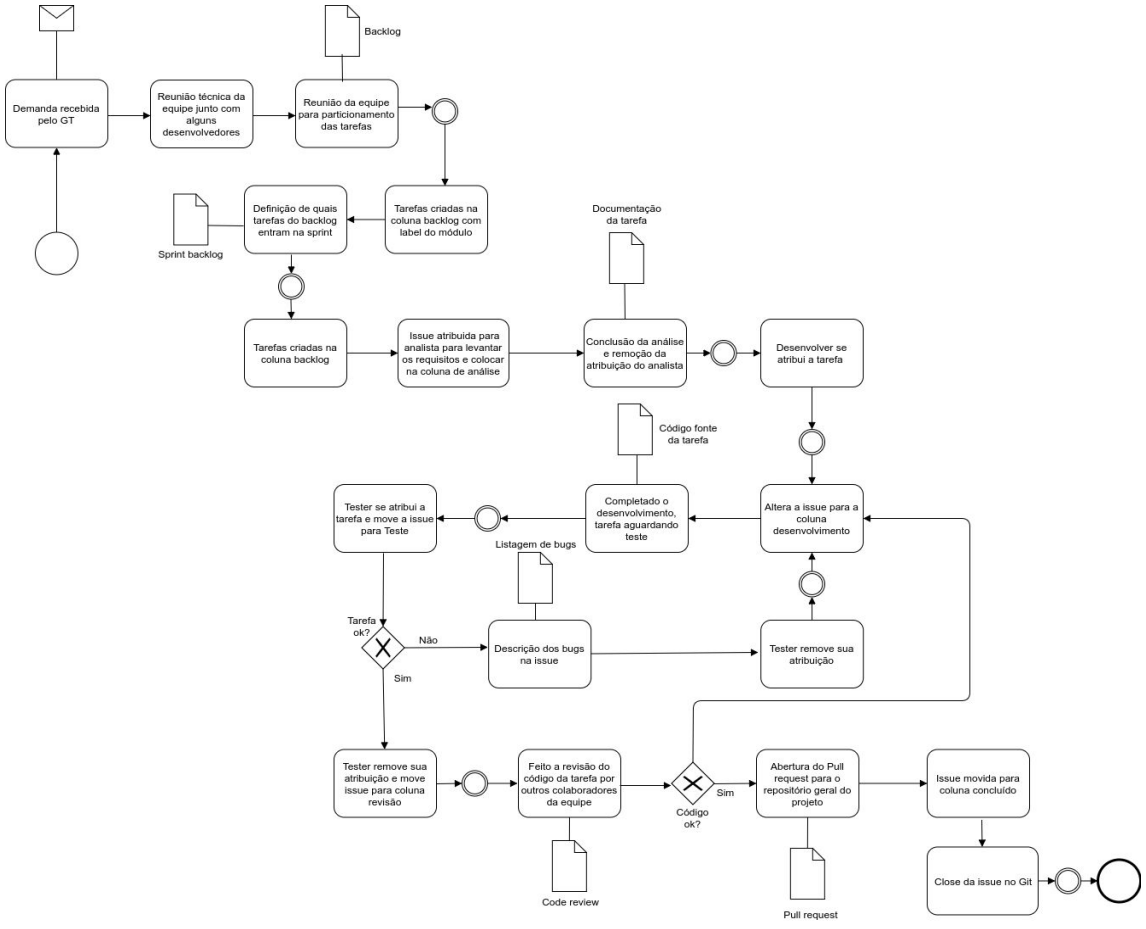
A equipe trabalha também com demandas complexas, ou seja, os módulos. Cada módulo é planejado com reuniões técnicas e partilhado em menores tarefas. Essas menores tarefas ficam em um backlog, onde ao final de cada semana é decidido quais tarefas entrarão na próxima *sprint*.

As reuniões foram realizadas no início do segundo semestre de 2019, e nesse momento foi buscado entender como a equipe trabalha, como particionam suas tarefas e quais são suas ações quando um novo módulo era designado-los.

Em outro momento foi realizado um *brainstorming,* com ideias do que seria interessante obter automaticamente de uma ferramenta, como quais indicativos e métricas através de dados que o GitHub pode fornecer.

Após a análise do fluxo de tarefas quando um novo módulo é designado, foi realizado a seguinte modelagem do processo da equipe Royal Flush:

Figura 12: Fluxo do processo da equipe Royal Flush



Fonte: Diagrama elaborado pela autora.

4.4 Relação com o Kanban

O Kanban é um sistema de controle de produção adotado pela maioria das equipes ágeis do projeto e-SUS do laboratório Bridge. As equipes populam uma planilha com os dados dos boards do GitHub. Esse processo de população da planilha é feito de forma manual.

Essas métricas além de obter uma visualização do processo da equipe, auxiliam na estimativa de tempo de entregas, no processo de partição das tarefas em partes e menores e também ajudam a identificar os problemas que aconteceram em determinadas tarefas que saíram do *lead time* esperado. Como um objetivo geral, o Kanban é utilizada para visualização e melhoria de processos nas equipes.

Um dos membros da equipe, que acompanha de perto a implantação do Kanban nas equipes, juntamente com a equipe de gestão, promoveram uma planilha com todos os dados interessantes para serem populados, e então a partir deles serem plotados gráficos de análise. Todos esses indicadores e gráficos serão abordados na próxima seção: Análise de indicadores.

O preenchimento da planilha é feito semanalmente, mesmo quando as equipes optam por *sprints* quinzenais. Análises feitas como rever as possíveis divergências que tenham acontecido, como por exemplo tarefas demorando mais que o esperado, *lead time* médio oscilando muito ou até identificação de gargalos no processo são situações que podem ser medidas através de poucas semanas utilizando o Kanban.

Além das análises que podem ser realizadas em pouco tempo, o Kanban é visado como um dos pontos fortes para fornecer *key results* para certos objetivos da estrutura de OKR do laboratório. Portanto, é uma método para melhorias a curto prazo e para alcance de objetivos a longo prazo.

4.5 Análise dos indicadores

Esta seção explica todos os indicadores utilizados para geração de métricas no kanban. Todas essas informações constam nas planilhas utilizadas pelas equipes ágeis do laboratório e tem embasamento nas métricas abordadas no livro Métricas ágeis (Albino, 2017).

4.5.1 Quantidade de tarefas em cada estado

Esses são os indicadores que são populados semanalmente (de forma manual) na planilha do Kanban, de acordo com o estado das *issues* no *board* da equipe no GitHub.

Ao final de cada semana estipulado pela equipe, é preenchido a quantidade de cada issue nos seguintes estados: Backlog, Análise, Desenvolvimento, Teste e *Done* (*Throughput*). Apesar de esses serem os estados básicos do Kanban, cada board que cada equipe tem suas próprias colunas estilizadas de acordo com o fluxo do seu processo.

A partir desses dados, alguns indicadores são recalculados após cada semana finalizada, são os indicadores: *work in progress*, *throughput acumulado* e coeficiente de variação.

4.5.1.1 Work in progress

As equipes utilizam como WIP (*work in progress)* na planilha do Kanban a soma da quantidade de tarefas da semana que estão em progresso na semana, ou seja, as tarefas de todos as colunas exceto a coluna *done*.

O que algumas equipes também adotam é limitar a quantidade de tarefas em work in progress. Segundo Albino (2017), Se ao acompanhar sistematicamente o fluxo de trabalho do time perceber que as demandas não estão se movendo continuamente ao longo do fluxo, é bem provável que um gargalo esteja se formando em alguma etapa.

A equipe Royal flush limita o WIP utilizando a seguinte fórmula: quantidade de colaboradores da equipe mais dois. Já a equipe Supernova tem a quantidade de tarefas em progresso por número de colaboradores na equipe mais três.

4.5.1.2 Throughput acumulado

Throughput nada mais é do que uma medida que determina quantas unidades de trabalho foram completadas em uma unidade de tempo. Olhando pela perspectiva de vazão (ou saída), é possível afirmar que o throughput é uma medida que determina quanto o fluxo é capaz de processar (Albino, 2017).

O throughput acumulado utilizado pela planilha do kanban nas equipes, é então, a soma de todas as tarefas finalizadas em semanas anteriores e na semana finalizada.

4.5.1.3 Coeficiente de variação

O coeficiente de variação é a divisão do desvio-padrão pela média multiplicado por 100. É sempre dado em percentual. O coeficiente de variação fornece a dispersão dos dados em torno da média em percentual, constituindo uma medida alternativa ao desvio-padrão. Quando se deseja comparar a variabilidade entre dois conjuntos de dados, o coeficiente de variação é a medida de dispersão indicada (Battisti, 2018).

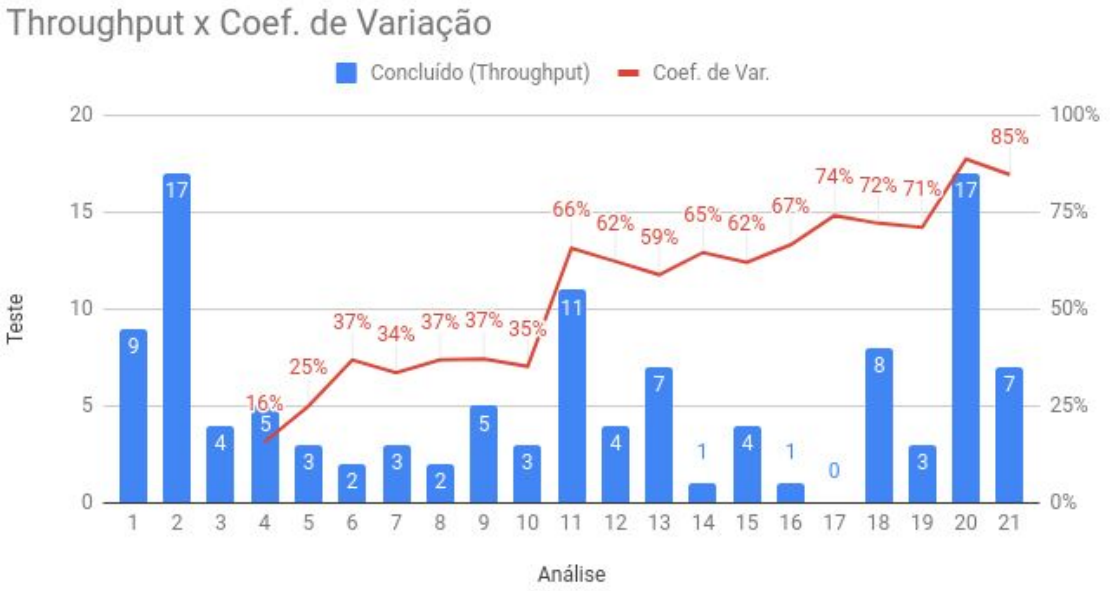
Neste contexto, o coeficiente de variação é uma medida estatística calculada com desvio padrão e média da quantidade de tarefas finalizadas, servindo de grande auxílio para verificar a variação de issues finalizadas ao passar do tempo. Por exemplo, se a equipe tem como objetivo entregas de acordo o *lead time* médio, o coeficiente de variação tende a ficar estável. Porém se por algum motivo, as tarefas finalizadas de uma semana para outra baixaram drasticamente, esse coeficiente tende a variar muito, então as equipes podem parar e analisar os motivos de as entregas não serem constantes.

4.5.2 Throughput x Coeficiente de variação

Para as equipes ágeis que utilizam o Kanban, é determinado como tarefa completada a issue que passou por todas as etapas e foi fechada. A contagem de tarefas concluídas é realizada por semana de acordo com a figura 16.

Na figura abaixo podemos ver as issues finalizadas e o coeficiente de variação ao longo de 21 semanas da equipe Royal Flush:

Figura 13: Gráfico Throughput x Coeficiente de variação



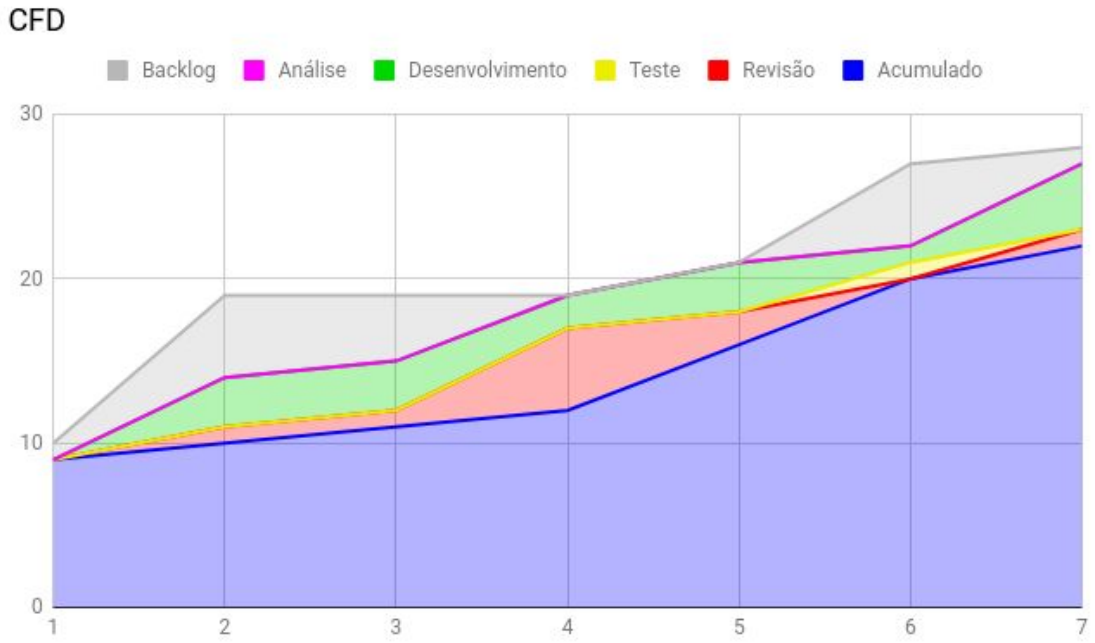
Fonte**:** Planilha Kanban da equipe Royal Flush

4.5.3 Cumulative flow diagram

De acordo com Albino (p. 190, 2017), o CFD (cumulative flow diagram) é uma excelente forma de compreender o fluxo de trabalho ao longo de um processo, afinal, o gráfico pode nos dar um panorama geral do que está acontecendo durante o desenvolvimento de um projeto ou produto. É um instrumento de muito valor no processo de monitoramento em projetos ágeis, pois, usando-o, você poderá rapidamente analisar: quanto de trabalho foi realizado, quanto de trabalho está em progresso e quanto de trabalho ainda precisa ser feito.

A partir da quantidade de tarefas em cada estado por semana, a planilha utilizada plota o seguinte gráfico, como o gráfico CFD da equipe Supernova mostrado a seguir:

Figura 14: Gráfico CFD



Fonte: Planilha Kanban da equipe Supernova

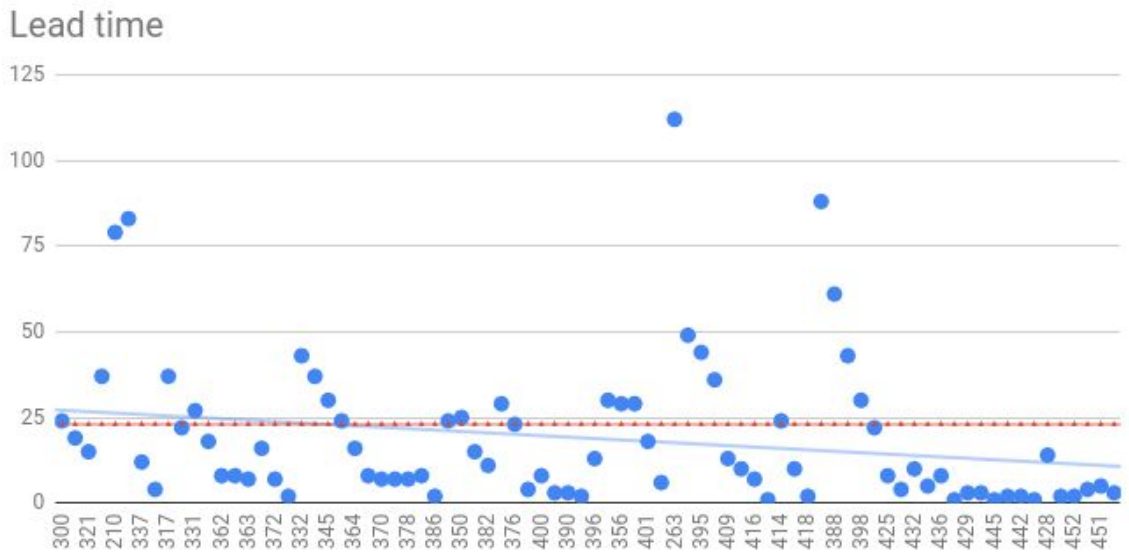
4.5.4 Gráfico de Lead time

Conforme Albino (p.101, 2017), no contexto de desenvolvimento de software, podemos considerar o lead time como sendo o número de dias entre o início e fim do processo de entrega de um item de trabalho (por exemplo, história do usuário, bugs etc.). É o tempo de atravessamento de um item a partir dos limites de entrada e saída definidos no processo de desenvolvimento.

O gráfico de lead time apresenta o tempo em dias de conclusão de cada issue, é uma das principais medidas para análise de tempo de entrega de tarefas. Segundo o gráfico da figura 17, no eixo X é possível observar o número da issue, e no eixo Y o tempo em dias que ela levou para ser completada. Também há uma linha em azul clara que mostra o lead time médio ao decorrer no tempo.

Um indicativo que muitas equipes julgam interessante é o lead time esperado, apontado no gráfico da figura 17 com uma linha vermelha. Nesse caso, a equipe Royal Flush definiu como lead time esperado 23 dias. Dessa forma as equipes podem analisar quanto as tarefas se aproximam ou distanciam da meta esperada.

Figura 15: Gráfico de Lead time



Fonte**:** Planilha kanban da equipe Royal Flush

4.6 Relação com a OKR

A OKR (*Objectives and Key Results*) é uma estrutura utilizada no laboratório Bridge. Foi utilizado no projeto SISMOB, porém tem como objetivo ser utilizado no projeto e-SUS e em todos os demais projetos do laboratório. Como a OKR é alimentada com indicadores de planejamentos de tarefas vindo das equipes, houve um interesse no aprofundamento para uma possível utilização dos indicadores obtidos neste trabalho.

A reunião com o responsável no laboratório pela OKR foi realizada dia 26 de outubro de 2018, onde foi alcançado um entendimento muito maior sobre a grandiosidade desse sistema e seus pontos de entrada. E a partir de então foram realizadas reuniões com as equipes e palestras de alinhamento para que os colaboradores entenderem mais a fundo essa estrutura e os seus benefícios a longo prazo.

4.6.1 Sistema de gestão por OKR

O sistema de gestão por OKR no laboratório Bridge visa um maior alinhamento, transparência e priorização de seus objetivos como organização. O sistema é dividido em três pilares: estratégico, tático e operacional.

Os objetivos estratégicos se realizam com ações de longo prazo. As ações de médio prazo dizem respeito aos objetivos táticos e as ações de curto prazo visam alcançar os objetivos operacionais.

Cada pilar tem seus objetivos, e para esses objetivos serem alcançados é preciso dos *key results* designados para cada um. Para um objetivo pode haver um ou mais *key results.* Os objetivos táticos estão ligados ao Kanban, pois cada equipe, primeiramente separado e depois em conjuntos, fornecerá *key results* para alcance de ações a curto prazo, como melhoria contínua de entregas.

O sistema desenvolvido neste trabalho conclusão de curso, deve então, gerar indicadores para proporcionar a coleta de key results que possam ser utilizados para o alcance dos objetivos do sistema de OKR do laboratório. Portanto, o Kanban auxilia as equipes na melhoria contínua de seus processos, e o OKR auxilia de uma forma geral a análise de melhorias na organização como um todo.

4.7 Levantamento de requisitos

Nesta seção são apresentados os requisitos levantados durante as entrevistas com as equipes. Os requisitos funcionais se fundem em algumas histórias de usuários, completadas por propostas de protótipo de tela. E também são descritos os requisitos não-funcionais, não somente decorrentes de conversas com os integrantes das equipes, mas também resultantes de pesquisas e estudos sobre as tecnologias recentes e adequadas para o desenvolvimento da ferramenta.

4.7.1 Requisitos funcionais

Após a modelagem do fluxo do processo de cada equipe, foi identificado através de eventos intermediários onde seria possível gerar indicadores de forma automática pelo GitHub (Figura 11), e assim realizar avaliações pontuais do processo. A relação de ideias obtidas através dos *brainstorming* com as duas equipes foi filtrada, lapidado e relacionado com os pontos de avaliação dentro do processo.

As equipes utilizam de forma contínua a planilha de métricas do Kanban, portanto foi visto juntamente com as equipes ao longo das entrevistas, quais dados seriam essenciais e que facilitariam os integrantes a popular essa planilha. O resultado desse processo resultou na listagem de requisitos funcionais.

Tabela 5: Requisitos funcionais

RF01	O sistema deve exibir a quantidade de tarefas em cada coluna do projeto da equipe. Essa quantidade deve ser populada semanalmente.
RF02	O sistema deve mostrar a quantidade de tarefas concluídas em uma semana (*throughput*), assim como o cálculo da quantidade acumulada de tarefas concluídas (*throughput acumulado*).
RF03	O sistema deve apresentar a quantidade de tarefas em progresso na semana (*work in progress*), assim como a quantidade total (tanto tarefas em progresso quanto concluídas).
RF04	O sistema deve apresentar o tempo que uma tarefa levou para ser completada (*lead time*), e também o lead time médio das tarefas.
RF05	O sistema deve mostrar a porcentagem de tarefas que ultrapassaram o *lead time* médio.
RF06	O sistema deve obter o valor médio de tarefas em progresso de uma equipe (*work in progres*s médio).
RF07	O sistema deve calcular a porcentagem do módulo pronto, através de tarefas finalizadas com a label referente ao módulo.

Fonte: Elaborado pela autora

Tabela 6: Histórias de usuário

História	**Descrição**	**RF referentes**
HU01	Como um integrante de uma equipe ágil, eu quero visualizar as informações semanais disponíveis sobre as tarefas para obter os dados de entrada para a planilha de análise de métricas da equipe.	**RF01, RF02, RF03, RF06**
HU02	Como um integrante da equipe ágil, eu quero visualizar o gráfico de *throughput* das tarefas, para auxiliar na melhoria de estimativas de entrega da equipe.	**RF02**
HU03	Como um integrante da equipe ágil, eu quero observar o gráfico de *lead time* das tarefas, para obter métricas que auxiliam na estimativa de tempo de tarefas.	**RF06, RF07**

HU04	Como um integrante da equipe ágil, eu quero visualizar o gráfico CFD do projeto (Cumulative Flow Diagram), para observar o progresso das tarefas e a identificação de gargalos no processo.	**RF01, RF03**
HU05	Como um integrante da equipe ágil, eu quero saber a quantidade do módulo pronto, a fim de obter métricas que auxiliem na estimativa de entrega e sirvam de insumo para planilhas utilizadas para análise na equipe e na organização.	**RF07**

Fonte: Elaborado pela autora

4.7.2 Requisitos não funcionais

Após um estudo sobre as ferramentas existentes de integração ao GitHub, foram levantados algumas questões técnicas essenciais para o desenvolvimento da ferramenta, como linguagem, API e biblioteca.

Também nas reuniões realizadas com as equipes surgiram sugestões que podem tornar a utilização da ferramenta mais prática e dinâmica, como o desenvolvimento de um GitHub app, ou seja, o sistema se tornar uma própria extensão da plataforma GitHub.

Tabela 7: Requisitos não funcionais

RNF01	O sistema deve ser web. Compatível com dispositivos *desktops,* nos navegadores *Google Chrome* e *Mozilla FIrefox.*
RNF02	O sistema deve ser integrado à plataforma GitHub, utilizando a *GitHub REST API v3.*
RNF03	O sistema deve ter uma estrutura de banco de dados para armazenar os valores. O banco deve ser PostgreSQL.
RNF04	O sistema deve utilizar o interpretador NodeJS v10.13.
RNF05	O sistema deve ser desenvolvido em JavaScript.
RNF06	O sistema deve utilizar o *express*, um *framework* de servidor padrão para o Node.js.

Fonte: Elaborado pela autora

4.8 Arquitetura geral da ferramenta

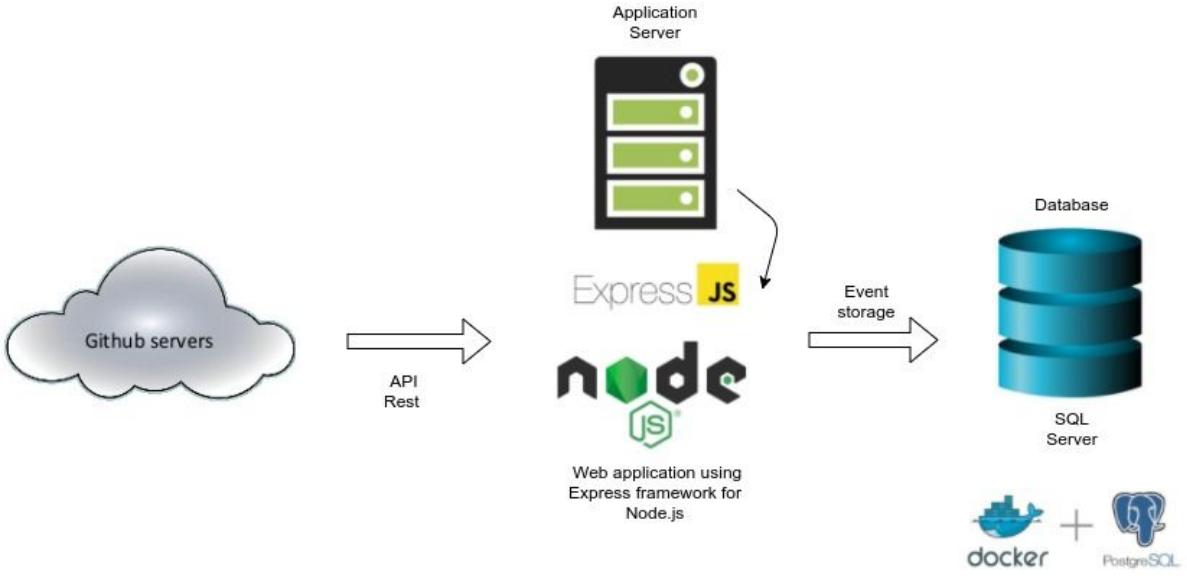
A ferramenta é uma aplicação Web criada com Express, um framework que utiliza Node.js para construções de sistemas. Portanto essa aplicação utilizará JavaScript tanto na parte *backend* quanto na parte *frontend*.

A comunicação da aplicação com o GitHub é realizada através da GitHub API v3, uma API em REST. Onde através de chamadas POST da aplicação, a API retorna os valores requeridos em uma estrutura JSON.

Os valores serão salvos em um banco de dados, este criado com auxílio da ferramenta docker, para salvar todos os estados do banco de dados. O banco de dados é PostgreSQL, ou seja, um banco de dados estrutural, portanto os dados que vem da requisição em JSON terão de ser tratados antes de serem salvos.

Os dados serão salvos no banco através de rotinas, e a cada vez que o usuário utilizar a aplicação, será feito consultas no banco para trazer os dados filtrados.

Figura 16: Arquitetura geral do sistema



Fonte: Elaborado pela autora

4.9 Protótipos de alta fidelidade

A partir da análise dos requisitos levantados e revisados pelas equipes ágeis, foi desenvolvido os protótipos de alta fidelidade para o sistema. Foi utilizado os elementos do Design System Bold (Bold, 2018) desenvolvido dentro do laboratório bridge, onde além de ter todos os componentes para criação de layout, também pode se importar a biblioteca em react para utilizar esses componentes no desenvolvimento de ferramentas. A Bold é aberto para todos e a biblioteca é open source.

As imagens a seguir representam a tela de escolha da equipe, ou seja, escolha do board em que ela participa para filtragem das informações. O segundo protótipo traz então todas as informações relevantes encaminhadas pelos requisitos funcionais. Os gráficos abordados anteriormente pelos indicadores do Kanban também aparecem, e além desses mais um gráfico onde mostra a porcentagem de um certo módulo pronto.

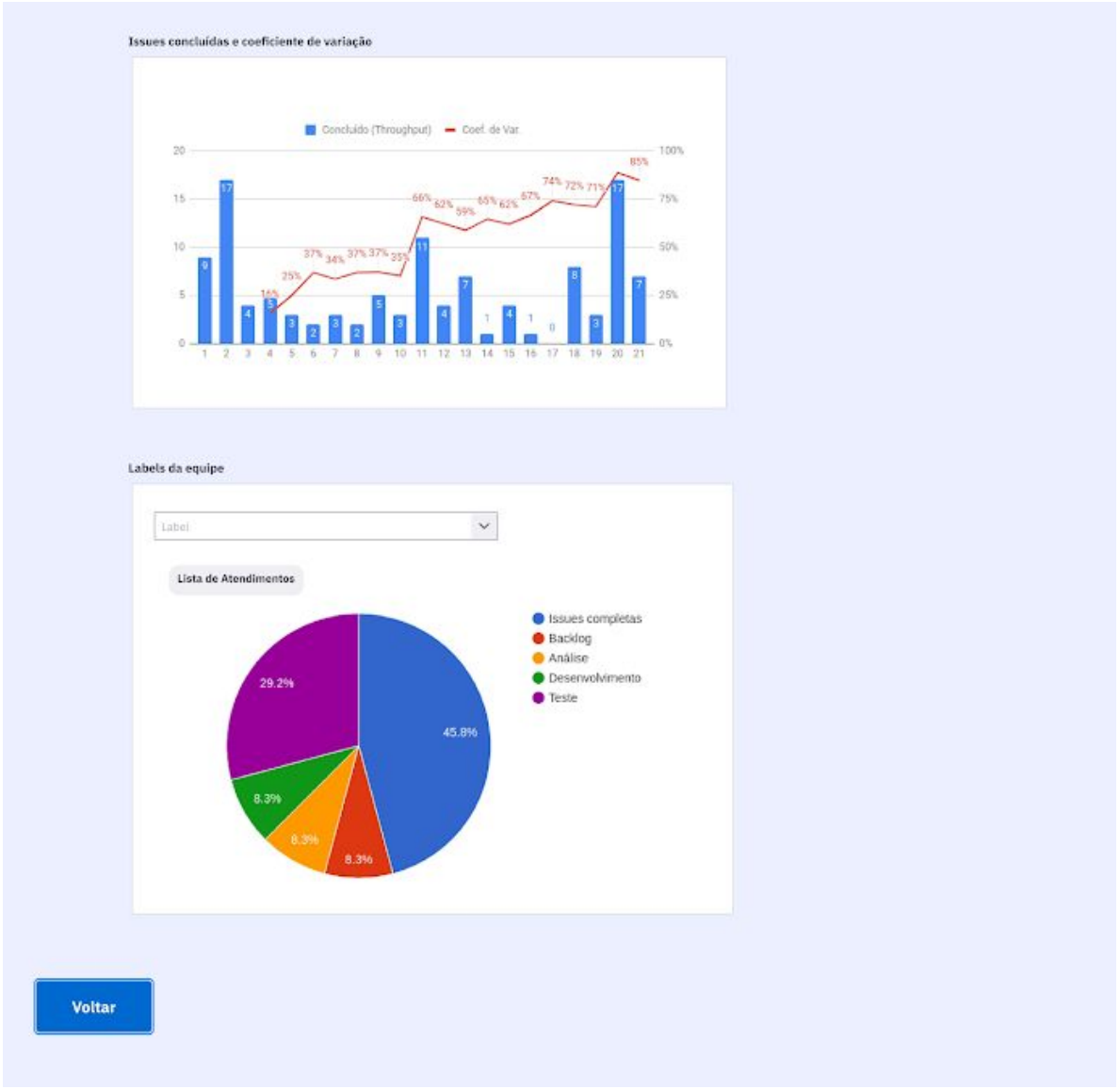
A figura 16, com as informações principais da ferramenta, apresenta:

- Estado semanal das tarefas e estado das Issues, conforme HU01
- Gráfico de *lead time* das issues, conforme HU03
- Gráfico de *Cumulative Flow Diagram,* conforme HU04
- Gráfico de *Throughput* das issues, conforme HU02
- Gráfico do módulo pronto através de labels, conforme HU05

Figura 17: Tela de escolha da equipe

Fonte: Elaborado pela autora

Figura 18: Tela principal de informações da ferramenta



Fonte: elaborado pela autora

5 DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento da ferramenta que obtém indicadores automatizados através da plataforma GitHub, chamada de *AppIssue* (uma junção de *application* com *issues*).

É apresentada a descrição de todos os passos utilizados, as principais decisões de projeto, as dificuldades encontradas e as superadas, as tecnologias utilizadas e também exemplo de funcionalidades implementadas.

5.1 Decisões de projeto

Antes de começar a implementação da ferramenta e configuração do ambiente, uma importante decisão teria que ser tomada: como realizar a comunicação da aplicação com o GitHub.

A primeira tentativa foi realizar a comunicação via *webhook*[2]*,* dessa forma, qualquer ação realizada pelo usuário no GitHub (dependendo das configurações escolhidas) seria enviada à aplicação por meio de um JSON com todas as informações do evento.

5.1.1 Configuração da aplicação teste

Foi utilizada uma plataforma online para codificar a aplicação teste, o *Glitch*[3]*.* Nessa plataforma é possível criar um template de aplicação em Node.js[4] utilizando Express[5]. A plataforma online foi escolhida pois o *webhook* mandaria as respostas para um certo endereço. Dessa forma, a aplicação já estar hospedada na nuvem facilitaria a comunicação.

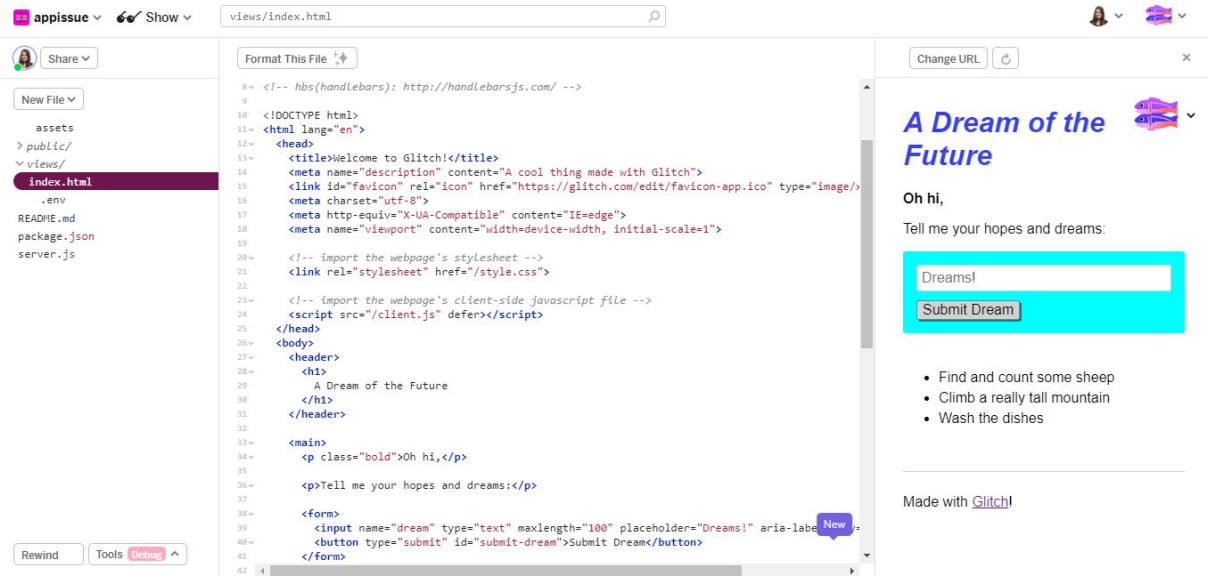
Na plataforma Glitch, foi criado um template em Express, utilizando JavaScript no Frontend e no Backend. Nesse teste não foi modificado em nada na parte frontend, apenas no serviço para tratar os dados recebidos.

Nessa aplicação teste apenas foi modificada a classe server.js, onde são tratados os dados recebidos do webhook, nos métodos POST. É possível realizar todos os testes utilizando o console da próprio Glitch, sem precisar então realizar modificações na parte visual.

Na figura abaixo é possível observar a estrutura geral da ferramenta Glitch, onde ao lado direito está a visualização da página desenvolvida - um exemplo genérico do template criado pela plataforma.

[2] https://developer.github.com/webhooks/
[3] https://glitch.com
[4] https://nodejs.org/en/about/
[5] https://expressjs.com/pt-br/

Figura 19: Estrutura geral da ferramenta Glitch



Fonte: Aplicação teste criada pela autora

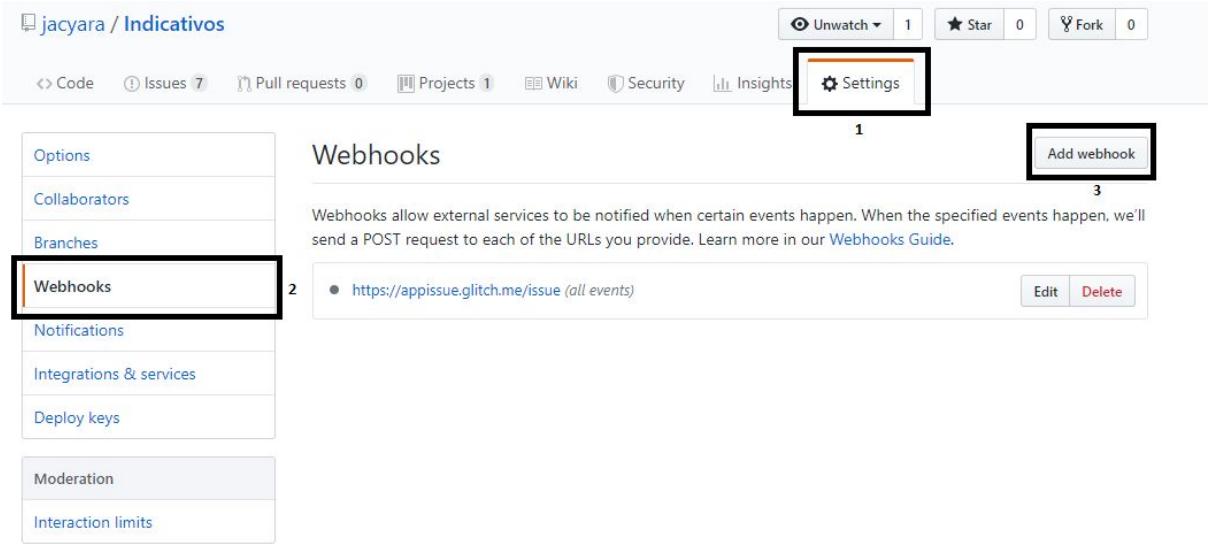
5.1.2 Configuração do webhook no repositório

Para configurar um webhook, é necessário fornecer o endereço para onde os eventos serão enviados. Nesse caso, é a aplicação desenvolvida no Glitch. Configurado dessa forma, cada ação tomada no repositório envia um JSON com o evento detalhado para a aplicação.

O webhook foi configurado em um repositório teste criado pela autora, os passos realizados foram os seguintes:

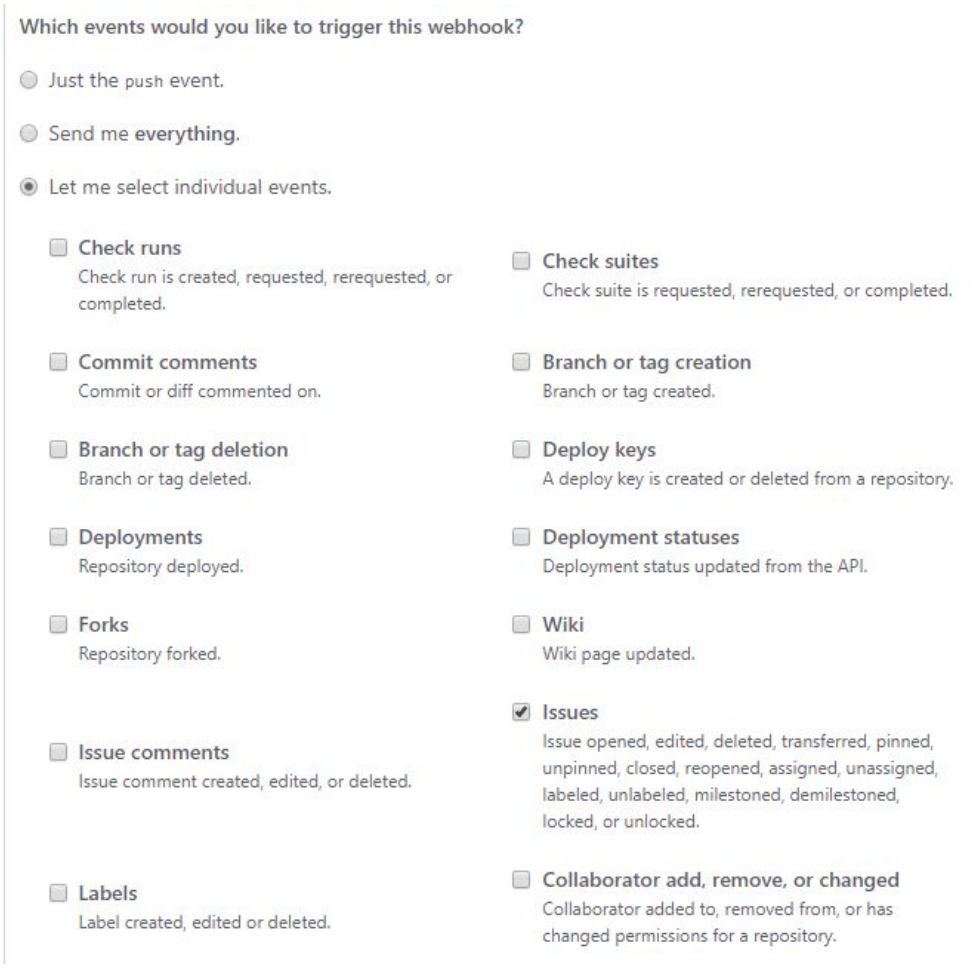
1. Navegar até as configurações do repositório
2. Escolher a opção Webhook
3. Adicionar novo webhook
4. Preencher o endereço para envio dos eventos (endereço da aplicação feita no Glitch)
5. Escolher quais eventos enviar.

Figura 20: Configuração do webhook no repositório



Fonte: Elaborado pela autora

Figura 21: Seleção dos eventos a serem enviados



Fonte: GitHub (https://github.com/jacyara/Indicativos/settings/hooks)

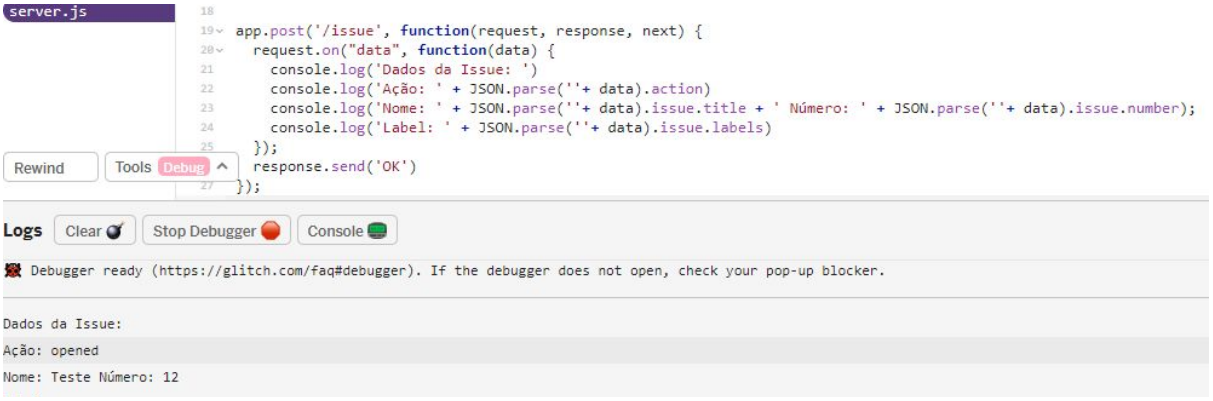
5.1.3 Teste do webhook na aplicação

Um método POST foi implementado no serviço da aplicação para tratar os eventos recebidos pelas ações do usuário no repositório. Foram adicionados alguns logs para observar esses dados.

O seguinte teste foi realizado: adicionado uma issue de número 12 com nome “Teste” no repositório.

Na figura abaixo mostra o método post implementado, com os logs, e abaixo o console onde é possível observar as informações sobre o evento.

Figura 22: Teste recebimento de dados via *webhook*



Fonte: Aplicação teste criada pela autora

5.1.4 Dificuldades dessa arquitetura

A principal característica de uma aplicação com essa configuração é que a comunicação é feita de forma passiva, ou seja, ela só recebe os eventos assim que haja alguma ação. Podem-se destacar alguns problemas que impossibilitaram essa arquitetura a seguir em desenvolvimento.

Primeiramente, a modelagem do banco de dados se tornaria muito extensa e complexa, pois os dados viriam de jeitos diferentes dependendo da ação. Além disso, a ferramenta só teria acesso a dados a partir daquele momento, e não a nenhuma ação feita no passado, o que complicaria para visualização de métricas.

Outro problema encontrado, é que o repositório geral da organização teria que ser configurado para enviar eventos a aplicação, o que se tornaria inviável levando em consideração que o repositório é utilizado por diversas pessoas e é a principal plataforma de trabalho da organização.

Assim, a nova arquitetura escolhida, e definitiva, foi a de realizar requisições à API REST[6] do GitHub, solicitando os dados e obtendo resposta, sem necessitar

[6] https://developer.github.com/v3/

de nenhuma configuração prévia no GitHub. Nas próximas seções é detalhada essa arquitetura.

5.2 Configuração do ambiente

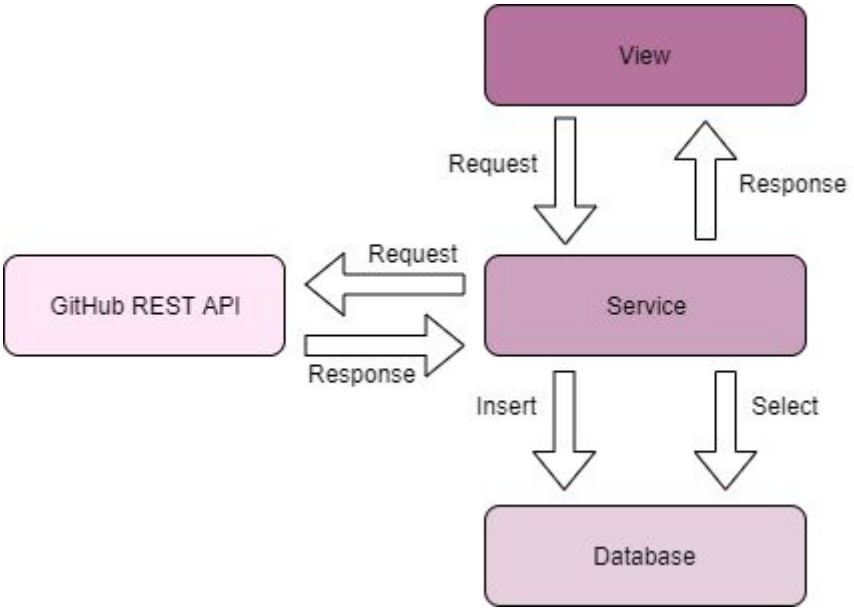
A IDE escolhida para desenvolver a aplicação foi o Visual Studio Code[7]. Primeiramente foi criado uma base para uma aplicação Node.js utilizando Express, do mesmo modo que no primeiro teste. Depois foram instaladas todas as dependências e feita a configuração da aplicação para utilização da biblioteca React [8].

Para configuração do banco de dados, foi utilizado o Docker[9], no qual, por meio de alguns comandos de configuração foi criado o container e a imagem de um banco de dados postgres.

Na camada view, onde são implementadas as telas do sistema, foi utilizado TypeScript - JavaScript com alguns recursos a mais como tipagem. Essa escolha foi dada em função de que os componentes do Design System *bold* são desenvolvidos dessa forma.

Na camada service é onde é realizada a comunicação com a API REST v3 do GitHub, e também é onde são inseridas e consultadas as informações no banco de dados. Para entender melhor essas três camadas a figura 21 mostra uma visão geral da arquitetura da aplicação.

Figura 23: Arquitetura geral das camadas da aplicação



Fonte: Elaborado pela autora

[7] https://code.visualstudio.com/
[8] https://pt-br.reactjs.org/
[9] https://www.docker.com/

Nas próximas seções é detalhado cada uma das camadas desenvolvidas: comunicação com a GitHub API, Banco de dados, Serviço e View.

5.3 GitHub API

O GitHub fornece uma interface para desenvolvedores que desejam desenvolver aplicativos integrados com a plataforma, chamada de API REST que está na versão 3. Por padrão, todas as solicitações para *https://api.github.com* recebem a versão v3 da API REST.

Neste trabalho são realizadas requisições do tipo GET para a API do GitHub. Para cada evento será enviada a requisição para um endereço específico, podendo conter header de Accept, conforme descrito na documentação da API do GitHub.

A fim de testar as requisições para API REST e observar os resultados, sem precisar fazer toda a implementação logo de início, foi utilizado o software *insomnia* [10].

5.3.1 Teste com o *insomnia*

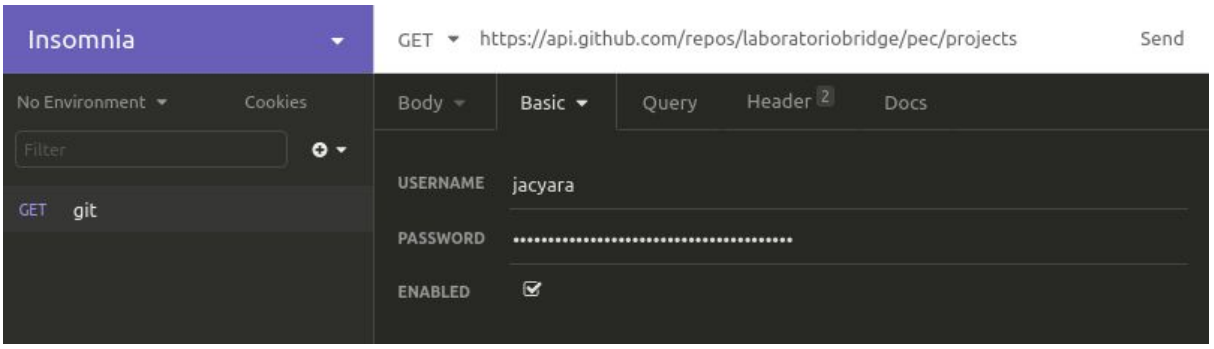
O primeiro passo para realizar o teste utilizando a ferramenta *insomnia* é adicionar o username e token de um usuário com acesso ao repositório que se deseja obter as informações. Depois, é necessário adicionar os headers necessários a partir da v3 da API REST do GitHub, que podem ser encontrados na documentação.

Então, deve-se colocar ao lado da escolha do método (neste caso GET), o endereço da requisição que deseja obter informações. Ao clicar em send, a string em JSON com todas as informações da requisição é mostrada ao usuário.

Na figura abaixo, utiliza-se como exemplo a requisição para obter todos os boards (no caso do laboratório cada board representa uma equipe) que pertencem ao repositório do laboratório.

[10] https://insomnia.rest/

Figura 24: Teste requisição à API REST



Fonte: Teste com ferramenta insomnia elaborado pela autora

5.3.2 Implementação das requisições

Na aplicação foi implementado um método genérico que sintetiza os passos feitos anteriormente. Como parâmetro é passado o endereço da requisição GET. É definido o username, o token (password), os headers (foram adicionados todos os headers que seriam necessários de uma vez só) e o tipo da autorização.

Para realizar as requisições é utilizado o axios[11], dessa forma o método chama uma função get através do axios, com a url passada por parâmetro e os outros parâmetros definidos.

Figura 25: Método geral de requisição à API

```
async function getGitURL(url) {
  let data;
  const username = "";
  const password = "";
  const headers = {
    Accept:
      "application/vnd.github.inertia-preview+json, application/vnd.github.symmetra-preview+json, " +
      "application/vnd.github.machine-man-preview, application/vnd.github.starfox-preview+json",
    Authorization: "Basic " + Base64.encode(username + ":" + password)
  };
  try {
    const response = await axios.get(url, { params: {}, headers });
    data = response.data;
  } catch (error) {
    console.error(error);
  }
```

Fonte: Elaborado pela autora

[11] https://github.com/axios/axios

5.3.3 Métodos utilizados

A fim de obter diferentes informações sobre o repositório, deve-se aplicar uma url específica no método GET da requisição. Esses endereços estão definidos na documentação da API REST GitHub v3. Para esta aplicação foram utilizadas as seguintes url abaixo:

Para obter cada board (equipe) do repositório:

“https://api.github.com/repos/laboratoriobridge/pec/projects”

Para cada coluna de cada equipe:

"https://api.github.com/projects/" + id da equipe + *"/columns"*

Os cards (issues) de cada coluna da equipe:

"https://api.github.com/projects/columns/" + id da coluna + *"/cards"*

Lista de eventos ocorridos no repositório, separados por página:

"https://api.github.com/repos/laboratoriobridge/pec/issues/events?page=" + nº da página

5.4 Banco de dados

O banco de dados utilizado para desenvolver a aplicação foi o PostgreSQL. Para realizar a configuração inicial foi utilizado o Docker. Após essa configuração, foi realizada a conexão do banco de dados no DBeaver[12], ferramenta onde é feita as migrações do banco, assim como testes, consultas e alterações.

Depois de criado o banco localmente, foi feita a conexão desse banco com a aplicação. Foi utilizada a função *client* do JavaScript para realizar essa integração, assim como está descrito abaixo. No código abaixo, primeiro foram definidas as informações básicas como a porta, *host*, *user/password* e o nome do banco. Na segunda parte, é feita a chamada connect para o *client*, concluindo a conexão com o banco.

[12] https://dbeaver.io/

Figura 26: Função conexão com o banco

```
const client = new Client({
  host: "localhost",
  port: 5436,
  user: "root",
  password: "",
  database: "appissue"
});

client.connect(err => {
  if (err) {
    console.error("connection error", err.stack);
  } else {
    console.log("connected");
  }
});
```

Fonte: Elaborado pela autora

Foram então criadas as tabelas e as colunas no banco de acordo com cada dado coletado pelas requisições. A principal tabela na modelagem é a tabela “eventos”, onde é salvo todos os eventos do repositório, relacionando com cards, issues, colunas e board.

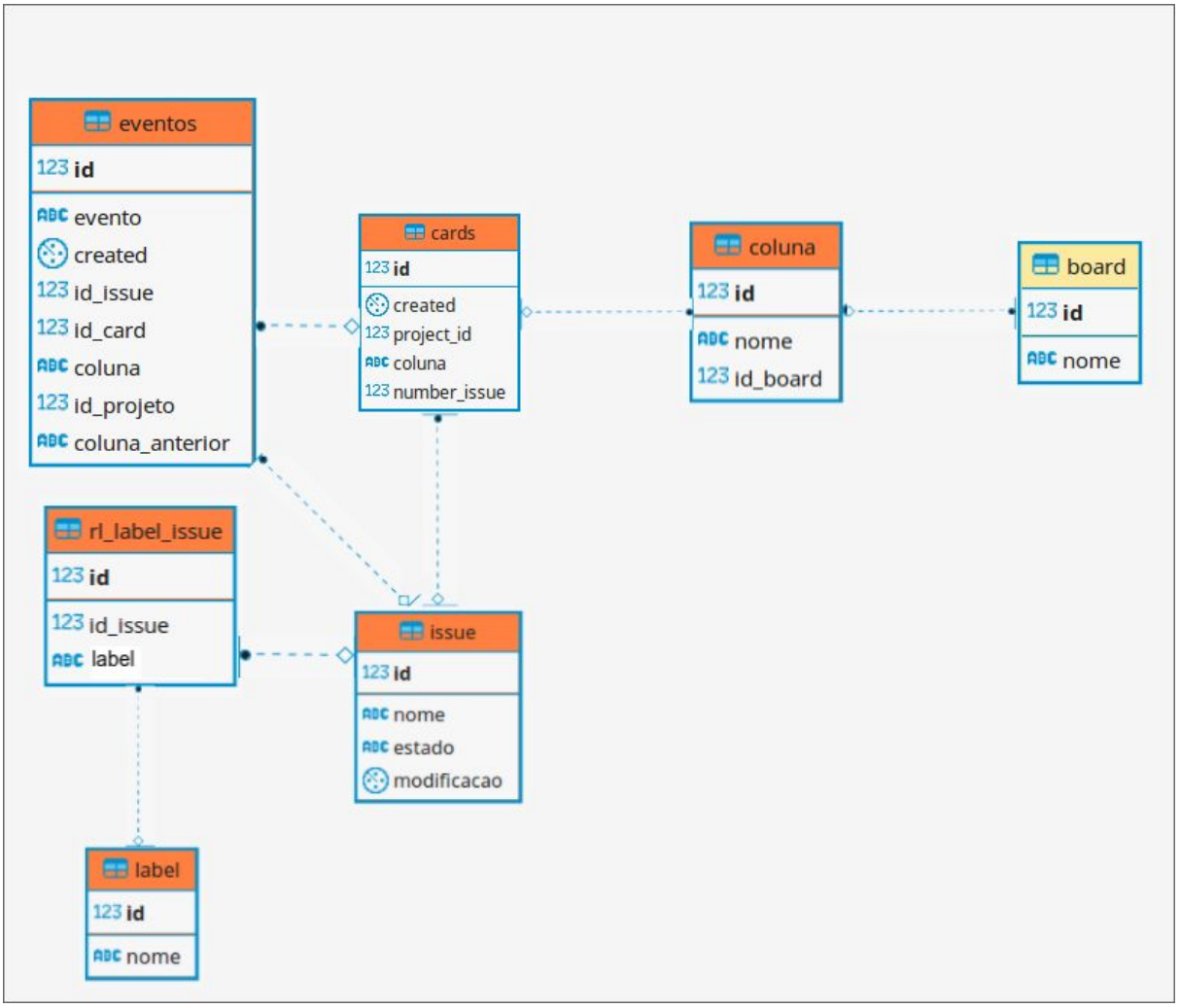
Existe uma tabela para “cards” e outra para “issues” pelo motivo que o card sempre está em um board, portanto sempre está atrelado à uma equipe, o que não acontece em uma issue que não vira um card.

Também foram criado as tabelas: “board”, com todas as equipes, a tabela “coluna”, que contém todas as colunas de cada board, e a tabela “label”, onde está todas as labels do repositório.

A tabela “rl_label_issue” relaciona a issue com as labels, ou seja, por esta tabela é possível saber quais labels estão atreladas em cada issue.

A modelagem completa do banco de dados utilizado por essa ferramenta pode ser observado na figura abaixo.

Figura 27: Modelagem do banco de dados



Fonte: Elaborado pela autora

5.5 Serviço

Na camada de serviço, onde a classe principal é a server.js, é onde são realizadas as operações com o banco de dados. Alguns dos métodos implementados nessa classe já foram explicados anteriormente, como a requisição para a API REST do GitHub e a conexão com o banco de dados.

Além desses métodos, o resultado de cada requisição específica é inserido no banco de dados. E também, o service realiza consultas ao banco de dados toda vez que a cama view faz uma requisição.

5.5.1 Inserção de dados

As funções que inserem dados, sempre utilizam o retorno da resposta do método de requisição à API REST do GitHub. Depois disso, para cada item coletado, é feita a inserção no banco de dados, passando um ou mais parâmetros no meio da query.

No trecho de código abaixo, é inserido no banco todas as colunas de cada projeto (passado por parâmetro) do repositório. É utilizado o método *client.query()* para fazer a integração do trecho sql com o banco de dados.

Figura 28: Método de inserção de colunas

```
async function insertColumns(projectId) {
  const colunas = await getColunas(projectId);
  console.log(colunas);
  colunas.map(item => {
    var sqlSelect = "SELECT board.id FROM board WHERE id = $3";
    var sql =
      "INSERT INTO coluna (id, nome, id_board) VALUES ($1, $2, (" +
      sqlSelect +
      "))";
    client.query(sql, [item.id, item.name, projectId], function(err, result) {
      try {
        if (err) throw err;
        console.log("Number of records inserted: " + result.affectedRows);
      } catch (error) {
        console.log(error);
      }
    });
  });
}
```

Fonte: Trecho de código elaborado pela autora

5.5.2 Consulta de dados

Toda vez que a view faz uma requisição de dados para uma url (explicado mais detalhadamente na próxima seção sobre a camada *view*), o método app.get() no *service* com a url referente é invocado.

Esse método é então executado, podendo conter ou não parâmetros (chamados de request) trazidos pela view. A resposta da query é enviado de volta para a view através do *response.send()*.

No trecho de código abaixo, ao receber uma requisição da view com o endereço “/col”, o método *colunasExpected()* que recebe o id da equipe é executado. É realizado o select passado como parâmetro o resultado do request (id da equipe), e o r*esponse.send()* envia o retorno do select para a view.

Figura 29: Método de consulta de colunas de uma equipe

```
async function colunasExpected(request, response) {
  var sql = "SELECT coluna.nome FROM coluna WHERE coluna.id_board=$1";
  client.query(sql, [request.query.id], function(err, result) {
    try {
      if (err) throw err;
    } catch (error) {
      console.log(error);
    }
    response.send({ result });
  });
}

app.get("/col", colunasExpected);
```

Fonte: Trecho de código elaborado pela autora

Contudo, algumas consultas se tornaram um pouco complexas devido ao fato das regras do Kanban e também pelo fato que impactou no sistema como um todo: cada equipe tem um número e nome de colunas diferentes.

A consulta SQL abaixo retorna a quantidade de issue em cada coluna de cada equipe em um determinado período. É a consulta que retorna as informações bases para um gráfico CFD ou então para as análises de métricas semanais que as equipes realizam. O parâmetro $1 é o id da equipe, e o $2 e $3 são as datas de início e data de fim do período, respectivamente. Essas datas são definidas pelo usuário na *view*.

Figura 30: Consulta issues em cada coluna por período

```
select
    coluna,
    count(coluna)
from
    (
    select
        max(id) as id,
        id_issue
    from
        eventos as e
    where
        created between to_date( $2, 'dd/mm/yyyy') and to_date($3, 'dd/mm/yyyy')
        and id_projeto = $1
        and evento not in( 'closed',
        'reopened',
        'lebeled',
        'unlabeled',
        'removed_from_project')
        and id_issue not in(
        select
            distinct ev.id_issue
        from
            eventos as ev
        join cards on
            cards.number_issue = ev.id_issue
        where
            evento like 'closed'
            and ev.created < to_date($2, 'dd/mm/yyyy')
            and cards.project_id = $1)
    group by
        e.id_issue) as sub
join eventos on
    sub.id = eventos.id
group by
    coluna
```

Fonte: Trecho de código elaborado pela autora

5.6 *View*

Como mencionado anteriormente, o frontend da aplicação é feito na linguagem *Typescript*, uma especialização do JavaScript com tipagem de objetos. A aplicação também é desenvolvida utilizando a tecnologia *React*, pois é uma das bibliotecas atualmente mais utilizadas, utilizada pelo *Facebook, Instagram, Netflix* e outras grandes plataformas.

As classes na view são dividas por funcionalidade, e a classe App é a principal, é essa classe que o index.js chama. A partir da classe App, as outras funcionalidades são chamadas passando os parâmetros necessários de cama uma.

A comunicação da view com o backend, a fim de obter os valores para mostrar na tela, se dá da seguinte maneira: é utilizado um método *axios.get()*

passando por parâmetro a url em que o método no backend será invocado, como foi explicado anteriormente. A resposta do método invocado no service é então colocada em uma *promise* após a chamada *then()* do método na view. É utilizado o *useEffect()* para a requisição ocorrer só uma vez e não ficar replicando os dados infinitamente. Também é possível passar um argumento no final do *useEffect()* para este ser executado novamente quando esse valor mudar.

No trecho de código abaixo, é feita a chamada da view para o backend para trazer todas as equipes. Como essa funcionalidade é chamada uma vez apenas, quando a tela é renderizada pela primeira vez, não é necessário passar argumentos após o *useEffect()*.

Figura 31: Função na view que faz a requisição das equipes para o *service*

```
export default function Equipes() {
  const [projetos, setProjetos] = useState<Projeto[]>();
  const [statusSelecionado, setstatusSelecionado] = useState<Projeto>();

  useEffect(() => {
    axios.get("/equipes").then(resp => {
      setProjetos(resp.data.res.rows);
    });
  }, []);
```

Fonte: Trecho de código elaborado pela autora

Já no trecho abaixo, a view faz a requisição de todas issues abertas para o *service*. E a cada vez que o usuário muda a equipe (também chamada de *project* ou projeto como é visualizada no GitHub) esses dados precisam ser atualizados, por esse motivo é passado o argumento no final da função *useEffect()*.

Figura 32: Função na view que faz a requisição das issues abertas por equipe para o *service*

```
export const Abertas = (props: AbertasProps) => {
  const [abertas, setAbertas] = useState<Abertas[]>();
  const { classes } = useStyles(createStyles);

  useEffect(() => {
    axios.get("/abertas", { params: { id: props.projeto.id } }).then(resp => {
      console.log("abertas: ", resp.data.result.rows);
      setAbertas(resp.data.result.rows);
    });
  }, [props.projeto]);
```

Fonte: Trecho de código elaborado pela autora

5.6.1 *Design System*

O laboratório Bridge possui seu próprio Design System, o *bold*, que é público e pode ser importado para qualquer projeto em React. O Design System já é utilizado no redesign do projeto e-SUS e no sistema de gestão de pessoas interno do laboratório.

Todos os componentes utilizados para o desenvolvimento da aplicação foram importados da biblioteca do *bold,* que são especificados com exemplos no próprio site do Design System (*https://bold.bridge.ufsc.br/*).

5.6.2 Gráficos

A biblioteca utilizada para a geração dos gráficos na ferramenta, foi a biblioteca do Google Charts[13] para aplicações em React.

Para utilizar essa biblioteca, é necessário importar e instalar o *package* no projeto para ter acesso aos componentes definidos como <Chart />. Também é necessário definir os dados para preencher o gráfico.

No trecho de código abaixo, é apresentado a função que preenche um gráfico de *issues* por *lead time,* juntamente com a média desses dias. Essa função percorre todas as issues fechadas e preenche o *array* de dados que possui o id da issue e os dias calculados de *lead time.*

Figura 33: Função que preenche os dados do gráfico de *lead time*

```
function fillData(fechadas: Fechadas[]) {
  const data: any[] = [];
  let acum = 0;
  let i = 0;
  for (; i < fechadas.length; i++) {
    let item = fechadas[i];
    const days = calculaLeadTime(item);
    acum += days;
    data.push([
      "Issue nº: " + item.id + " - " + item.nome,
      days,
      days !== 0 ? acum / (i + 1) : days
    ]);
  }
  return [["Issues", "Dias", "Media"], ...data];
}
```

Fonte: Trecho de código elaborado pela autora

[13] https://www.npmjs.com/package/react-google-charts

5.7 Funcionalidades

Nesta seção é apresentado o fluxo de utilização da ferramenta, com imagens da ferramenta desenvolvida. São mostradas as funcionalidades na versão atual da ferramenta, pretende-se atualizar essas imagens até a entrega final, com a versão mais estável da ferramenta.

1: Selecionar a equipe ágil que pretende-se obter as métricas

Figura 34: Combo de seleção da equipe

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

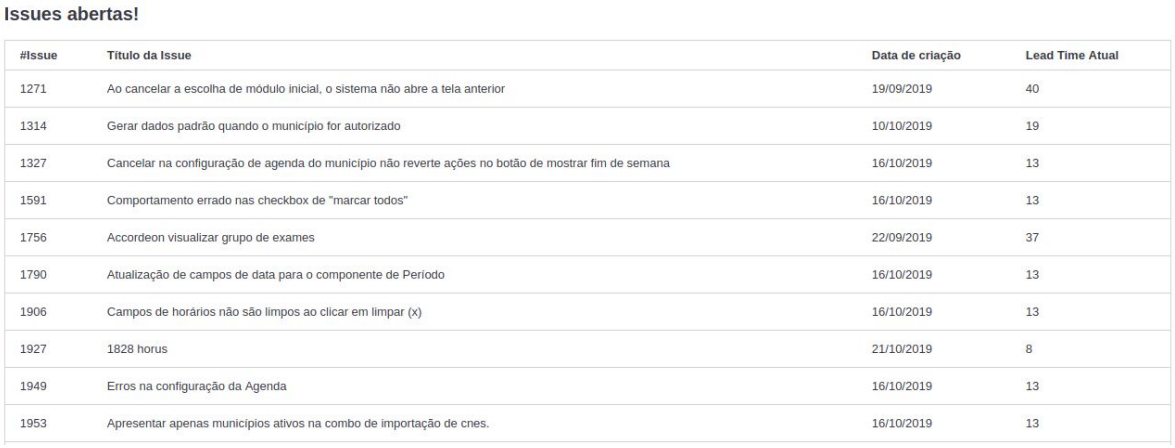
2: Selecionar o período em que pretende-se obter informações sobre os estados das tarefas em cada coluna do *board* da equipe (referente à **HU01**).

Figura 35: Quantidade de tarefas em cada coluna filtrada por período

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

3: Visualização do *lead time* das issues abertas no *board* da equipe (referente à **HU03**).

Figura 36: Tabela de *issues* no estado aberta



Issues abertas!

#Issue	Título da Issue	Data de criação	Lead Time Atual
1271	Ao cancelar a escolha de módulo inicial, o sistema não abre a tela anterior	19/09/2019	40
1314	Gerar dados padrão quando o município for autorizado	10/10/2019	19
1327	Cancelar na configuração de agenda do município não reverte ações no botão de mostrar fim de semana	16/10/2019	13
1591	Comportamento errado nas checkbox de "marcar todos"	16/10/2019	13
1756	Accordeon visualizar grupo de exames	22/09/2019	37
1790	Atualização de campos de data para o componente de Período	16/10/2019	13
1906	Campos de horários não são limpos ao clicar em limpar (x)	16/10/2019	13
1927	1828 horus	21/10/2019	8
1949	Erros na configuração da Agenda	16/10/2019	13
1953	Apresentar apenas municípios ativos na combo de importação de cnes.	16/10/2019	13

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

4: Visualização do *lead time* das issues fechadas no *board* da equipe (referente à **HU03**).

Figura 37: Tabela de *issues* no estado fechada

Issues fechadas

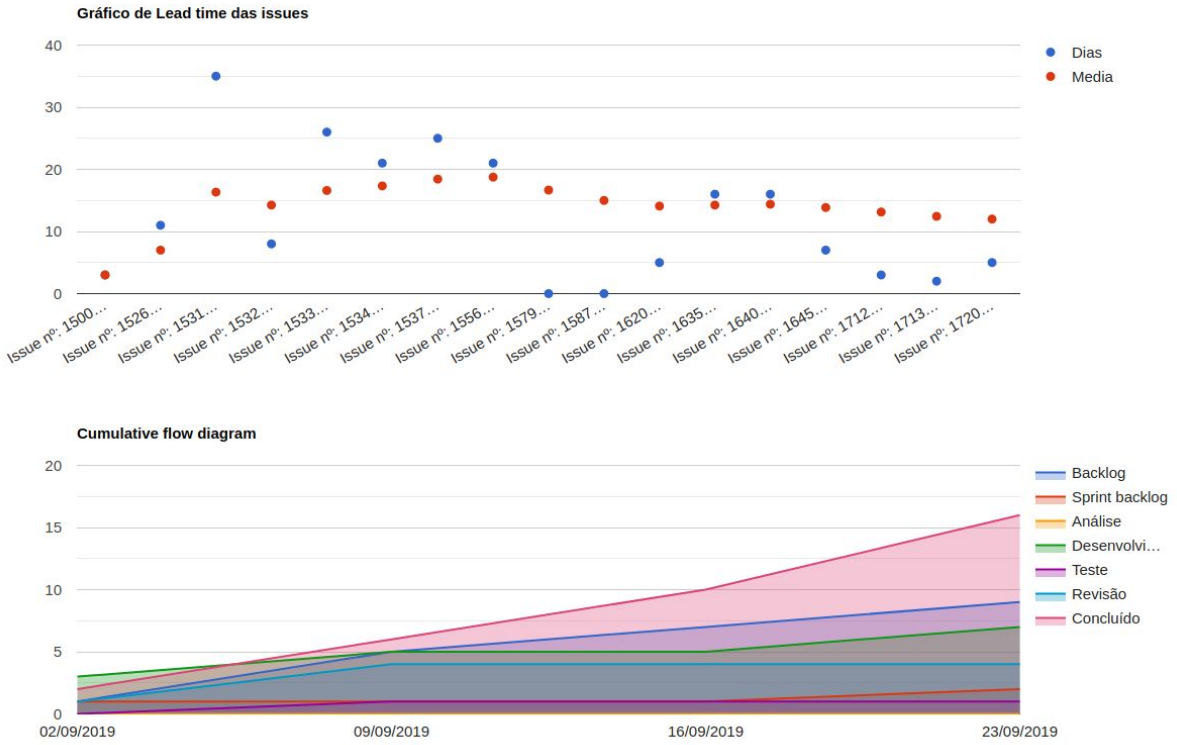
#Issue	Título da Issue	Lead Time
1500	Criação da tela da Busca Ativa	3
1526	Criação da tela contendo o filtro de Problemas e condições	11
1531	Criação da listagem de cidadãos encontrados	35
1532	Elaboração da documentação do módulo Lista de cidadãos por condições de saúde	8
1533	Criação das factories para os testes de integração	26
1534	Acesso do módulo de lista de cidadãos por condições de saúde	21

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

5: Visualização dos gráficos:

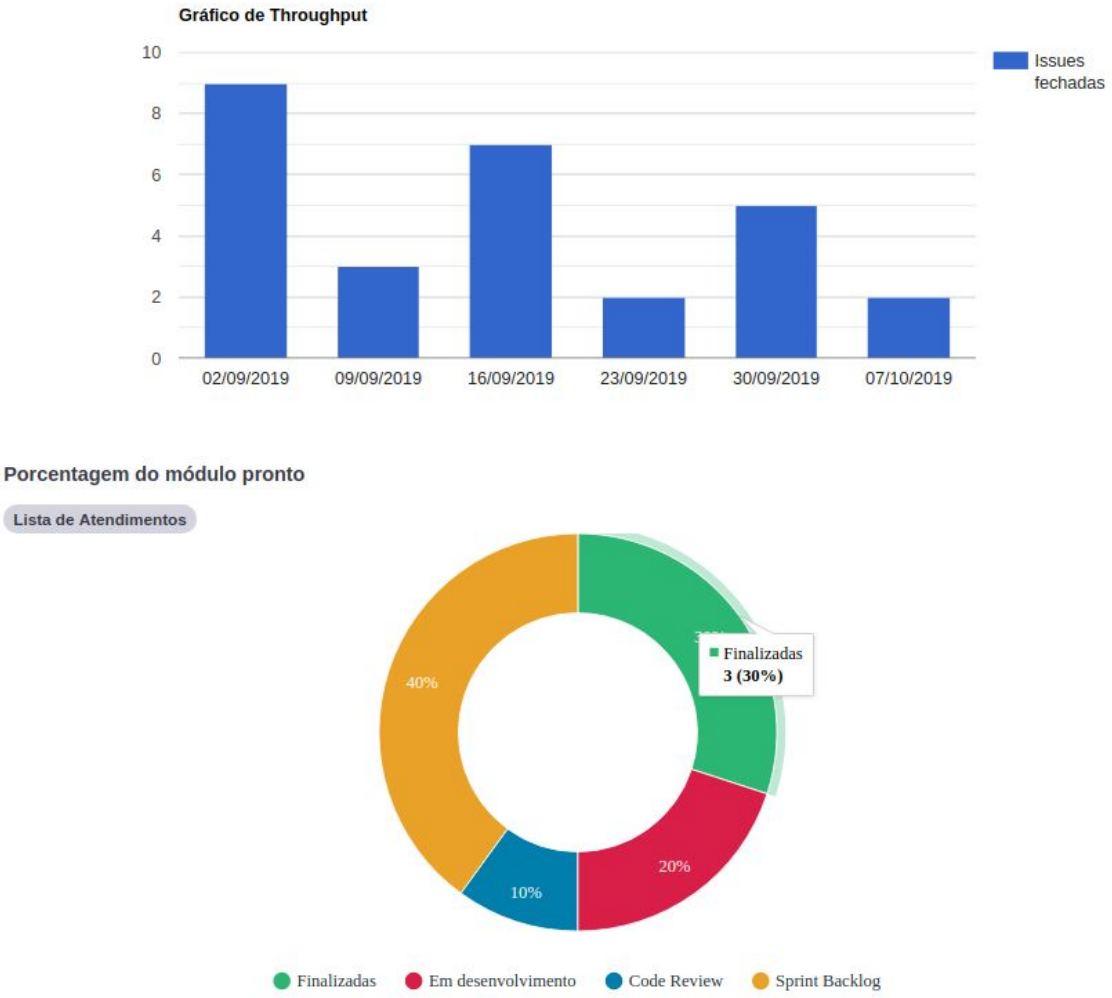
- Gráfico de lead time com as issues fechadas representadas em azul e a média acumulada representada em vermelho (referente à **HU03**).
- Gráfico *Cumulative Flow Diagram* representando a quantidade de tarefas acumuladas separadas por colunas no *board* da equipe (referente à **HU04**).
- Gráfico de Throughput, mostrando a quantidade de issues fechadas por semana (referente à **HU02**).
- Gráfico de Porcentagem do módulo pronto, apresentando quantas tarefas pertencentes àquela label estão em cada estado (referente à **HU05**).

Figura 38: Gráficos de *lead time* e CFD



Fonte: *Screenshot* da ferramenta desenvolvida pela autora

Figura 39 - Gráficos de Throughput e Porcentagem do módulo pronto



Fonte: *Screenshot* da ferramenta desenvolvida pela autora

6 AVALIAÇÃO

Este capítulo apresenta a avaliação da utilidade e da funcionalidade da ferramenta desenvolvida, com enfoque no ambiente de desenvolvimento ágil. A avaliação engloba participantes de equipes ágeis do laboratório Bridge, e também a gerência de desenvolvimento de projetos.

Primeiramente são definidos os objetivos da avaliação, e na sequência são apresentadas as motivações para a escolha das técnicas de avaliação adotadas. Logo após, são apresentadas a execução da avaliação e a análise dos resultados.

6.1 Objetivos

A avaliação tem como principal objetivo:

> Avaliar a funcionalidade e utilidade da ferramenta de coleta de métricas ágeis integrada à plataforma GitHub, sob o ponto de vista da gerência de desenvolvimento e membros de equipes ágeis.

Desta forma, em termos de funcionalidade e utilidade, pretende-se avaliar a qualidade do sistema. A funcionalidade estabelecida precisa ser satisfeita pelo software quando é usado em condições específicas, devendo ser adequada, precisa e segura (ISO/IEC 29110-5-1-2, 2011). Conforme a ISO/IEC/IEEE 24765 (2017), um ferramenta de software possui utilidade quando ela é projetada para executar alguma função de suporte frequentemente usada.

Inicialmente, é avaliada a utilidade da ferramenta num contexto geral em ambientes de desenvolvimento ágil, e como ela poderia agregar na visualização de melhoria de processos e planejamento das equipes.

Após essa etapa, é avaliada a funcionalidade, focando nas informações que são trazidas e realizando uma verificação englobando os requisitos funcionais. A última etapa consiste em considerações do usuário, como críticas e sugestões.

6.2 Planejamento da avaliação

O processo de avaliação foi divido em duas partes. A primeira com enfoque na gerência do laboratório Bridge, aplicando a avaliação da utilidade da ferramenta com o Gerente de Desenvolvimento. Na segunda parte foi realizada a avaliação com três equipes ágeis, sendo duas dessas equipes que foram as fornecedoras de requisitos.

As equipes que realizaram a avaliação foram a Supernova, Royal Flush e Legacy. Sendo a Supernova e Royal Flush as fornecedoras de requisitos que estavam por dentro do andamento da ferramenta. Com o intuito de realizar também a avaliação com uma equipe que não tinha ainda visto até então os requisitos da ferramenta, podendo conceber uma diferente visão, foi conversado com mais uma equipe ágil para realizar a avaliação, a equipe Legacy.

A função de cada participante que respondeu o questionário, assim como seu tempo de experiência e área de formação, está descrita na Tabela 8.

Para os dois tipos de avaliação abordados, foi realizado o mesmo método de aplicação. Primeiro foi realizado uma conversa de 10 a 15 minutos com o usuário alvo da aplicação da avaliação, nessa conversa foi apresentada a ideia geral do trabalho e os objetivos da ferramenta, também abordando as funcionalidades e esclarecendo dúvidas.

A segunda parte do método de avaliação consistia na aplicação de um *survey* (Sauro, 2015), técnica que oferece diversas vantagens, tais como permitir o anonimato das respostas, a concentração de dados em um só lugar e possibilitar que os usuários respondam no momento que lhes pareça mais apropriado. Portanto, após a breve apresentação dos objetivos da ferramenta, foi aplicado um *survey* via Google forms[14], e deixado os usuários a vontade para respondê-lo.

Tabela 8: Descrição dos participantes das equipes ágeis que responderam a avaliação

Função exercida	**Tempo exercendo função**	**Curso**	**Formação**	**Equipe**
Desenvolvimento	5 anos	Sistemas de Informação	Completo	Royal Flush
Desenvolvimento	2 anos	Sistemas de Informação	Completo	
Tester	5 anos	Sistemas de Informação	Cursando	
Tester	2 anos	Sistemas de Informação	Cursando	
Análise	3 anos	Sistemas de Informação	Completo	
Desenvolvimento	4 anos	Ciências da Computação	Completo	Supernova
Desenvolvimento	1 ano	Sistemas de Informação	Cursando	
Tester	4 anos	Sistemas de Informação	Cursando	
Tester	3 anos	Sistemas de Informação	Cursando	
Análise	6 anos	Sistemas de Informação	Completo	
Desenvolvimento	6 anos	Ciências da Computação	Completo	Legacy
Desenvolvimento	4 anos	Ciências da Computação	Cursando	
Tester	2 anos	Ciências da Computação	Cursando	
Análise	4 anos	Ciências da Computação	Cursando	
Designer	4 anos	Design	Completo	

Fonte: Elaborado pela autora

[14] https://www.google.com/forms/about/

6.3 Aplicação da avaliação

Como citado anteriormente na seção 6.2, a primeira parte da avaliação consistiu em uma pequena reunião com os usuários. Nessa reunião foram explicados os objetivos principais do trabalho com enfoque na concepção da ferramenta, abordando de maneira geral os requisitos funcionais e brevemente explicando as funcionalidades do sistema.

Na segunda parte foi aplicado um *survey* via Google forms. Foram criados dois forms separados por: questionário para equipes ágeis (Apêndice B) e questionário para gerência (Apêndice C). O primeiro consiste em perguntas levando em consideração as características tanto de utilidade quanto funcionalidade da ferramenta, já o segundo forms tem enfoque apenas na característica de utilidade.

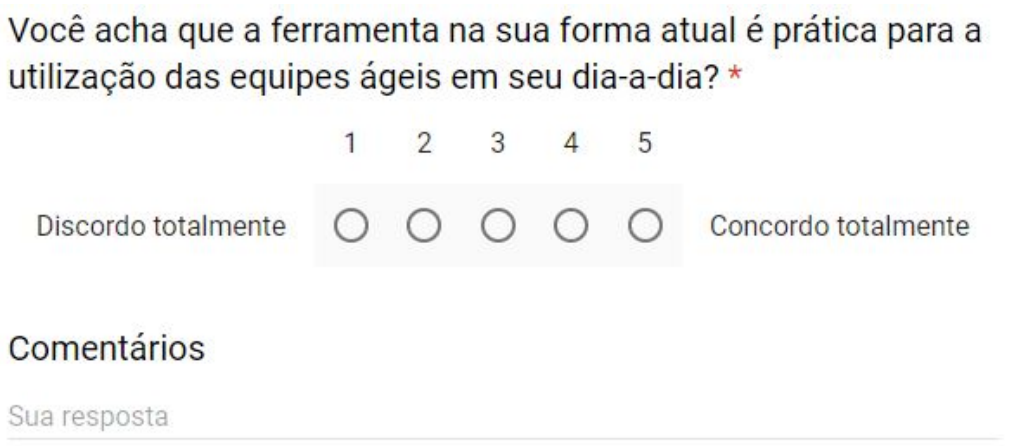
Como pode-se observar na tabela 8, o questionário é dividido em perguntas sobre a utilidade e funcionalidade, tendo uma última seção comum para críticas e sugestões.

As perguntas sobre utilidade tem como objetivo questionar as equipes e a gerência se a ferramenta pode trazer benefícios sendo implantada no processo de trabalho, assim como se as informações contidas oferecem a possibilidade de traçar melhorias no planejamento das equipes.

Na segunda parte do questionário, com perguntas voltadas à funcionalidade do sistema, são abordadas questões relacionadas aos requisitos funcionais levantados pelos fornecedores, à consistência das informações apresentadas e também se as funcionalidades presentes são suficientes aos objetivos propostos.

Com relação às respostas, exceto as duas últimas perguntas da tabela 8, é apresentado uma escala linear a partir de “Discordo totalmente” até “Concordo totalmente”. Abaixo de todas as perguntas existe um campo de texto para o usuário submeter comentários com relação à sua resposta.

Figura 40: Exemplo de pergunta sobre a utilidade da ferramenta



Fonte: Formulário (via Google forms*)* elaborado pela autora

A penúltima pergunta diz respeito à verificação de erros na aplicação, portanto a resposta é apenas a seleção entre “Sim” ou “Não”. Caso a resposta seja “Sim”, existe um campo de texto para o usuário descrever sobre o *bug* encontrado.

Já a última questão apresenta apenas um campo de texto para o usuário submeter críticas, sugestões de melhoria, dúvidas ou qualquer comentário extra que não foi abordado nas outras questões.

A tabela 8 abaixo apresenta todas as perguntas existentes no *survey,* separadas por papel do usuário (equipes ágeis ou gerência) e por característica.

Tabela 9: Questionário de avaliação da ferramenta

Característica	**Questionário**	
	Questionário para equipes ágeis	**Questionário para gerência**
Utilidade	Você considera a ferramenta útil para o autogerenciamento de equipes em ambientes de desenvolvimento ágil?	
	Você acha que a ferramenta na sua forma atual é prática para a utilização das equipes ágeis em seu dia-a-dia?	
	Você acha que a ferramenta oferece informações suficientes para apoio ao desenvolvimento do método Kanban dentro de equipes ágeis?	
	O conjunto de informações fornecidas pelo sistema é suficiente para realizar o planejamento, como estimativa de entregas e partição de tarefas em equipes ágeis?	
		Como gerente de desenvolvimento, você considera a ferramenta útil para mensurar o desempenho das equipes ágeis?

		Como gerente de desenvolvimento, você considera a ferramenta útil para diagnosticar possíveis gargalos ou oportunidades de melhoria no fluxo de trabalho de equipes em ambiente de desenvolvimento ágil?
		Possui algum comentário, melhoria ou sugestão sobre a ferramenta desenvolvida?
Funcionalidade	De acordo com as diretrizes do método Kanban utilizados na sua equipe ágil, você acha que as informações disponibilizadas são suficientes?	
	Você considera que o sistema disponibiliza informações suficientes para a aferição de quantidade de tarefas em cada coluna do *board*?	
	Você considera que a ferramenta possui funcionalidades suficientes para automatizar um processo que hoje é manual?	
	Você acha que a ferramenta possui informações suficientes para a observação do *lead time* das tarefas?	
	Você considera que a ferramenta indica corretamente as tarefas que se encontram no estado de *work in progress*?	
	Você observou algum erro (bug) em relação a funcionalidade da ferramenta?	
Crítica/Sugestão	Possui algum comentário, melhoria ou sugestão sobre a ferramenta desenvolvida?	

Fonte: Elaborado pela autora

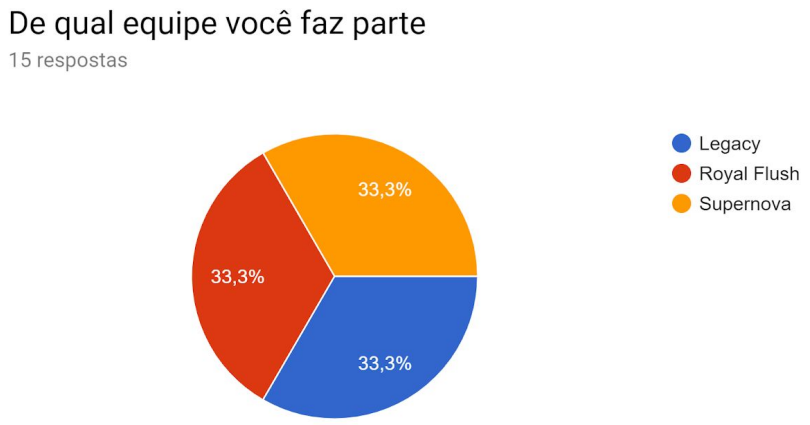
6.4 Análise dos resultados

Nas seções a seguir é apresentada a análise dos resultados obtidos pela aplicação no *survey*, separados por equipes ágeis do laboratório e gerência de desenvolvimento.

6.4.1 Equipes ágeis

A avaliação feita com três equipes ágeis contou com a participação de quinze usuários no total (Figura 38), sendo cinco participantes de cada equipe. Os usuários assumem diferentes funções dentro das equipes, como análise, desenvolvimento de software e teste de qualidade.

Figura 41: Participantes por equipe que responderam o questionário



Fonte: Resultado do Google forms elaborado pela autora

Nas seções abaixo 6.4.1.1 e 6.4.1.2 será feita uma análise das respostas separadas por características. As respostas são classificadas de 1 a 5, sendo o 1 “Discordo totalmente” até 5 “Concordo totalmente”.

6.4.1.1 Sobre a utilidade

Pergunta 1: Você considera a ferramenta útil para o autogerenciamento de sua equipe no contexto de desenvolvimento ágil?

No contexto de desenvolvimento ágil, 93,3% dos usuários concordam totalmente (nível 5) que a ferramenta é útil para o autogerenciamento de sua equipe, enquanto 6,7% escolheram 4. Na tabela 9 a seguir apresentados os comentários dos usuários sobre a pergunta 1.

Tabela 10: Comentários sobre a pergunta 1 do questionário de equipes ágeis

Utilizamos o Kanban dentro de nossa equipe, porém fazemos a coleta de métricas tudo manualmente. A ferramenta automatiza alguns processos da equipe
Ao coletar e exibir dados relevantes de forma rápida, a ferramenta demonstrou ser capaz de tornar o nosso processo de autogerenciamento mais eficiente.
Poupa tempo ter uma ferramenta que já retira essas informações diretamente do github
Achei a funcionalidade de ver o histórico do CFD muito interessante. Acho importante pensar no cenário dos times usarem um project a nível de organização, não de repositório (por exemplo, ao invés do project ficar no repositório PEC, ele fica na organização laboratório bridge). Alguns times compilam os dados de sua sprint considerando todos os repos que trabalharam.
A ferramenta é muito útil para a equipe, pois ela dá um apoio no que diz respeito à coleta e síntese das principais informações sobre as tarefas da equipe
É útil para gerar indicadores do desenvolvimento das tarefas

Fonte: Respostas obtidas do Google forms

Pergunta 2: Você acha que a ferramenta na sua forma atual é prática para a sua equipe em seu dia-a-dia?

Em relação ao uso no dia-a-dia de uma equipe ágil, 80% responderam que concordam totalmente que a ferramenta é prática, enquanto 20% (nível 4) concordam com a praticidade da aplicação.

Pergunta 3: Você acha que a ferramenta oferece informações suficientes para apoio ao desenvolvimento do método Kanban dentro da sua equipe?

Sobre o apoio ao desenvolvimento do método Kanban em sua equipe, 40% concordam totalmente que a ferramenta fornece informações suficientes, enquanto 53,3% concordam e 6,7% nem concorda e nem discorda . Na tabela 10 abaixo é apresentado os comentários dos usuários sobre a pergunta 3.

Tabela 11: Comentários sobre a pergunta 3 do questionário de equipes ágeis

Nossa equipe está bem madura no Kanban, estamos coletando mais dados que a ferramenta ainda não tem.
Adicionar um valor do working in progress pra semana filtrada e também um gráfico para avaliar o coeficiente de variação também são informações legais para o Kanban, mas acho que as principais já existem que é o cfd e o gráfico de lead time
Poderia ter a data de criação e de fechamento da issue na lista de issues fechadas.
Ele se complementa com a planilha. Inclusive acho interessante essa ferramenta absorver a planilha e ser integrada no Meu Bridge.

Ainda faltam algumas poucas informações, porém ela facilita muito o trabalho da equipe.

Fonte: Respostas obtidas do Google forms

Pergunta 4: O conjunto de informações fornecidas pelo sistema é suficiente para realizar o planejamento, como estimativa de entregas e partição de tarefas na sua equipe?

Com relação ao planejamento, 20% dos usuários concordam totalmente que o conjunto de informações fornecidas é suficiente, 73,3% concordam e 6,7% nem concordam nem discordam.

A partir de uma análise mais detalhada das respostas apresentadas acima é possível ter algumas ideias sobre a utilidade do sistema:

- A ferramenta pode ser útil no ambiente de desenvolvimento, pois maioria dos usuários concordam com isso. Principalmente no que diz respeito a autogerenciamento, planejamento e observações de possíveis melhorias no fluxo, relacionados às perguntas 1 e 4.
- Os usuários acreditam que a ferramenta seria útil na implantação do método Kanban na equipe, portanto é possível inferir que a ferramenta tem alguma utilidade nesse contexto. Mesmo alguns usuários comentando que necessitam de mais informações do que as que estão disponíveis, obteve-se alguns comentários positivos no que diz respeito a coleta de informações para o desenvolvimento do método Kanban, como pode-se observar nas tabelas 9 e 10.
- É possível que a ferramenta seja útil e prática, pois além da utilidade, a maioria dos usuários concorda totalmente na praticidade da ferramenta, como é possível perceber nas respostas da pergunta 2.

6.4.1.2 Sobre a funcionalidade

Pergunta 5: De acordo com as diretrizes do método Kanban utilizado na sua equipe ágil, você acha que as informações disponibilizadas são suficientes?

Com relação às diretrizes do método Kanban, 20% dos usuários concordam totalmente que as informações disponibilizadas são suficientes, 73,3% concordam e 6,7% nem concordam nem discordam.

Pergunta 6: Você considera que o sistema disponibiliza informações suficientes para a aferição de quantidade de tarefas em cada coluna do *board*?

Sobre as tarefas em cada coluna do *board* da equipe, 80% concorda totalmente que as informações disponibilizadas são suficientes, 13,3% apenas concorda e 6,7% nem concorda e nem discorda.

Pergunta 7: Você considera que a ferramenta possui funcionalidades suficientes para automatizar um processo que hoje é manual?

60% dos usuários concordam totalmente que com as funcionalidades da ferramenta é possível automatizar algum processo manual, enquanto 40% concordam. Os comentários abaixo na tabela 11 dizem respeito à quais processos manuais a ferramenta poderia auxiliar.

Tabela 12: Comentários sobre a pergunta 7 do questionário de equipes ágeis

Contagem do Lead time, Contagem de quantas tarefas tem em cada coluna
Coleta do lead time médio da equipe
A ferramenta nos facilita a visibilidade do lead time de cada issue, que dentro de minha equipe é coletado semanalmente, toda segunda-feira, de forma manual. Fazemos essa coleta para avaliar as issues individualmente e verificar se alguma passou do tempo médio da equipe, para tentar identificar problemas. Nesse ponto a ferramenta seria um facilitador para a equipe. O gráfico do CFD também permite a detecção de anomalias no nosso Kanban e observar os gargalos de certas áreas, como desenvolvimento e testes.
Melhoria do processo em geral; Identificação de gargalos; Geração automática dos gráficos de lead time e CFD.
A construção do gráfico de lead time e o CFD
O cálculo do CFD e do lead time.
Atualmente alguém precisa entrar no histórico de issues do GitHub e verificar quais tarefas foram fechadas e quanto tempo foi necessário para fechá-las, com a ferramenta, este processo fica automático
Preenchimento semanal da planilha do Kanban utilizada pela equipe.
Passar os dados do board na mão
Obtenção dos dados das issues para o Kanban

Fonte: Respostas obtidas do Google forms

Pergunta 8: Você acha que a ferramenta possui informações suficientes para a observação do *lead time* das tarefas?

Com relação ao *lead time* das tarefas, 80% concorda totalmente que a ferramenta possui informações suficiente, enquanto 20% apenas concorda.

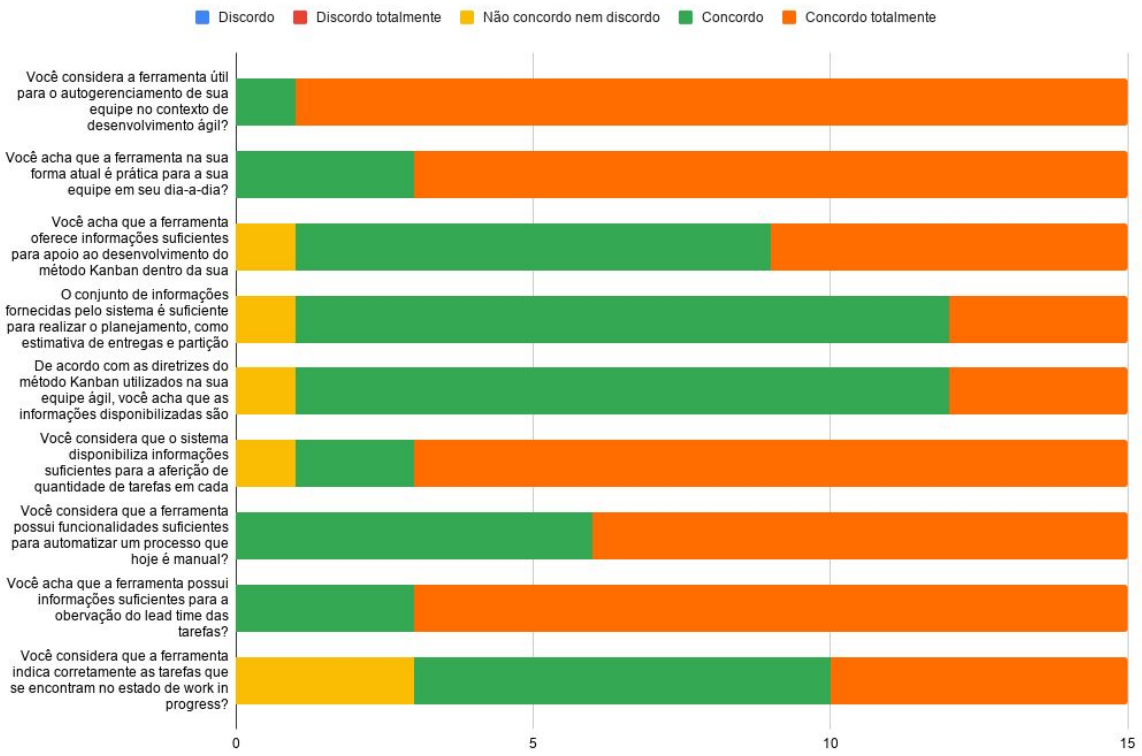
Pergunta 9: Você considera que a ferramenta indica corretamente as tarefas que se encontram no estado de *work in progress*?

Com relação ao estado das tarefas, 33,3% concorda totalmente que a ferramenta indica as tarefas que estão em *work in progress*, 46,7% apenas concorda e 20% nem concorda e nem discorda.

A partir de uma análise mais detalhada das respostas sobre a funcionalidade da ferramenta, é possível ter algumas ideias sobre o sistema:

- A maioria dos usuários concorda que a ferramenta possui funcionalidades que permitem a aferição de métricas como *lead time,* tarefas em *work in progress* e quantidade de cada tarefa em cada coluna do *board.* Conforme pode-se observar nas respostas das perguntas 6, 8 e 9. Portanto, é possível que o sistema possa auxiliar na coleta dessas métricas ágeis.
- A aplicação pode fornecer a automatização de processos manuais, pois a maioria dos usuários concorda totalmente que a ferramenta oferece funcionalidades para isso, conforme respostas da pergunta 7. Dentre os processos citados pelos usuários (tabela 11), estão a contagem de tarefas do *board*, a geração do gráfico de cumulative flow diagram (CFD) e a contagem do *lead time*.
- É possível inferir que apesar da ferramenta obter a maior parte das métricas necessárias para a implantação do Kanban nas equipes, ainda faltam algumas informações, pois a maior parte dos usuários apenas concorda com isso, conforme respostas à pergunta 5. Algumas equipes que tem o processos mais amadurecido sentem falta de algumas informações, como pode-se perceber na tabela 10.

Figura 42 - Respostas do questionário de avaliação



Fonte: Elaborado pela autora

6.4.2 Gerência de desenvolvimento

Com relação às três primeiras perguntas (tabela 8), o Gerente de Desenvolvimento entrevistado concorda sobre a utilidade, praticidade e coleta de métricas utilizadas no Kanban fornecidas na ferramenta.

Na resposta da quarta pergunta, o gerente nem concorda e nem discorda que as informações fornecidas sejam suficientes para realizar o planejamento como estimativa de entrega e partição de tarefas. Ele comenta em um trecho que, “Planejamento e estimativas dependem mais de uma mudança cultural significativa na equipe do que acesso às estatísticas”, além disso destaca que as informações fornecidas ajudam as equipes a saber se estão em um caminho certo.

O gerente discorda que a ferramenta seja útil para mensurar o desempenho das equipes ágeis (pergunta 5). Ele conclui que desempenho é algo relativo, pois depende da forma que as equipes fragmentam as tarefas, podendo às vezes poucas tarefas agregar mais valor que muitas.

No papel de gerente de desenvolvimento, ele concorda totalmente que a ferramenta seja útil para diagnosticar possíveis gargalos ou oportunidades de melhoria no fluxo de trabalho (pergunta 6).

Na última pergunta em aberto para críticas e sugestões, o gerente de desenvolvimento destaca que a ferramenta é uma ideia muito boa e promissora.

A partir de uma análise mais detalhada das respostas da utilidade da ferramenta a partir da gerência de desenvolvimento, é possível inferir algumas ideias:

- A ferramenta tem utilidade e é prática para equipes ágeis, assim como apoia o desenvolvimento do método Kanban, como demonstra a resposta das três primeiras perguntas.
- A ferramenta não tem como um dos objetivos medir o desempenho de equipes, por se tratar de algo que possui muitos fatores envolvidos além de coleta de dados, como pode-se observar na resposta da pergunta 5.
- Através da ferramenta, é possível identificar gargalos e melhorias no fluxo de trabalho de equipes ágeis, conforme resposta da pergunta 6.

6.5 Considerações finais

Neste capítulo foi apresentada a avaliação da ferramenta de coleta de métricas ágeis integrada à plataforma GitHub. Essa avaliação foi realizada com dois enfoques, um na utilidade e outro na funcionalidade da aplicação. Os usuários foram divididos em equipes ágeis e gerência de desenvolvimento para um melhor direcionamento do questionário.

Através das ideias inferidas a partir da análise das respostas, é possível levantar indícios que a ferramenta tenha cumprido seu papel de ser útil no contexto de ambientes de desenvolvimento ágil, e de que a ferramenta traz as principais métricas ágeis utilizadas no processo das equipes.

7 CONCLUSÃO

Neste trabalho são apresentadas a análise e implementação de uma ferramenta que possibilita a coleta de métricas ágeis por meio da integração com a plataforma GitHub, com enfoque nos indicadores necessários dentro do processo de software de equipes em ambientes de desenvolvimento ágil.

Primeiramente realizou-se um estudo da literatura na área de avaliação de processos de software. Esse estudo envolveu os conceitos de métricas ágeis em ambientes de desenvolvimento ágil e também os assuntos pertinentes ao desenvolvimento do trabalho relacionados à plataforma GitHub.

Nesse contexto de avaliação automatizada de processos em equipes ágeis, o estado da arte elaborado contou com o estudo sobre os trabalhos que utilizam ferramentas para obtenção desses resultados. Foi possível obter informações sobre como modelos automatizados vem sendo utilizados na prática para realização de avaliações de processos software, e também de implantação de algumas ferramentas Web.

Na sequência, foi realizado o levantamento e análise dos requisitos, por meio de entrevistas com equipes ágeis e com colaboradores responsáveis pela implantação do método Kanban e da *Objective and Key Results* (OKR) na organização. Essas entrevistas tiveram como objetivo a compreensão de métricas ágeis necessárias na identificação de melhorias no fluxo de desenvolvimento. Como resultado dessas entrevistas, foram levantados os requisitos funcionais.

A partir dos requisitos funcionais, foram criados os protótipos de tela da ferramenta. Nessa etapa também foi concebida a modelagem do sistema, incluindo as tecnologias a serem utilizadas, linguagem de programação e a arquitetura das camadas.

Após definida a modelagem, o sistema foi implementado, coletando dados de equipes ágeis de maneira automatizada com integração à plataforma GitHub. A partir de uma versão estável, equipes ágeis e a gerência de desenvolvimento utilizaram e avaliaram a ferramenta.

Por fim, com os resultados da avaliação da utilidade e funcionalidade da ferramenta, conclui-se que os membros de equipes ágeis e da gerência mostram uma visão positiva sobre os objetivos da ferramenta. Percebe-se pelas respostas e comentários coletados, que os usuários acreditam que a ferramenta acrescenta na identificação de melhorias no fluxo da equipe, assim como possibilita uma praticidade ao obter métricas ágeis de forma automática.

Desta forma, entende-se que o objetivo geral deste trabalho de realizar a análise e implementação de métricas ágeis em equipes através de dados coletados de forma automatizada foi atingido. Além disso, a ferramenta possibilita uma visão dos indicadores utilizados de acordo com a necessidade dos usuários dentro de ambientes de desenvolvimento ágil.

7.1 Trabalhos futuros

Os trabalhos futuros para a ferramenta de coleta de métricas ágeis com integração à plataforma GitHub foram identificados com base no desenvolvimento e resultado da avaliação da ferramenta.

- Configuração do estado inicial de início da *sprint* de cada equipe ágil. Ou seja, existência de um módulo em que o líder da equipe possa definir a partir de qual coluna uma tarefa começa a entrar em *work in progress*.
- Implementação de um módulo de autenticação a partir da conta de usuário do Google, sendo possível realizar a verificação com o e-mail da organização.
- Realizar ajustes para que o sistema funcione não apenas a nível de repositório, mas sim a nível de organização na plataforma GitHub.
- Permitir a generalização da ferramenta rodar em qualquer repositório do GitHub, desde que esse tenha *board* com informações como *issues.*

REFERÊNCIAS

ALBINO, Raphael. **Métricas Ágeis - Obtenha melhores resultados em sua equipe**. Impresso e PDF: 978-85-5519-276-0. Ano publicação: 2017.

ANACLETO, Alessandra; WANGENHEIM, C. G.; SALVIANO, C. Um método de avaliação de processos de software em micro e pequenas empresas. **SBQS-Simpósio Brasileiro de Qualidade de Software. Porto Alegre**, 2005.

ANDERSON, David J.; CARMICHAEL, Andy. **Essential Kanban condensed**. Blue Hole Press, 2016. Acesso em: agosto de 2019.

BATTISTI, Iara Denise Endruweit; BATTISTI, Gerson. Métodos estatísticos. 2008. Acesso em: Outubro de 2019.

BOEG, Jesper. **Kanban em 10 passos**. Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media, 2010.

Bold Design System. Disponível em: https://bold.bridge.ufsc.br/ Acesso em: agosto de 2019.

BR, MPS. **MPS. BR–Melhoria de Processo do Software Brasileiro**. 2011.

Castro, Felipe (2015). **Agile Goal Setting with OKR - Objectives and Key Results**, InfoQ, Disponível em: https://www.infoq.com/articles/agile-goals-okr, 2015. [Acesso em: 12 de novembro de 2018]

COHN, M. **Agile Estimating and Planning**. Massachusetts: Pearson Education, 2006.

Deborah Hartmann and Robin Dymond. **Appropriate agile measurements: Using metrics and diagnostics to deliver business value**. In Agile 2006 Conference, pages 126–131, 2006

GitHub Articles (About Repository). Disponível em: https://help.github.com/articles/about-repositories/ [Acesso em: 11 de novembro de 2018]

GitHub documentation. Disponível em: https://guides.github.com/ [Acesso em: 3 de novembro de 2018]

GitHub Glossário. Disponível em: https://help.github.com/articles/github-glossary/ [Acesso em: 11 de novembro de 2018]

GitHub Issues. Disponível em: https://guides.github.com/features/issues/, atualizado em 2014 [Acesso em: 11 de novembro de 2018]

IEEE standard glossary of software engineering terminology, 1990. IEEE Std 610.12.

Inthurn, Cândida. **Qualidade & Teste de Software**. 2. ed. Florianópolis: Visual Books Editora, 2001. 108 p.

ISO/IEC/IEEE 15939:2017(E) **International Standard - Systems and software engineering Measurement process**.

ISO, I. S. O. iec/ieee international standard-systems and software engineering–vocabulary. ISO/IEC/IEEE 24765: 2017 (E), 2017.

ISO/IEC, 2008] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION. ISO/IEC 12207:2008 S**ystems and software engineering — Software life cycle processes**, Geneve: ISO, 2008.

JARVINEN, J. Measurement based continuous assessment of software engineering processes. **VTT PUBLICATIONS**, v. 4, n. 2, p. 6, 2000.

John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. **Practical Software Measurement: Objective Information for Decision Makers.** Addison-Wesley Professional, 2002

Laboratório Bridge Portfólio. Disponível em: https://bridge.ufsc.br/portfolio Acesso em: 20 de novembro de 2018.

LARMAN, C., **Agile and Iterative Development: A Manager's Guide**, Addison-Wesley Professional, 2003.

LEAL, Stéphanie. **Um deployment package de implementação dos processos do perfil básico da norma ISO/IEC 29110.** Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2016.

LUNA, Alexandre JH de O. et al. Uma Abordagem para o Gerenciamento Estratégico Ágil em Saúde utilizando PES, OKR e MAnGve. **Revista Eletrônica da Estácio Recife**, v. 3, n. 2, 2017.

Manifesto Ágil. Disponível em http://www.manifestoagil.com.br/ Acesso em: 30 de maio de 2018.

MONTONI, Mariano et al. **Uma abordagem para Condução de Iniciativas de Melhoria de Processos de Software**. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2008/018.pdf Acesso em: 15 de abril de 2018.

NIVEN, Paul R.; LAMORTE, Ben. **Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs**. John Wiley & Sons, 2016.

PEINADO, JURANDIR; AGUIAR, G. **Compreendendo o Kanban: um ensino interativo ilustrado**. Revista DaVinci. Curitiba–PR, v. 4, n. 1, p. 133-146, 2007.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software** - 8ª Edição. McGraw Hill Brasil, 2016.

RAO, Kuda Nageswara; NAIDU, G. Kavita; CHAKKA, Praneeth. A study of the agile software development methods, applicability and implications in industry. **International Journal of Software Engineering and its applications**, v. 5, n. 2, p. 35-45, 2011.

RISING, Linda; JANOFF, Norman S. The Scrum software development process for small teams. **IEEE software**, v. 17, n. 4, p. 26-32, 2000.

ROCHA, Lais MA; SILVA, Thiago Henrique P.; MORO, Mirella M. **Análise da Contribuição para Código entre Repositórios do GitHub**. 2016. Disponível em: http://homepages.dcc.ufmg.br/~mirella/pdf/2016.SBBDshort.Rocha.pdf Acesso em: 30 de maio de 2018.

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, v. 139, 2007.

SAURO, Jeff. Using Surveys to Measure the User Experience. 2015. Disponível em: https://measuringu.com/survey-ux/. Acesso em: Outubro de 2019.

SCAFF, Mariana. **Um Estudo de Caso de Melhoria de Processos do Nível G para o Nível F do Modelo MPS Para Software**. Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2016.

SCHWABER, Ken; BEEDLE, Mike. **Agile Software Development with SCRUM**. Prentice Hall, 2002.

SOMMERVILLE, Ian; ARAKAKI, Reginaldo; MELNIKOFF, Selma Shin Shimizu. **Engenharia de software** - 9ª Edição - Pearson Prentice Hall, 2011.

THUNG, Ferdian et al. Network structure of social coding in GitHub. In: **Software maintenance and reengineering (csmr), 2013 17th european conference on**. IEEE, 2013. p. 323-326.

WAZLAWICK, Raul. **Engenharia de software: conceitos e práticas**. Elsevier Brasil, 2013.

Zwirtes, Augusto. **UM MODELO DE AVALIAÇÃO SEMI-AUTOMATIZADA DE PROCESSOS DE SOFTWARE ALINHADO AO BPMN.** Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2018.

APÊNDICE A: TERMO DE CONCORDÂNCIA

DECLARAÇÃO DE CONCORDÂNCIA COM AS CONDIÇÕES PARA O DESENVOLVIMENTO DO TCC NA INSTITUIÇÃO

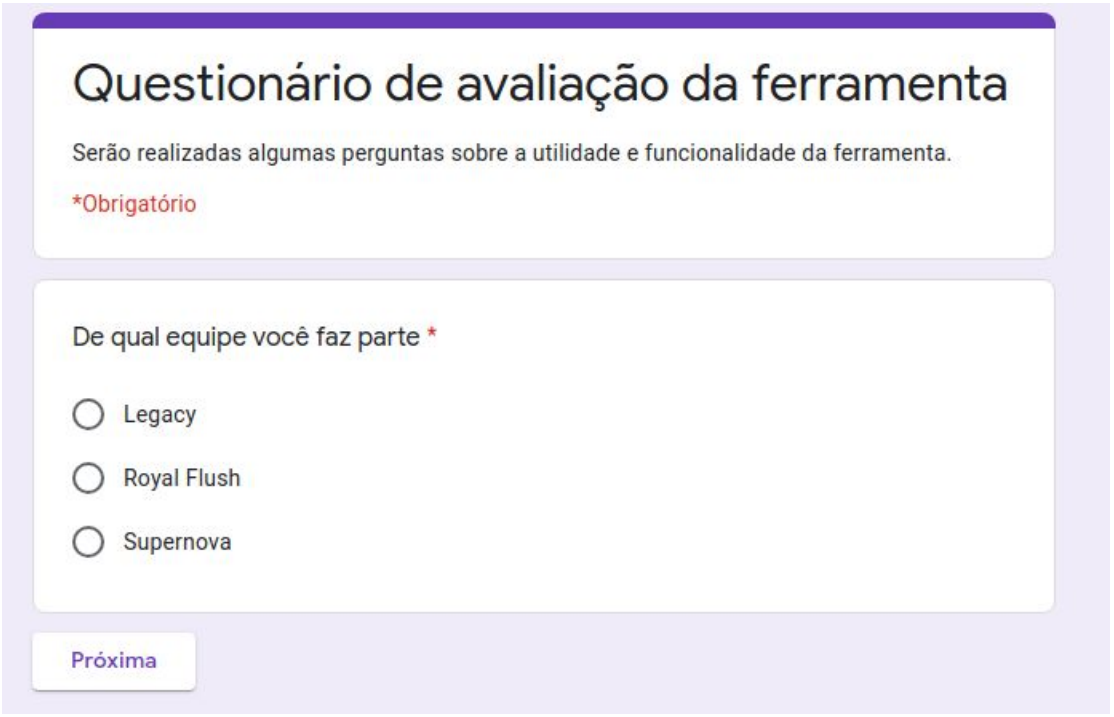
Declaro estar ciente das premissas para a realização de Trabalhos de Conclusão de Curso (TCC) de Ciência da Computação da UFSC, particularmente da necessidade do TCC envolver a aplicação e caso de estudo dentro da instituição. Sendo a descrição dos métodos, processos utilizados e resultados entregue integralmente, como parte integrante do relatório final do TCC. Também declaro estar ciente que o código fonte e/ou documentação completa a ser entregue condiz apenas com o TCC realizado pela acadêmica. Ciente dessa condição básica, declaro estar de acordo com a realização do TCC identificado pelos dados apresentados a seguir.

Instituição	Laboratório Bridge
Nome do responsável	Prof. Dr. Raul Sidnei Wazlawick
Cargo/Função	Coordenador Geral
Fone de contato	(48) 37215991
Acadêmica	Jacyara Bosse
Título do trabalho	Ferramenta para avaliação semi-automatizada de processos de software
Curso	Ciência da Computação - INE/UFSC

Florianópolis, 10 de outubro de 2018

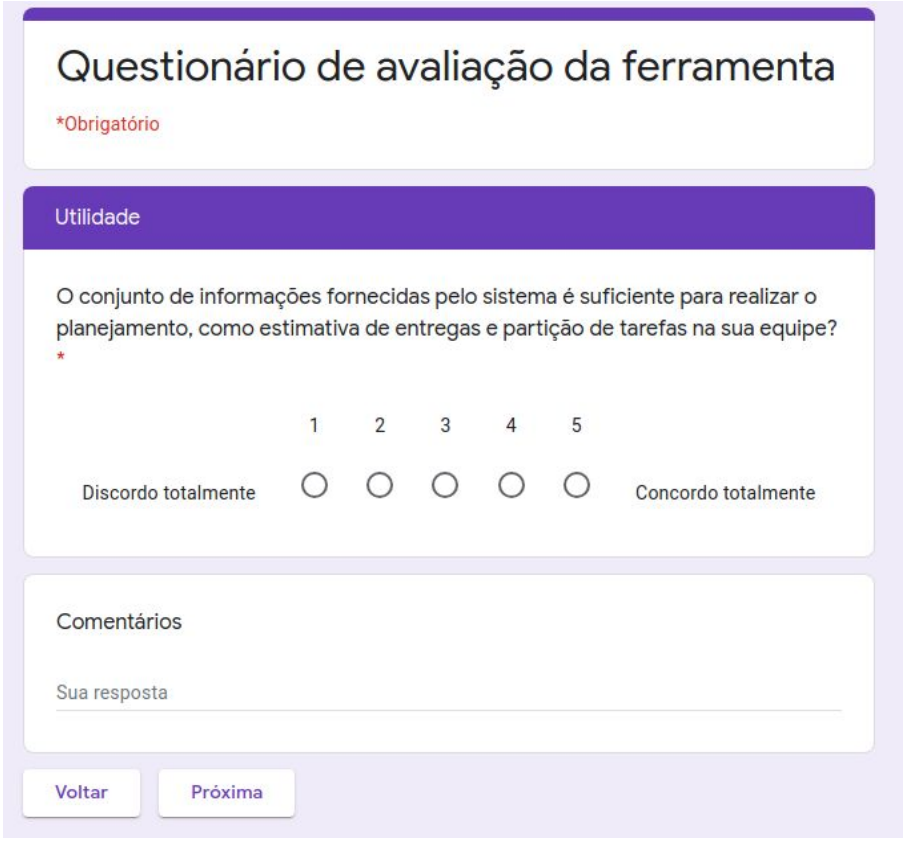
Assinatura do responsável: Prof. Raul Sidnei Wazl... Coord. Geral

APÊNDICE B: AVALIAÇÃO PARA EQUIPES ÁGEIS




Questionário de avaliação da ferramenta
*Obrigatório
Utilidade
Você acha que a ferramenta na sua forma atual é prática para a sua equipe em seu dia-a-dia? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário de avaliação da ferramenta
*Obrigatório
Utilidade
Você acha que a ferramenta oferece informações suficientes para apoio ao desenvolvimento do método Kanban dentro da sua equipe? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário de avaliação da ferramenta
*Obrigatório
Utilidade
O conjunto de informações fornecidas pelo sistema é suficiente para realizar o planejamento, como estimativa de entregas e partição de tarefas na sua equipe? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
De acordo com as diretrizes do método Kanban utilizados na sua equipe ágil, você acha que as informações disponibilizadas são suficientes? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima

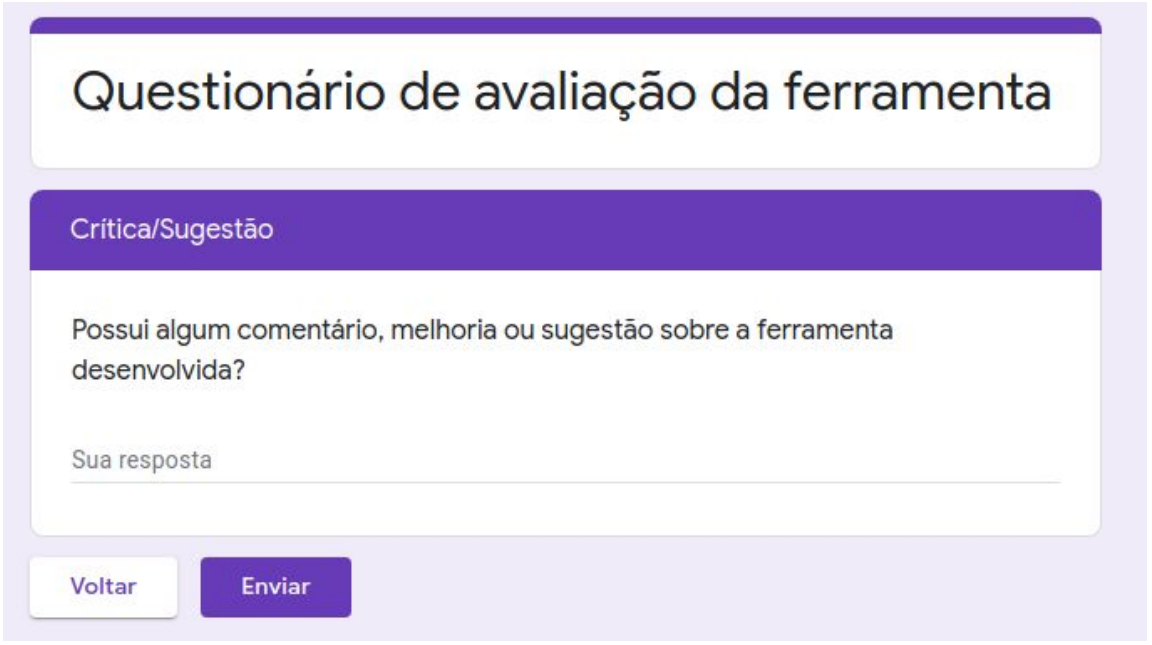

Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
Você considera que o sistema disponibiliza informações suficientes para a aferição de quantidade de tarefas em cada coluna do board? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
Você considera que a ferramenta possui funcionalidades suficientes para automatizar um processo que hoje é manual? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Se sim, quais?
Sua resposta
Comentários
Sua resposta
Voltar
Próxima

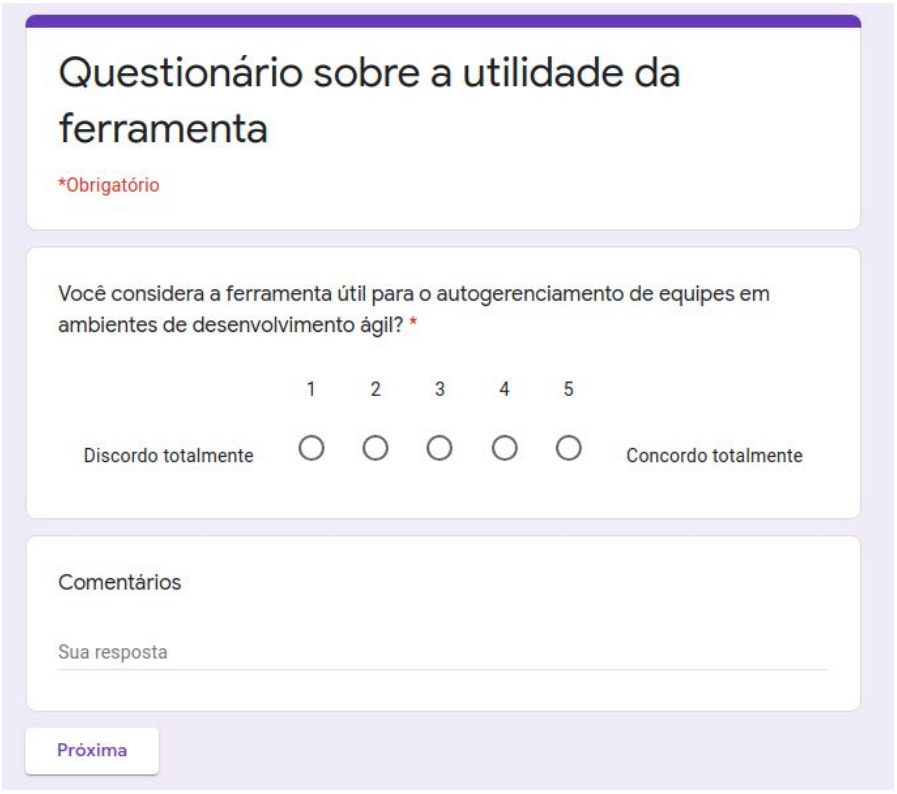

Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
Você acha que a ferramenta possui informações suficientes para a obervação do lead time das tarefas? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima

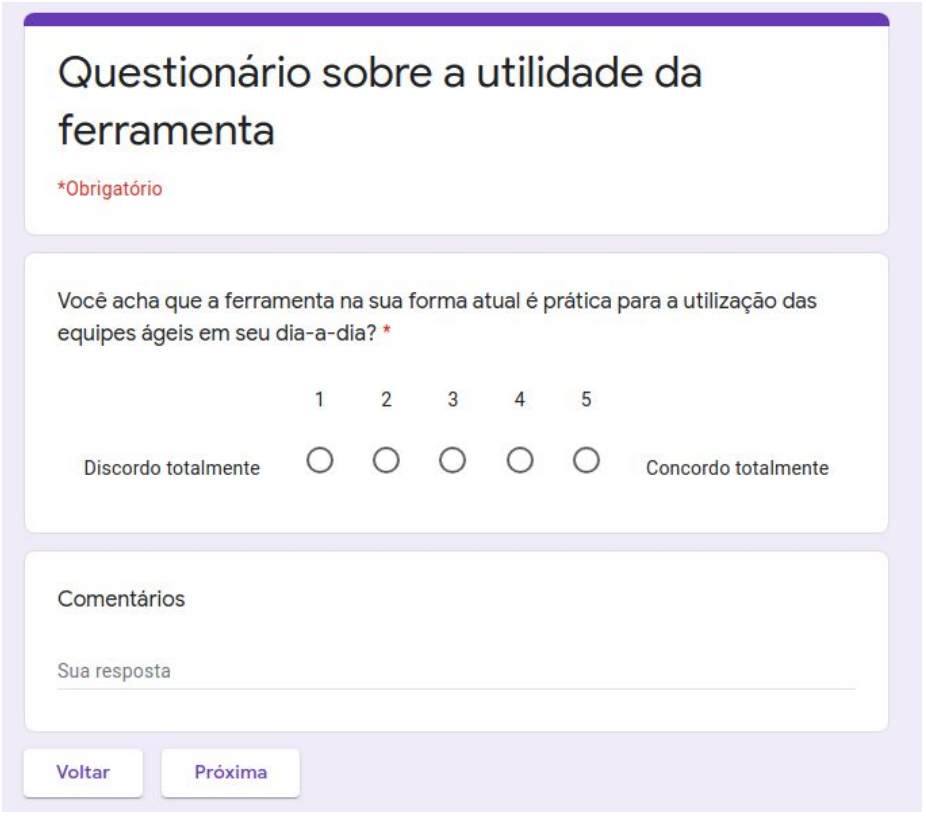

Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
Você considera que a ferramenta indica corretamente as tarefas que se encontram no estado de work in progress? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima

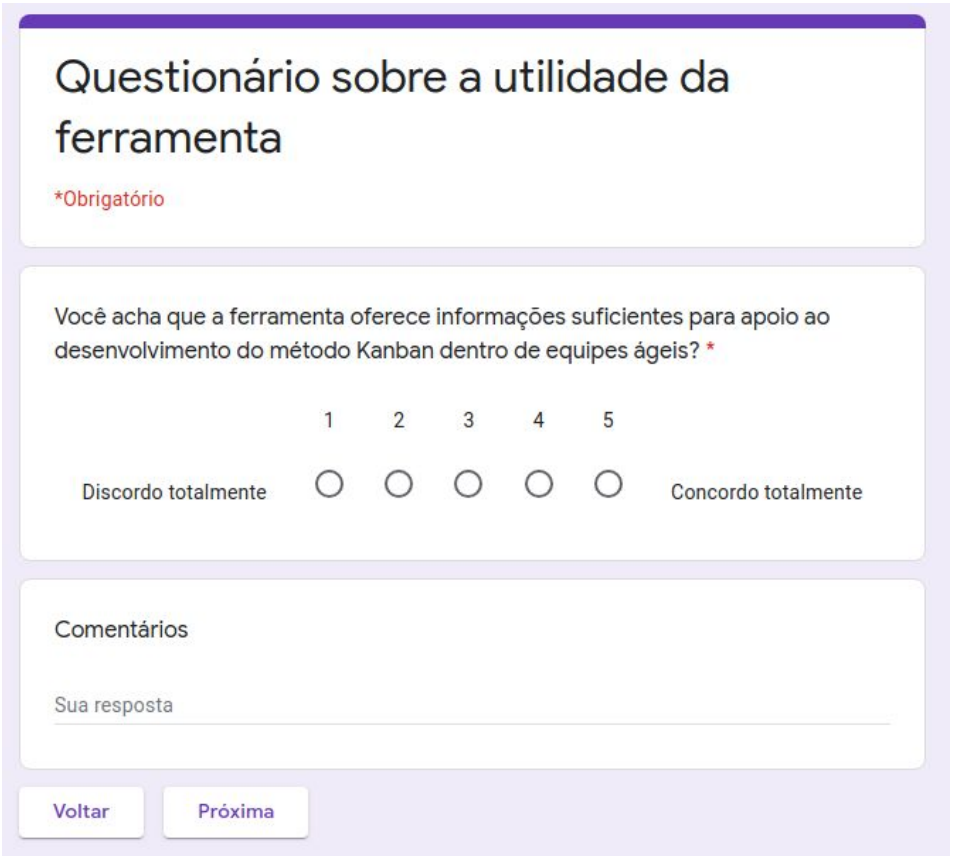

Questionário de avaliação da ferramenta
*Obrigatório
Funcionalidade
Você observou algum erro (bug) em relação a funcionalidade da ferramenta? *
Sim
Não
Se sim, poderia descrever?
Sua resposta
Voltar
Próxima


Questionário de avaliação da ferramenta
Crítica/Sugestão
Possui algum comentário, melhoria ou sugestão sobre a ferramenta desenvolvida?
Sua resposta
Voltar
Enviar

APÊNDICE C: AVALIAÇÃO PARA GERÊNCIA DE DESENVOLVIMENTO







Questionário sobre a utilidade da ferramenta

*Obrigatório

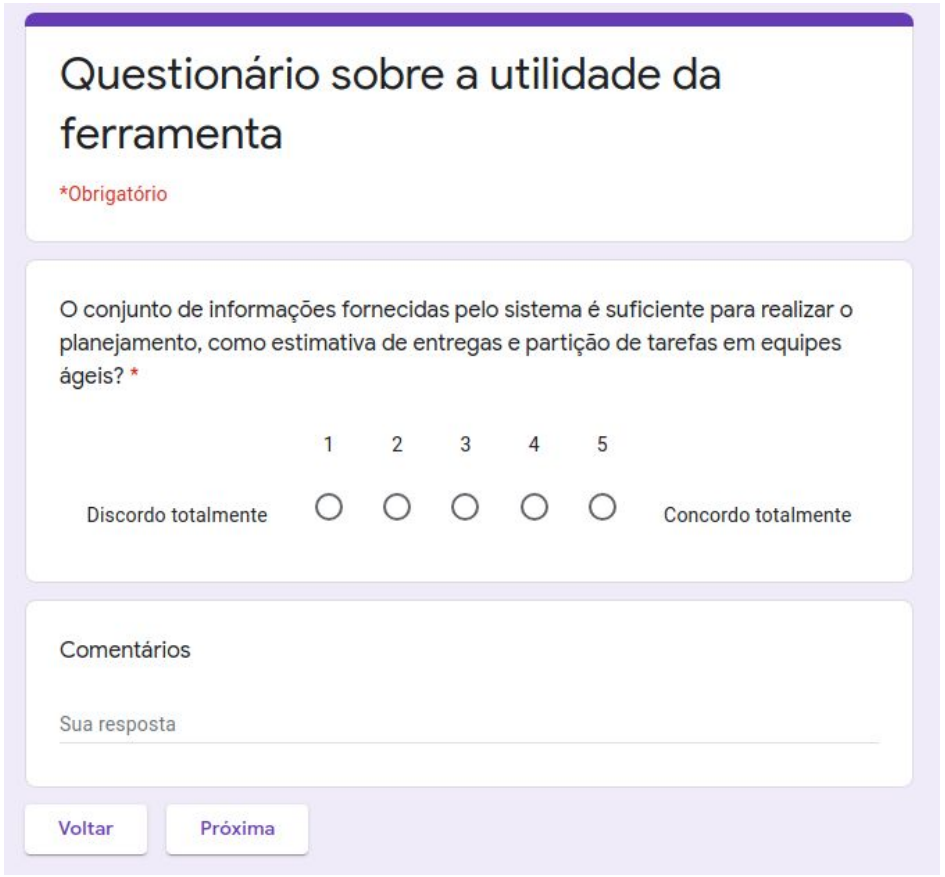
Você acha que a ferramenta oferece informações suficientes para apoio ao desenvolvimento do método Kanban dentro de equipes ágeis? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Comentários

Sua resposta

Voltar Próxima



Questionário sobre a utilidade da ferramenta

*Obrigatório

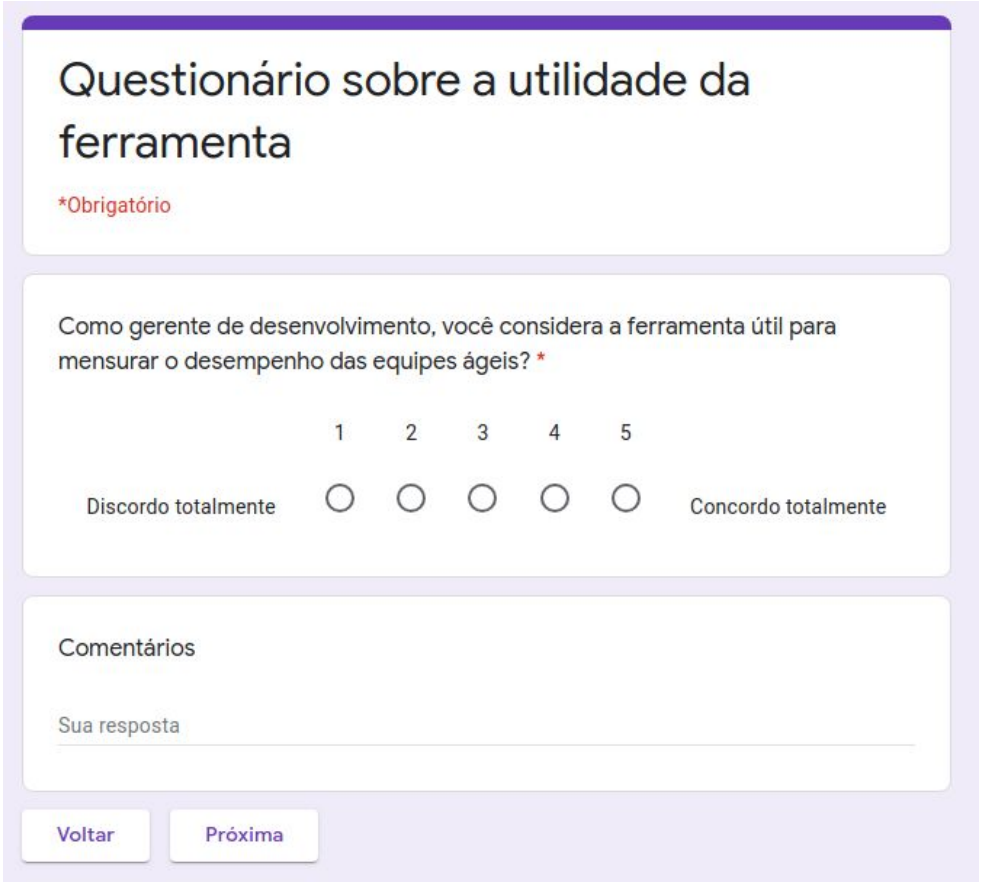
O conjunto de informações fornecidas pelo sistema é suficiente para realizar o planejamento, como estimativa de entregas e partição de tarefas em equipes ágeis? *

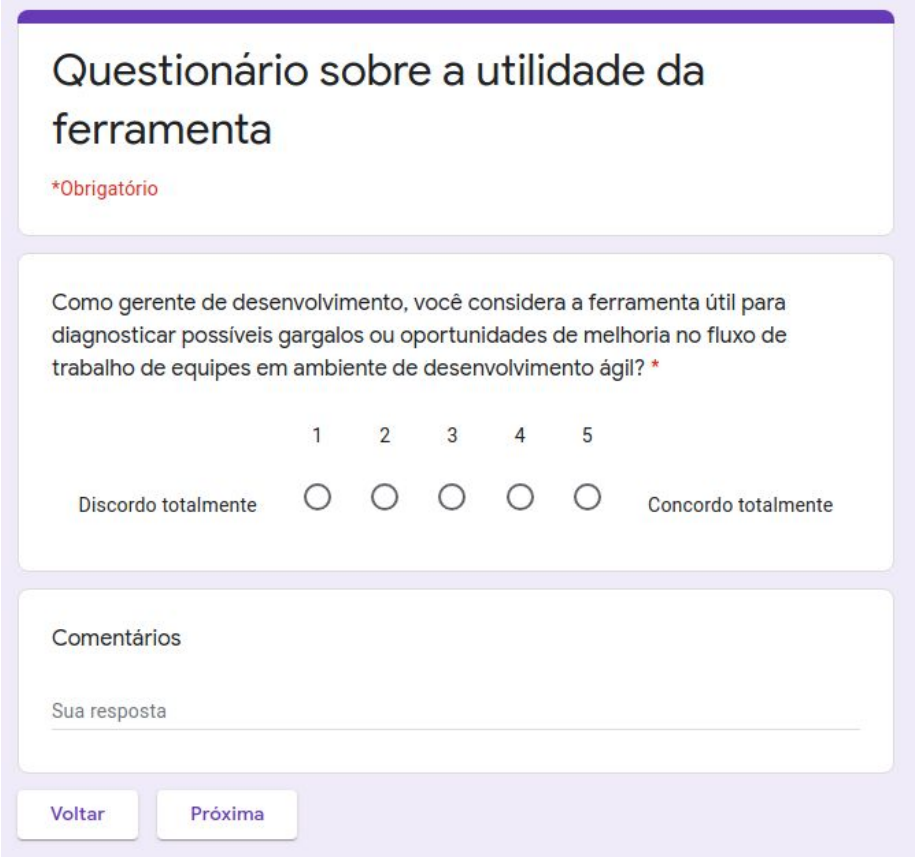
	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

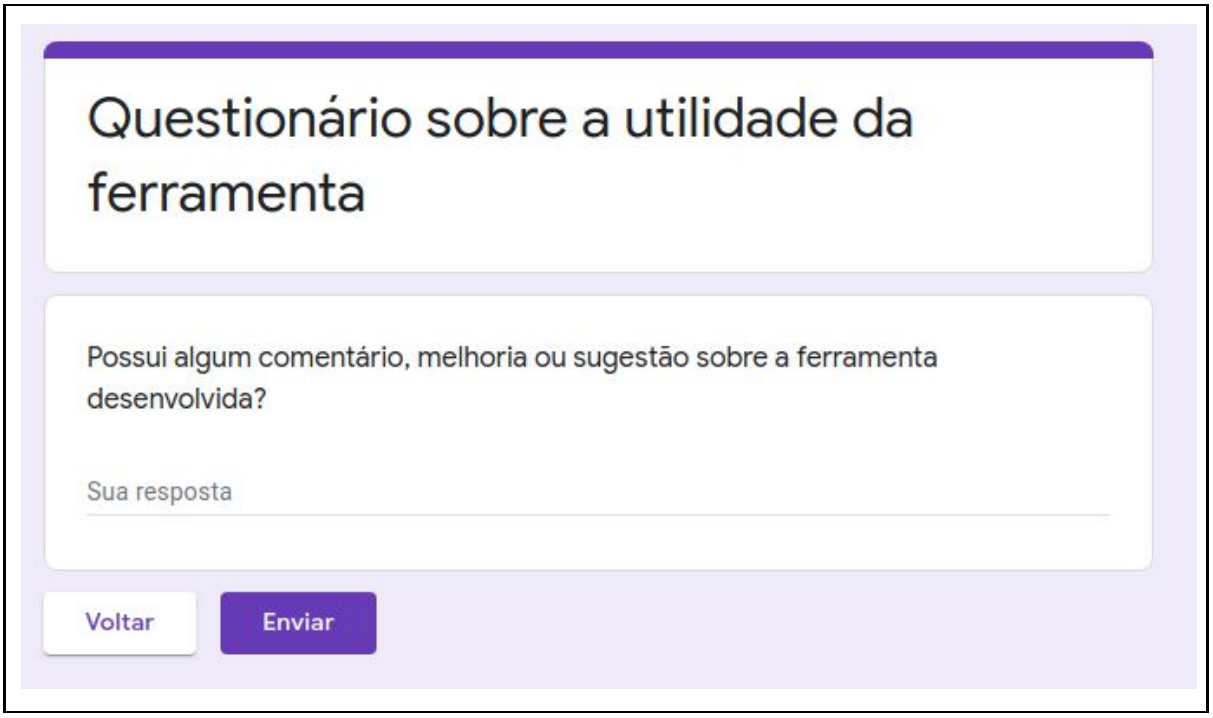
Comentários

Sua resposta

Voltar Próxima


Questionário sobre a utilidade da ferramenta
*Obrigatório
Como gerente de desenvolvimento, você considera a ferramenta útil para mensurar o desempenho das equipes ágeis? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário sobre a utilidade da ferramenta
*Obrigatório
Como gerente de desenvolvimento, você considera a ferramenta útil para diagnosticar possíveis gargalos ou oportunidades de melhoria no fluxo de trabalho de equipes em ambiente de desenvolvimento ágil? *
1 2 3 4 5
Discordo totalmente
Concordo totalmente
Comentários
Sua resposta
Voltar
Próxima


Questionário sobre a utilidade da ferramenta
Possui algum comentário, melhoria ou sugestão sobre a ferramenta desenvolvida?
Sua resposta
Voltar
Enviar

APÊNDICE D: CÓDIGO FONTE

- O código fonte da aplicação desenvolvida está disponível no seguinte link dos arquivos UFSC: https://arquivos.ufsc.br/d/a3fc529adcdf4312bb0e/

APÊNDICE E: ARTIGO FORMATO SBC

Implementação de métricas ágeis integradas à plataforma GitHub

Jacyara Bosse[1]

[1]Universidade Federal de Santa Catarina

Abstract. *With the great growth of the software market in recent years, organizations increasingly seek a high quality standard for their products, allowing the organization to grow and consolidate in the market. This constantly involves pursuing the improvement of software processes within the organization, often in the context of teams that use agile approaches. Managing these manual processes can often be costly, time consuming and ineffective. Automated metric collection can streamline management. The devices often use platforms to assist task management, and this work is covered in the GitHub platform, which is not only known for storing and versioning code, but also has tools to perform or manage tasks. Therefore, this work develops a web tool that collects agile metrics, integrated with the GitHub platform, aiming to identify improvements in the flow of teams in agile environments. The tool is evaluated in the context of agile teams, and with the results obtained, evidence suggests that a tool uses recommended indicators for software evaluation in agile development environments, and also providing automation of a process often done manually.*

Resumo. *Com o crescimento do mercado de software nos últimos anos, as organizações buscam cada vez mais um alto padrão de qualidade para seus produtos, permitindo que se consolidem no mercado. Isso implica constantemente em buscar a melhoria dos processos de software dentro da organização, muitas vezes no contexto de equipes que utilizam abordagens ágeis. A gestão desses processos de forma manual pode ser muitas vezes cara, demorada e ineficaz. A coleta de métricas de forma automatizada pode agilizar na gestão. Equipes ágeis costumam utilizar plataformas para auxiliar o gerenciamento de suas tarefas, e neste trabalho é abordada a plataforma GitHub, que além de ser conhecida pelo armazenamento e versionamento de código, também possui ferramentas para realizar o gerenciamento de tarefas. Portanto, neste trabalho é desenvolvida uma ferramenta Web que coleta métricas ágeis, integrada à plataforma GitHub, com objetivo de identificação de melhorias no fluxo em equipes em ambientes ágeis. A ferramenta é avaliada no contexto de equipes ágeis, e com os resultados obtidos levantam indícios de que a ferramenta fornece indicadores suficientes para a avaliação de software em ambientes de desenvolvimento ágil, e também fornecendo uma automatização de um processo feito muitas vezes de forma manual.*

1. Introdução

Graves problemas com a qualidade de software foram inicialmente percebidos na década de 1960 com o desenvolvimento dos primeiros grandes sistemas de software e continuaram a desafiar a engenharia de software ao longo do século XX. Na chamada crise do software, o software entregue era lento e pouco confiável, difícil de manter e de reusar (Sommerville, 2011). Com motivação nesses problemas, a Engenharia de Software surge no final dos anos 60 com o objetivo de contornar esses diversos problemas, buscando um tratamento mais controlado ao desenvolvimento de sistemas complexos e propondo soluções como a criação de processos de software, com a intenção de gerar produtos finais com qualidade.

A avaliação de processos de software é um procedimento de medição subjetivo que envolve o julgamento de pessoas qualificadas para identificação quantitativa de pontos fortes e fracos nos processos (Anacleto, 2005). Essa avaliação geralmente ocorre com o objetivo de se conhecer a situação atual dos processos executados pela organização (Anacleto, 2005).

A qualidade de um processo, como uma entidade intangível, é tipicamente avaliada com base nos produtos gerados pela sua execução. Essa avaliação normalmente parte da análise das características dos artefatos gerados em relação a um conjunto de resultados esperados de um modelo de referência (ACUÑA, 2000).

A partir dos anos 90, surgiram as chamadas metodologias ágeis para desenvolvimento de software (Manifesto Ágil, 2001). Além de uma nova visão sobre como desenvolver software, ser ágil está associado a uma mudança cultural, uma nova forma de pensar, e não apenas mais uma abordagem, modelo, método ou processo de desenvolvimento. Ser ágil é assumir um conjunto de valores, princípios e práticas que flexibilizam o desenvolvimento de software de forma a responder às constantes mudanças do mercado, enfocando o que é importante (Larman, 2003). Este trabalho é aplicado em equipes que promovem essa cultura ágil, ou seja, as equipes ágeis.

A medição tem como principal foco apoiar a tomada de decisão em relação aos projetos, processos e atendimento aos objetivos organizacionais. Segundo a ISO/IEC 12207 [ISO/IEC, 2008] o propósito da Medição é coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar o efetivo gerenciamento dos processos e demonstrar objetivamente a qualidade dos produtos.

Segundo McGarry (2002), a medição é o nível mais importante em um projeto. A medição de software ajuda o gerente de projetos a fazer um trabalho melhor. Ajuda a definir e implementar planos mais realistas, alocar adequadamente recursos escassos para implementar esses planos e monitorar com precisão o progresso e o desempenho em relação a esses planos.

Segundo a ISO/IEE 15939 (2017), Uma medida é definida em termos de um atributo e o método para quantificá-lo. Uma medida base é funcionalmente independente de outras

medidas, que captura informações sobre um único atributo. E a medida derivada é uma medida definida como uma função de dois ou mais valores de medidas básicas. Derivado as medidas capturam informações sobre mais de um atributo ou o mesmo atributo de várias entidades.

Segundo Hartman e Dymond (2006), a métrica ágil deve reforçar princípios ágeis, medir resultados e não saídas, seguir tendências e não números, responder uma pergunta específica para uma pessoa real e pertencer a um conjunto pequeno de métricas e diagnósticos.

Diversas equipes em ambientes de desenvolvimento ágil utilizam o método Kanban. Este método é uma abordagem enxuta para o desenvolvimento de software ágil. Na realidade, o Kanban possui uma série de significados. Literalmente, é uma palavra japonesa que significa "cartão visual". Portanto, no contexto de desenvolvimento de software ele é relativamente novo, enquanto no processo produtivo o termo já é utilizado há mais de meio século (Albino, 2017).

Conforme Boeg (2010), a maioria dos especialistas concorda que o Kanban é um método de gestão de mudanças, dando ênfase em alguns princípios, como visualizar o trabalho em andamento, limitar o trabalho em progresso e tornar explícitas as políticas sendo seguidas.

Algumas organizações vêm implantando a OKR para atingir objetivos organizacionais. OKR (do inglês, Objective and Key-Results), é um framework de definição e gerenciamento de objetivos. O framework tem dois componentes básicos: o objetivo (o que queremos alcançar) e um conjunto de resultados-chaves (como saber se estamos chegando lá) (Castro, 2015). O propósito do OKR é criar alinhamento na organização. Assim, objetivos ambiciosos e resultados-chaves mensuráveis são estabelecidos e precisam ser visíveis para todos os níveis da organização (Niven e Lamorte, 2016).

Muitas equipes utilizam plataformas para gerenciamento de código e de tarefas. Uma ferramenta que permite não só o gerenciamento de código mas também o de tarefas é a plataforma GitHub, onde os usuários são desenvolvedores que podem criar/contribuir/compartilhar/buscar por repositórios de projetos de acordo com assuntos e linguagens de programação em que foram desenvolvidos (Rocha, 2016).

A plataforma GitHub (GitHub, 2019) começou a operar em 2007, quando foi realizado seu primeiro “commit”. Hoje em dia são mais de 40 milhões de usuários e mais de 100 milhões de repositórios, conforme estatísticas de outubro de 2019 no “sobre” do site do GitHub (GitHub, 2019).

Mesmo com as facilidades de gerenciamento de tarefas oferecidas pelas ferramentas e técnicas disponíveis, muitas equipes ainda possuem dificuldades em estimar tarefas, tempo de entregas e acompanhar o esforço aplicado.

Assim, este trabalho visa a automação parcial do gerenciamento de processos utilizando a API do GitHub para realizar a integração de uma ferramenta com a plataforma. A partir da coleta de dados de tarefas, versões, “pull requests”, e outras funções do GitHub, essa otimização do processo de avaliação traz benefícios para as equipes ágeis que utilizam a plataforma GitHub.

Com isso, espera-se que a partir da coleta de tais dados aplicados a necessidade de cada equipe ágil, seja possível gerar indicadores e resultados que auxiliarão as equipes em vários quesitos de melhoria de processos e estimativa de desenvolvimento de tarefas.

1.1 Objetivos

A partir da contextualização do problema, o objetivo geral e objetivos específicos deste trabalho são assim definidos:

1.1.1 Objetivo geral

O objetivo deste trabalho de conclusão de curso é a análise e implementação de métricas ágeis utilizadas no gerenciamento de projetos por equipes ágeis, coletadas de forma automatizada e integrada à plataforma GitHub.

O trabalho realizado é aplicado em estudo de caso aplicado em equipes ágeis do laboratório Bridge. Essas equipes utilizam o método Kanban nos seus processos internos de desenvolvimento, incluindo a estimativa de entregas e particionamento de tarefas.

O preenchimento dos dados em uma planilha do Kanban utilizada como referência nas equipes, é preenchida de forma manual. É observando o estados das tarefas no quadro da equipe na plataforma GitHub, e então passado para a planilha, semanalmente. A ferramenta a ser implementada tem como objetivo automatizar esse processo, obtendo um histórico consultando as informações obtidas na plataforma GitHub, de acordo com cada equipe.

A implementação de uma ferramenta que busque dados da plataforma GitHub de forma automatizada, irá permitir então uma melhor avaliação dos processos de software nos ambientes ágeis do laboratório. Esta pesquisa busca por meio do desenvolvimento de um sistema web, viabilizar a melhoria na visualização dos processos e fluxos da equipe.

Por meio da integração com o GitHub, pretende-se gerar indicadores que servirão como uma entrada para estruturas de obtenção de resultados. O laboratório além do Kanban, utiliza a OKR como estrutura para o alcance de objetivos organizacionais. Portanto, além da inclusão de dados para geração de métricas do Kanban, de forma interna da equipe, busca-se também alimentar o acompanhamento de key results que servem como análise de entrada dos objetivos organizacionais do laboratório.

1.1.2 Objetivos específicos

- Análise das métricas ágeis utilizadas por equipes em ambientes de desenvolvimento ágil.
- Análise e modelagem de uma ferramenta com base na análise de métricas significativas para equipes ágeis.
- Desenvolvimento da Ferramenta Web com integração à plataforma GitHub
- Avaliação da ferramenta em um estudo de caso com equipes ágeis na organização alvo.

2. Fundamentação teórica

Este capítulo apresenta os principais conceitos e abordagens utilizadas ao longo deste trabalho. Nesta etapa é realizada a revisão bibliográfica com o enfoque nos conceitos de Engenharia de Software aplicados na avaliação de processos de software.

Além dos assuntos envoltos pela Engenharia de Software, é realizado também um estudo também sobre as estruturas e plataformas relacionadas ao trabalho, como a estrutura Objective and Keys Result (OKR) e a plataforma GitHub.

2.1 Processo de Software

A ISO/IEC TR 29110-1:2011 (ISO/IEC, 2011) introduz o conceito de processo como conjunto de atividades inter-relacionadas ou interativas que transformam entradas em saídas. Desta forma, de acordo com Pressman (2010) o processo de software pode ser definido como uma coleção de padrões que definem um conjunto de atividades, ações, tarefas de trabalho, produtos de trabalho e/ou comportamentos relacionados necessários ao desenvolvimento de softwares de computador.

Muitas empresas adotam modelos de processos com o intuito de definir regras de forma geral sobre seus processos. Um modelo de processo apresenta uma filosofia, uma forma geral de comportamento, baseada na qual processos específicos podem ser definidos. Um modelo de processo para as atividades de projeto e desenvolvimento de software também pode ser chamado de ciclo de vida. Assim, quando uma empresa decide adotar um processo, ela deve inicialmente buscar um modelo de processo e adaptar a filosofia e práticas recomendadas para criar seu próprio processo. A partir daí, todos os projetos da empresa deverão seguir este processo definido (Wazlawick, 2013).

A definição e atuação dos processos de software dentro de uma empresa tem um papel fundamental para conseguir a garantia de qualidade do produto desenvolvido. Segundo Pressman (2010, p.25), "Pela combinação de padrões, uma equipe de software pode construir um processo que melhor satisfaça às necessidades de um projeto". Então o desenvolvimento da melhoria de processos é um fator importante para obter esses objetivos alinhados à uma maior produtividade e controle.

Inclusive, de acordo com Wazlawick (2013), há várias vantagens em se definir desenvolvimento de software como um processo, entre as principais estão: O tempo de treinamento pode ser reduzido, os produtos podem ser mais uniformizados e a possibilidade de capitalizar experiências.

E como este trabalho irá abordar processos de software estabelecidos de forma independente por equipes ágeis, é importante ressaltar que, assim como aponta Sommerville (2011, p.495), "Não existe algo como um processo de software 'ideal' ou 'padrão' que seja aplicável a todas as organizações ou para todos os produtos de software de um tipo particular".

2.2 Abordagens ágeis

Nos dias de hoje, as empresas operam em um ambiente global, com mudanças rápidas. Assim, precisam responder a novas oportunidades e novos mercados, a mudanças nas condições econômicas e ao surgimento de produtos e serviços concorrentes. Softwares fazem parte de quase todas as operações de negócios, assim, novos softwares são desenvolvidos rapidamente para obterem proveito de novas oportunidades e responder às pressões competitivas (Sommerville, 2011).

Ainda neste âmbito, de acordo com Sommerville (2011), processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida, projetar, construir e testar o sistema não estão adaptados ao desenvolvimento rápido de software. Com as mudanças nos requisitos ou a descoberta de problemas de requisitos, o projeto do sistema ou sua implementação precisa ser refeito ou retestado. Como consequência, um processo convencional em cascata ou baseado em especificações costuma ser demorado, e o software final é entregue ao cliente bem depois do prazo acordado.

Os modelos ágeis de desenvolvimento de software seguem uma filosofia diferente dos modelos prescritivos. Apesar de serem usualmente mais leves, é errado entender os métodos ágeis como modelos de processos meramente menos complexos ou simplistas. Não se trata de apenas de simplicidade, mas de focar mais nos resultados do que no processo (Wazlawick, 2013).

Os princípios de desenvolvimento de software ágil que são seguidos e defendidos surgiram dos princípios tradicionais de desenvolvimento de software e de várias experiências baseadas nos sucessos e fracassos dos projetos de software (Rao, 2011).

Conforme o Manifesto Ágil (Agile, 2001), O movimento da metodologia ágil não é anti-metodologia; Na verdade, é a restauração da credibilidade da palavra. Também é a restauração de um equilíbrio: adotar a modelagem, mas não apenas para arquivar alguns diagramas em um repositório corporativo empoeirado.

Aprofundando mais sobre o assunto, o propósito do Manifesto Ágil (2001) é o seguinte:

"Estamos descobrindo maneiras melhores de desenvolver software fazendo isso e ajudando os outros a fazer isso. Valorizamos:

- Indivíduos e interações sobre processos e ferramentas.
- Software funcionando sobre documentação completa.
- Colaboração do cliente sobre negociação de contrato.
- Responder às mudanças sobre seguir um plano."

Portanto, este trabalho irá focar em equipes ágeis, ou seja, equipes de desenvolvimento de software que fazem uso das abordagens ágeis no dia a dia. Essas equipes utilizam princípios da metodologia ágil *Scrum*.

2.2.1 Scrum

Segundo Schwaber (2002, p.2), a metodologia *Scrum* tem como objetivo definir um processo para gerência de projetos de software, que seja focado nas pessoas e que seja indicado para ambientes em que os requisitos surgem e mudam rapidamente.

Antes de entrar em detalhes sobre a metodologia Scrum, é necessário definir alguns conceitos como os de *sprint*, *product backlog* e *Scrum teams* (equipes).

Portanto, conforme Schwaber (2002), *sprints* são divisões do desenvolvimento em intervalos de tempos de, no máximo 30 dias. As equipes são pequenas de, no máximo, 7 integrantes. E *product backlog* é uma lista de atividades que provavelmente serão desenvolvidas durante o projeto.

A ideia por trás do Scrum é que toda a equipe deve ter poderes para tomar decisões, de modo que o termo 'gerente de projeto' tem sido deliberadamente evitado. Toda a equipe participa das reuniões diárias para a manutenção do foco da equipe. Durante a reunião, todos os membros da equipe compartilham informações, descrevem seu progresso desde a última reunião, os problemas que têm surgido e o que está planejado para o dia seguinte. Isso garante que todos na equipe saibam o que está acontecendo e, se surgirem problemas, poderão replanejar o trabalho de curto prazo para lidar com eles (Sommerville, 2011).

Sobre as fases que compõem o ciclo de vida do Scrum, Sommerville (2011) aponta a existência de três fases. A primeira é uma fase de planejamento geral, em que se estabelecem os objetivos gerais do projeto e da arquitetura do software. Em seguida, ocorre uma série de ciclos de sprint, sendo que cada ciclo desenvolve um incremento do sistema. Finalmente, a última fase do projeto encerra o projeto, completa a documentação exigida, como quadros de ajuda do sistema e manuais do usuário, e avalia as lições aprendidas com o projeto.

Figura 1: Ciclo *Sprint*

Fonte: Sommerville, 2011

. Rising e Janoff (2000), que aplicaram a metodologia ágil Scrum em pequenas equipes ágeis, relatam uma série de resultados positivos após algumas iterações. Sendo esses alguns dos pontos destacados:

- O produto se torna uma série de partes gerenciáveis.
- O progresso é feito, mesmo quando os requisitos não são estáveis.
- Tudo é visível para todos.
- A comunicação da equipe melhora e compartilham sucessos ao longo do caminho
- Os clientes vêem a entrega pontual de incrementos e obtêm feedback frequente sobre como o produto realmente funciona,
- Um relacionamento com o cliente se desenvolve, a confiança e o conhecimento cresce, e é criada uma cultura onde todos esperam que o projeto seja bem-sucedido.

2.3 Kanban

Conforme descrito por Anderson (2016), o Kanban é um método de organização e gerenciamento de serviços profissionais. Ele usa conceitos de *Lean*, como limitar o trabalho em andamento para melhorar os resultados. Um sistema Kanban é um meio de equilibrar a demanda de trabalho a ser realizado com a capacidade disponível para iniciar um novo trabalho.

A palavra "Kanban" vem do japonês e significa "Cartão Visual". Uma pesquisa pela palavra no Google retorna mais de 5 milhões de resultados; isto porque a palavra também é utilizada para descrever o sistema que vem sendo utilizado há décadas pela Toyota para visualmente controlar e equilibrar a linha de produção. O termo tem se tornado quase sinônimo da implementação dos princípios *Lean*. Então, embora sistemas kanban sejam conceito relativamente novo em TI, vêm sendo utilizados por mais de 50 anos no sistema de produção Lean na Toyota. (Boeg, 2010).

Segundo Anderson, (2016) o Kanban é sustentado em seis prioridades, sendo elas:

- Visualize o fluxo de trabalho
- Limite o trabalho em progresso
- Meça e gerencie o fluxo de trabalho
- Torne as políticas explícitas
- Desenvolva loops de feedback
- Melhore de forma colaborativa

Boeg (2010) aponta que existem diversas abordagens para o Kanban, mas a maioria dos especialistas concorda que o Kanban é um método de gestão de mudanças, que dá ênfase aos seguintes princípios:

- Visualizar o trabalho em andamento;
- Visualizar cada passo em sua cadeia de valor, do conceito geral até software que se possa lançar;
- Limitar o Trabalho em Progresso (WIP – *Work in Progress*), restringindo o total de trabalho permitido para cada estágio;
- Tornar explícitas as políticas sendo seguidas;
- Medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessas decisões;
- Identificar oportunidades de melhorias, criando uma cultura Kaizen, na qual a melhoria contínua é responsabilidade de todos.

Outros benefícios podem ser observados ao adotar o kanban em uma equipe ágil. Os gargalos do processo ficarão visíveis em tempo real, é fornecido uma maneira de criar software de forma ágil, é útil para situações nas quais há uma alta taxa de incerteza e alta variabilidade, e também aumenta a visibilidade de tudo que está acontecendo (Albino, 2017).

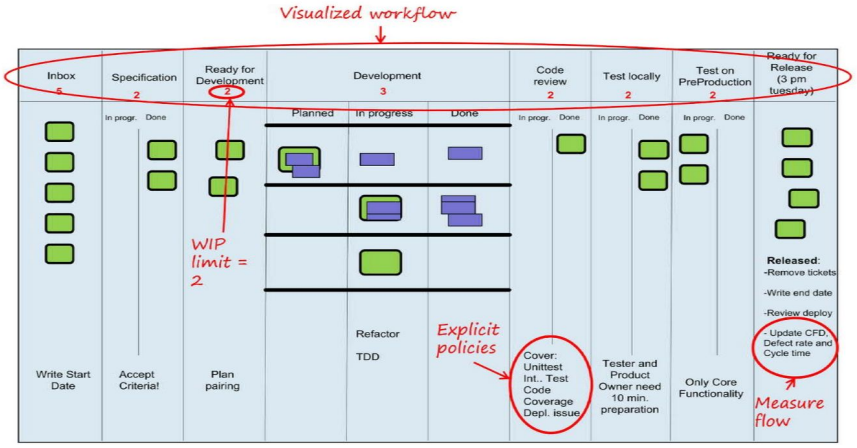
2.3.1 Quadro Kanban

O quadro Kanban é composto de colunas com os estados onde o cartão Kanban está situado. Segundo Peinado (2007), o cartão kanban é responsável pela comunicação e pelo funcionamento de todo o sistema, nele devem estar contidos as informações mínimas para o bom funcionamento da linha de produção. Sendo necessário, ele poderá conter um número maior de informações, desde que sejam importantes para a área específica, onde se pretende implementar o sistema kanban.

Segundo Boeg (2010), Visualizar seu fluxo de trabalho gera uma série de benefícios, sendo os mais importantes:

- Foco no "todo": Fica visível exatamente como o seu trabalho afeta as outras pessoas e vice-versa;
- Transparência: Todos sabem exatamente o que está acontecendo e nenhuma informação é escondida;
- Identificação de desperdícios: Surgirá naturalmente um questionamento da razão pela qual as coisas são feitas;

Figura 2: Quadro Kanban



Fonte: Boeg, 2010

2.4 Métricas de software

Segundo Pressman (2010) a mensuração é aplicada no processo de desenvolvimento de software ou atributos de um produto com o objetivo de melhorá-lo de forma contínua que utilizando ao longo do projeto auxilia na estimativa, no controle de qualidade, na avaliação da produtividade e no controle do projeto.

Com as métricas os gerentes conseguem informações quantitativas que auxiliam na tomada de decisões e para obter uma visão melhor do trabalho que está sendo realizado, podendo assim desenvolver um software mais confiável (Inthurn, 2001).

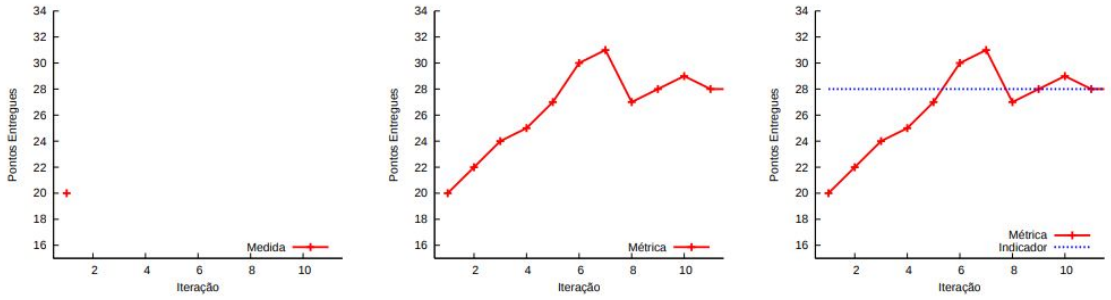
2.4.1 Métricas, medidas e indicadores

Para discutir o papel das métricas no acompanhamento de projetos ágeis, é importante conhecer as diferenças entre o conceitos de medidas, métricas e indicadores.

Uma medida é a avaliação de um atributo segundo um método de medição específico, funcionalmente independente de todas as outras medidas e capturando informação sobre um único atributo (McGarry, 2002).

Conforme a IEEE (IEEE Std 610.12, 1990), uma métrica é um método para determinar se um sistema, componente ou processo possui um certo atributo. Um indicador é um dispositivo ou variável que pode ser configurado para um determinado estado com base no resultado de um processo ou ocorrência de uma determinada condição.

Figura 3: Diferença entre métrica, medida e indicadores



(a) Medida (b) Métrica (c) Indicador

Fonte: Sato, 2007

2.4.2 Métricas ágeis

Com relação às métricas aplicadas com objetivo de medir desempenho em equipes ágeis, Sato (2008) e Cohn (2008) citam uma série de critérios que uma boa métrica ágil deve ter, tais como: reforçar princípios ágeis, envolvendo toda a equipe, seguir tendências e não números; pertencer a um conjunto pequeno de métricas e diagnósticos; ser facilmente coletada, deixar claro os fatores que a influenciam para evitar manipulações, facilitar a melhoria do processo e fornecer *feedbacks*.

Conforme Albino (2017), outra característica importante sobre métricas de negócio é que elas devem ser compreensíveis, isto é, elas não podem ser complicadas e todos devem compreender o que elas representam. A métrica de negócio deve também ser uma relação ou

uma taxa. Números absolutos não devem ser usados. A quantidade de usuários pode ser pouco útil, porém, a porcentagem de usuários ativos diários já poderá trazer tendências de crescimento. Razões e taxas são comparativas, o que ajuda a tomar melhores decisões.

Algumas das métricas ágeis também destacadas pelo autor (Albino, 2017), especialmente quando se trata de métricas utilizadas dentro do desenvolvimento do método Kanban, são:

- *Work in Progress*
- Cumulative Flow Diagram
- *Lead time*
- *Throughput*

O *WIP (Work in progress), CFD (Cumulative Flow Diagram), lead time* e *Throughput* são descritos da seção 4.5, junto com outras métricas utilizadas em equipes ágeis.

2.5 Objectives and Key Results (OKR)

Uma das partes finais deste trabalho é a captura dos indicativos obtidos na avaliação do processo. Esses indicativos servirão como entrada para o OKR, pois é uma forma de medir resultados de processos e é utilizado na organização-alvo do estudo de caso deste trabalho.

A *Objectives and Key Results* - OKR, é uma estrutura de pensamento crítico e disciplina contínua que busca garantir que os funcionários trabalhem juntos, concentrando seus esforços para fazer contribuições mensuráveis que impulsionam a empresa para frente (Niven & Lamorte, 2016).

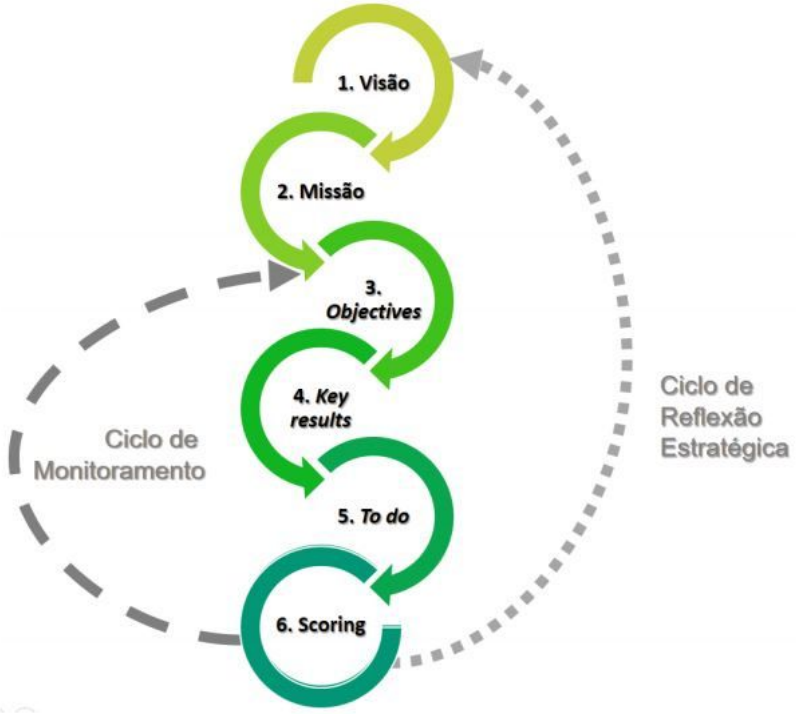
OKR é uma estrutura de definição de metas criada pela Intel e adotada por várias empresas do Vale do Silício. O Google é o caso mais famoso, tendo adotado o OKR em seu primeiro ano. Twitter, LinkedIn, Dropbox e Oracle estão entre outros adotantes (Castro, 2015).

Também segundo Castro (2015), essas são algumas das principais características que tornam a OKR única:

- Simplicidade: para permitir ciclos frequentes de estabelecimento de metas , o processo é extremamente simples. Os próprios OKRs devem ser simples e fáceis de entender.
- Cadência mais curta: Em vez de usar um processo anual de planejamento estático, o OKR usa ciclos mais curtos de definição de metas , permitindo um planejamento dinâmico e uma adaptação mais rápida às mudanças.
- Open Source: OKR é um framework de código aberto para que as empresas o adaptem a cada cultura e contexto, criando diferentes versões dele.
- Objetivos a longo prazo: objetivos que tiram o time da zona de conforto e fazem com que repensem o modo como trabalham para atingir o máximo desempenho.

Conforme aponta Luna (2017), a OKR possui cinco iterações e dois ciclos. A primeira parte da análise em como uma organização “quer ser reconhecida” (visão institucional), permitindo a identificação o “seu propósito” (missão institucional). A partir destas duas informações essenciais, o time procura identificar ao menos três objetivos (objectives), que uma vez alcançados auxiliarão a organização a cumprir sua missão. Para cada objetivo, então, são identificadas de uma a três métricas (key results) que permitirão medir o progresso daquele objetivo. E só então, são elaboradas as ações (to do) para alcançar os resultados descritos pelas métricas estipuladas. A partir daí, os OKRs são distribuídos no tempo e monitorados periodicamente através de reuniões de avaliação (Scorings).

Figura 4: Ciclos da OKR



Fonte: Luna, et al. 2017

Os ciclos de monitoramento ocorrem dentro das iterações de avaliação, visando avaliar como os objetivos estão sendo alcançados e realizando ajustes sempre que necessário. Já os ciclos de reflexão estratégica ocorrem sempre que a estratégia organizacional precisa ser adequada à uma nova realidade (Luna, 2017).

4 Análise e projeto

Este capítulo apresenta a análise dos requisitos, identificação dos processos utilizados nas equipes onde será aplicado o estudo e também uma descrição da arquitetura geral da ferramenta, juntamente com as tecnologias que serão utilizadas. São também analisadas as métricas que as equipes ágeis utilizam no dia a dia para gerenciamento de seus projetos, de forma a poder implementar a automatização da coleta de dados.

Ao final são apresentadas as propostas de protótipos de alta fidelidade de tela para o sistema, referentes às histórias de usuário sintetizadas depois das entrevistas com os usuários.

4.1 Análise do Contexto dos Processos

O Laboratório Bridge (Laboratório Bridge, 2019) é um laboratório integrado ao Centro Tecnológico da Universidade Federal de Santa Catarina voltado para a pesquisa e inovação em tecnologia da informação, entregando produtos para qualificar a gestão pública. O laboratório possui mais de 100 colaboradores e entre eles, a grande maioria são estudantes da Universidade Federal de Santa Catarina.

Um de seus projetos é o e-SUS AB (Laboratório Bridge, 2019), um sistema de prontuário eletrônico para ser utilizado nas unidades básicas de saúde do Brasil. Outro de seus projetos é o SISMOB (Laboratório Bridge, 2019), sistema que realiza o monitoramento de obras de engenharia e infraestrutura de diversos estabelecimentos de saúde financiados

pelo Ministério da Saúde, e o RNI (Laboratório Bridge, 2019), que é o sistema de registro nacional de implantes da ANVISA.

Para a realização e aplicação dos estudos deste trabalho de conclusão de curso (conforme declaração de consentimento no Apêndice A), será realizado o enfoque em duas equipes ágeis que trabalham no projeto e-SUS.

A maior parte das equipes ágeis que integram o projeto e-SUS AB, utilizam o Kanban no gerenciamento de suas tarefas. Esse trabalho é feito através de uma planilha base comum para todas as equipes, em que cada uma popula com os seus respectivos dados, fornecendo as métricas específicas para cada equipe.

Uma das equipes tem como integrante a autora deste trabalho, e a outra equipe a autora tem contato direto. Por esse motivo foi feita a escolha dessas duas equipes, por critérios de proximidade e conveniência.

Para o levantamento dos requisitos foram realizadas entrevistas com membros das duas equipes selecionadas e também com o responsável pelo gerenciamento de OKRs dentro do laboratório. A análise do contexto foi realizada por meio da modelagem descritiva do processo realizado pelas duas equipes.

4.2 Modelagem Descritiva dos Processos

Esta etapa do trabalho foi traçada diretamente com as equipes do laboratório Bridge que vão servir de base para o estudo e aplicação da avaliação. As duas equipes ágeis utilizam a plataforma GitHub tanto para o gerenciamento de tarefas como para gerenciamento de código. Cada equipe tem seu próprio projeto dentro do repositório da organização. Em cada projeto chamado de *board* as equipes incluem suas tarefas e os seus respectivos estados, chamados de coluna.

A planilha do Kanban utilizada pelas equipes é populada semanalmente de acordo com o estado das tarefas no seu *board.* Nesse documento se encontram gráficos e informações importantes para o direcionamento do planejamento de entregas de equipe. Essas métricas são explicadas na seção “4.5.1 Análise dos indicadores” .

4.3 Entrevistas com os usuários

Foram realizadas duas reuniões e entrevistas com cada equipe para alinhar seus processos e posteriormente identificar pontos que sejam interessantes para a avaliação, obtendo indicativos relevantes para o estudo.

Em paralelo às conversas com as equipes, também foram discutidos os possíveis requisitos e levantado ideias com a colaboradora Luisa Lacerda, responsável pela implantação e acompanhamento do Kanban nas equipes ágeis. Nessa conversa, revisamos os requisitos e importância levantado pelas equipes, onde as equipes achavam necessidade principalmente nas métricas em que é necessário para população da planilha do Kanban. Entretanto, por mais que essa seja a necessidade atual, é importante olhar além dessas métricas e levantar dados que até mesmo as equipes ainda não sabem o quão vantajosos possam ser no futuro.

Pelo fato do método Kanban ser abordado constantemente no dia a dia das equipes, algo que foi constatado após as entrevistas, as principais métricas coletadas são as principais para a visualização desse método

Nas próximas seções é feito um estudo do processo detalhado de cada equipe ágil escolhida para o estudo de caso, as equipes Supernova e Royal Flush.

4.3.1 Equipe Supernova

A equipe supernova possui cinco integrantes, dentre eles, existe um analista de sistemas, responsável pela análise e documentação das tarefas, três desenvolvedores *fullstack,* responsáveis pelas implementações, e dois testadores, responsáveis pela qualidade das tarefas.

A primeira reunião com a equipe foi realizada no dia 8 de outubro de 2018, e contou com a presença do gerente de projetos e dos componentes da equipe, incluído a autora que faz parte dessa equipe de trabalho. Neste momento foi relatado a ideia geral dos estudos e instigado os participantes a pensarem em possíveis indicadores consideráveis para poder obter-se uma análise de como seus processos fluem.

A segunda reunião foi realizada no dia 18 de outubro de 2018 e contou apenas com os membros da equipe. Foi efetuado um *brainstorming* e coletado diversas ideias que posteriormente se tornaram requisitos funcionais do sistema.

A terceira reunião aconteceu no início do segundo semestre de 2019, e foi importante para revisar todos os requisitos levantados anteriormente, ainda mais que nesse momento o Kanban está mais desenvolvido no dia a dia das equipes.

Mesmo a autora fazendo parte da equipe e estando ativamente envolvida nos processos, alguns pontos também foram levantados para a definição da modelagem do processo da equipe. A decisão foi de modelar o processo num nível maior, no nível de módulos.

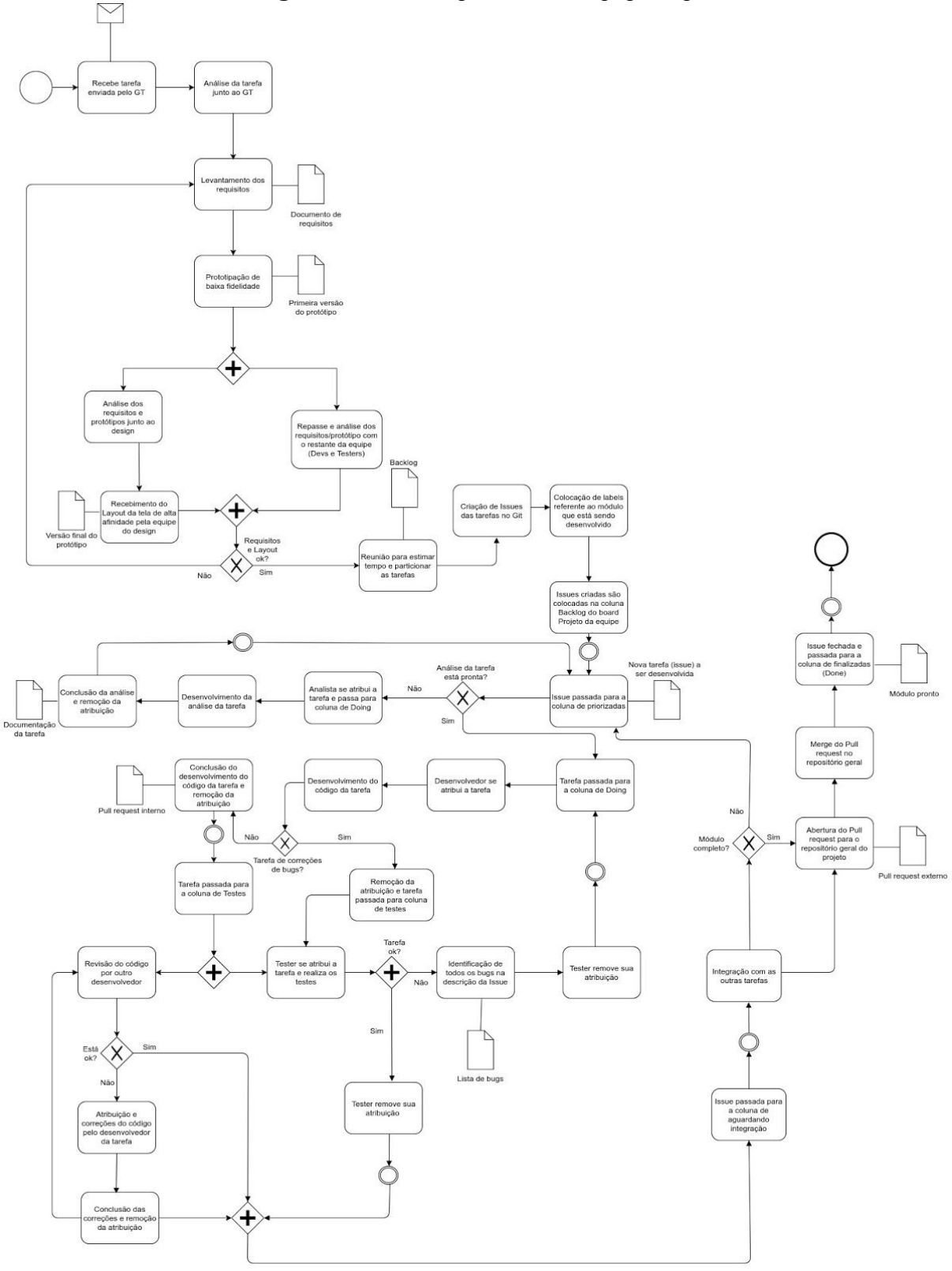
O módulo é um entregável com um nível alto de complexidade que no fim gera um produto dentro do projeto. Os ícones de eventos intermediários dispostos pelo fluxo (um círculo dentro de outro) representam as etapas onde vai ter coleta de indicativos.

Então, nesse sentido, segue na figura 9 a modelagem do processo, utilizando a notação BPMN 2, desde que um módulo chega como demanda para equipe até a momento que esse módulo é integrado com o restante do projeto:

Nomenclatura utilizada na modelagem dos processos:

1. **Git** - Plataforma GitHub
2. **GT** - Grupo de trabalho do Ministério da Saúde
3. **Devs/Testers** - Desenvolvedores fullstack/Testadores
4. **Board** - Quadro com várias colunas, que representam o estado de uma tarefa.
5. **Issue** - Pequena tarefa a ser desenvolvida cadastrada em uma issue do GitHub
6. **Pull request interno** - Pull request aberto no repositório interno, onde apenas membros da equipe e convidados tem acesso.
7. **Pull request externo** - Pull request aberto no repositório geral do projeto.
8. **Merge** - Quando um pull request externo é integrado ao código geral do projeto.
9. **Label** - Etiqueta colocada na issue com informação

Figura 5: Fluxo do processo da equipe Supernova



Fonte: Diagrama elaborado pela autora.

4.3.2 Equipe Royal Flush

Da mesma forma e dinâmica descrita pela equipe Supernova, a equipe ágil Royal Flush possui uma analista, três desenvolvedores fullstack e dois testadores.

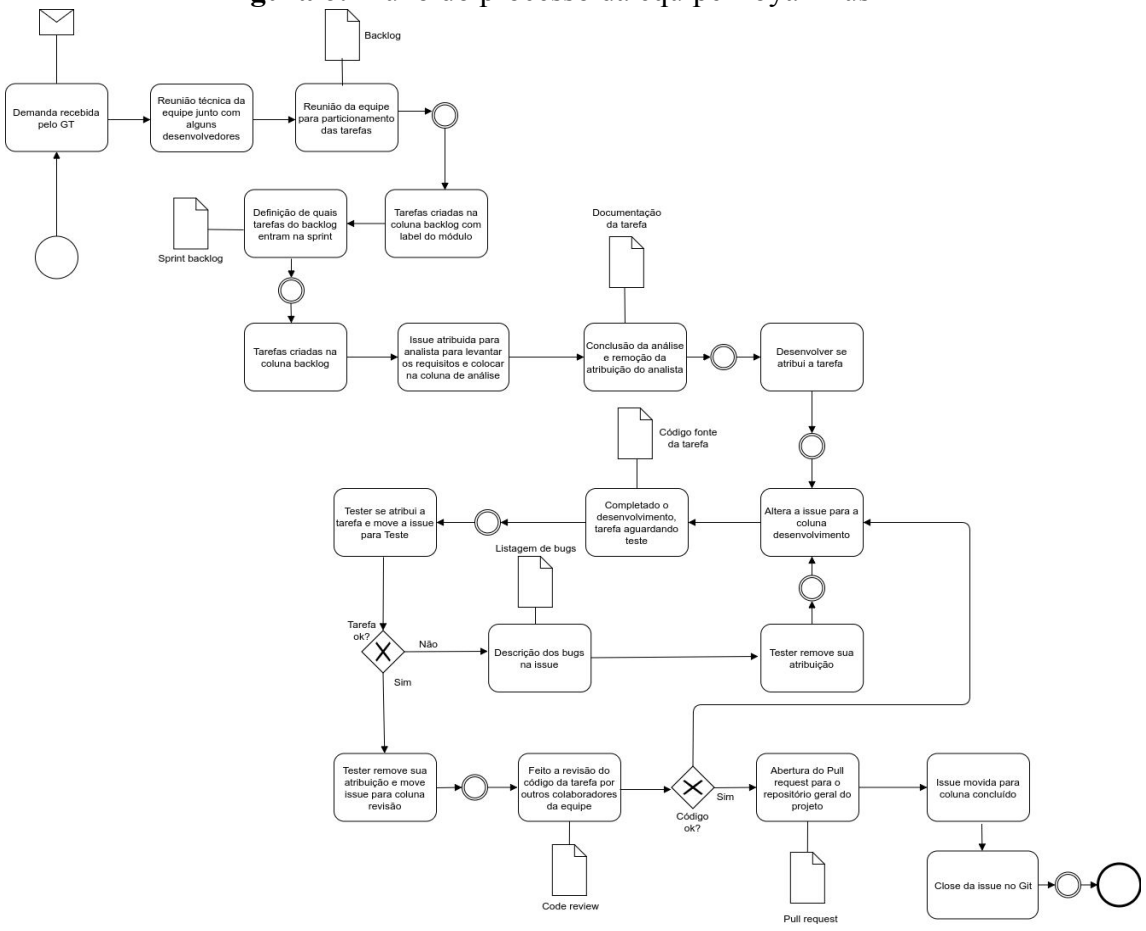
A equipe trabalha também com demandas complexas, ou seja, os módulos. Cada módulo é planejado com reuniões técnicas e partilhado em menores tarefas. Essas menores tarefas ficam em um backlog, onde ao final de cada semana é decidido quais tarefas entrarão na próxima *sprint*.

As reuniões foram realizadas no início do segundo semestre de 2019, e nesse momento foi buscado entender como a equipe trabalha, como particionam suas tarefas e quais são suas ações quando um novo módulo era designado-los.

Em outro momento foi realizado um *brainstorming,* com ideias do que seria interessante obter automaticamente de uma ferramenta, como quais indicativos e métricas através de dados que o GitHub pode fornecer.

Após a análise do fluxo de tarefas quando um novo módulo é designado, foi realizado a seguinte modelagem do processo da equipe Royal Flush:

Figura 6: Fluxo do processo da equipe Royal Flush



Fonte: Diagrama elaborado pela autora.

4.4 Relação com o Kanban

O Kanban é um sistema de controle de produção adotado pela maioria das equipes ágeis do projeto e-SUS do laboratório Bridge. As equipes populam uma planilha com os dados dos boards do GitHub. Esse processo de população da planilha é feito de forma manual.

Essas métricas além de obter uma visualização do processo da equipe, auxiliam na estimativa de tempo de entregas, no processo de partição das tarefas em partes e menores e

também ajudam a identificar os problemas que aconteceram em determinadas tarefas que saíram do *lead time* esperado. Como um objetivo geral, o Kanban é utilizada para visualização e melhoria de processos nas equipes.

Um dos membros da equipe, que acompanha de perto a implantação do Kanban nas equipes, juntamente com a equipe de gestão, promoveram uma planilha com todos os dados interessantes para serem populados, e então a partir deles serem plotados gráficos de análise. Todos esses indicadores e gráficos serão abordados na próxima seção: Análise de indicadores.

O preenchimento da planilha é feito semanalmente, mesmo quando as equipes optam por *sprints* quinzenais. Análises feitas como rever as possíveis divergências que tenham acontecido, como por exemplo tarefas demorando mais que o esperado, *lead time* médio oscilando muito ou até identificação de gargalos no processo são situações que podem ser medidas através de poucas semanas utilizando o Kanban.

Além das análises que podem ser realizadas em pouco tempo, o Kanban é visado como um dos pontos fortes para fornecer *key results* para certos objetivos da estrutura de OKR do laboratório. Portanto, é uma método para melhorias a curto prazo e para alcance de objetivos a longo prazo.

4.5 Análise dos indicadores

Esta seção explica todos os indicadores utilizados para geração de métricas no kanban. Todas essas informações constam nas planilhas utilizadas pelas equipes ágeis do laboratório e tem embasamento nas métricas abordadas no livro Métricas ágeis (Albino, 2017).

4.5.1 Quantidade de tarefas em cada estado

Esses são os indicadores que são populados semanalmente (de forma manual) na planilha do Kanban, de acordo com o estado das *issues* no *board* da equipe no GitHub.

Ao final de cada semana estipulado pela equipe, é preenchido a quantidade de cada issue nos seguintes estados: Backlog, Análise, Desenvolvimento, Teste e *Done* (*Throughput*). Apesar de esses serem os estados básicos do Kanban, cada board que cada equipe tem suas próprias colunas estilizadas de acordo com o fluxo do seu processo.

A partir desses dados, alguns indicadores são recalculados após cada semana finalizada, são os indicadores: *work in progress*, *throughput acumulado* e coeficiente de variação.

4.5.1.1 Work in progress

As equipes utilizam como WIP (*work in progress)* na planilha do Kanban a soma da quantidade de tarefas da semana que estão em progresso na semana, ou seja, as tarefas de todos as colunas exceto a coluna *done*.

O que algumas equipes também adotam é limitar a quantidade de tarefas em work in progress. Segundo Albino (2017), Se ao acompanhar sistematicamente o fluxo de trabalho do time perceber que as demandas não estão se movendo continuamente ao longo do fluxo, é bem provável que um gargalo esteja se formando em alguma etapa.

A equipe Royal flush limita o WIP utilizando a seguinte fórmula: quantidade de colaboradores da equipe mais dois. Já a equipe Supernova tem a quantidade de tarefas em progresso por número de colaboradores na equipe mais três.

4.5.1.2 Throughput acumulado

Throughput nada mais é do que uma medida que determina quantas unidades de trabalho foram completadas em uma unidade de tempo. Olhando pela perspectiva de vazão (ou saída), é possível afirmar que o throughput é uma medida que determina quanto o fluxo é capaz de processar (Albino, 2017).

O throughput acumulado utilizado pela planilha do kanban nas equipes, é então, a soma de todas as tarefas finalizadas em semanas anteriores e na semana finalizada.

4.5.1.3 Coeficiente de variação

O coeficiente de variação é a divisão do desvio-padrão pela média multiplicado por 100. É sempre dado em percentual. O coeficiente de variação fornece a dispersão dos dados em torno da média em percentual, constituindo uma medida alternativa ao desvio-padrão. Quando se deseja comparar a variabilidade entre dois conjuntos de dados, o coeficiente de variação é a medida de dispersão indicada (Battisti, 2018).

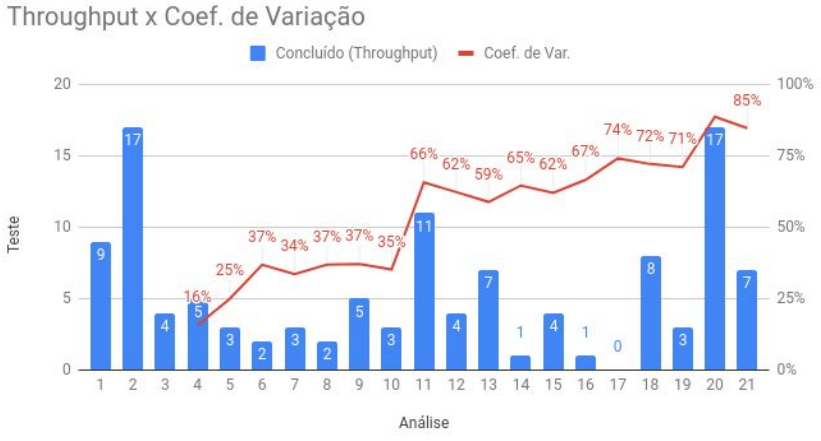
Neste contexto, o coeficiente de variação é uma medida estatística calculada com desvio padrão e média da quantidade de tarefas finalizadas, servindo de grande auxílio para verificar a variação de issues finalizadas ao passar do tempo. Por exemplo, se a equipe tem como objetivo entregas de acordo o *lead time* médio, o coeficiente de variação tende a ficar estável. Porém se por algum motivo, as tarefas finalizadas de uma semana para outra baixaram drasticamente, esse coeficiente tende a variar muito, então as equipes podem parar e analisar os motivos de as entregas não serem constantes.

4.5.2 Throughput x Coeficiente de variação

Para as equipes ágeis que utilizam o Kanban, é determinado como tarefa completada a issue que passou por todas as etapas e foi fechada. A contagem de tarefas concluídas é realizada por semana de acordo com a figura 16.

Na figura abaixo podemos ver as issues finalizadas e o coeficiente de variação ao longo de 21 semanas da equipe Royal Flush:

Figura 7: Gráfico Throughput x Coeficiente de variação



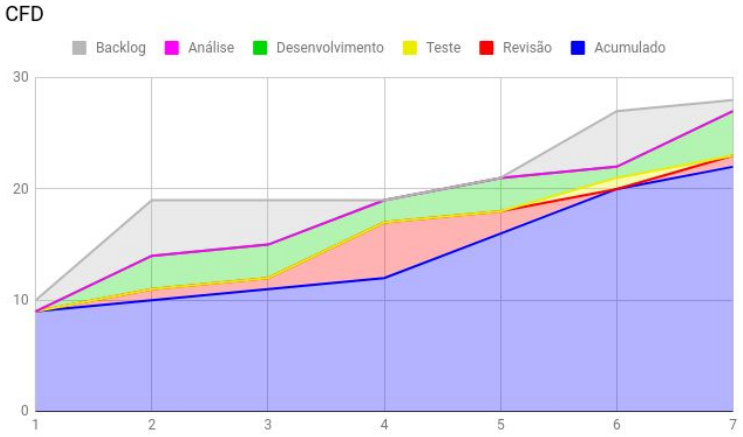
Fonte**:** Planilha Kanban da equipe Royal Flush

4.5.3 Cumulative flow diagram

De acordo com Albino (p. 190, 2017), o CFD (cumulative flow diagram) é uma excelente forma de compreender o fluxo de trabalho ao longo de um processo, afinal, o gráfico pode nos dar um panorama geral do que está acontecendo durante o desenvolvimento de um projeto ou produto. É um instrumento de muito valor no processo de monitoramento em projetos ágeis, pois, usando-o, você poderá rapidamente analisar: quanto de trabalho foi realizado, quanto de trabalho está em progresso e quanto de trabalho ainda precisa ser feito.

A partir da quantidade de tarefas em cada estado por semana, a planilha utilizada plota o seguinte gráfico, como o gráfico CFD da equipe Supernova mostrado a seguir:

Figura 8: Gráfico CFD



Fonte**:** Planilha Kanban da equipe Supernova

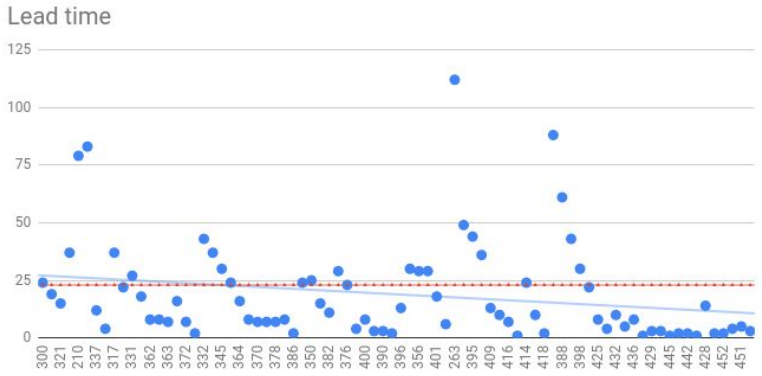
4.5.4 Gráfico de Lead time

Conforme Albino (p.101, 2017), no contexto de desenvolvimento de software, podemos considerar o lead time como sendo o número de dias entre o início e fim do processo de entrega de um item de trabalho (por exemplo, história do usuário, bugs etc.). É o tempo de atravessamento de um item a partir dos limites de entrada e saída definidos no processo de desenvolvimento.

O gráfico de lead time apresenta o tempo em dias de conclusão de cada issue, é uma das principais medidas para análise de tempo de entrega de tarefas. Segundo o gráfico da figura 17, no eixo X é possível observar o número da issue, e no eixo Y o tempo em dias que ela levou para ser completada. Também há uma linha em azul clara que mostra o lead time médio ao decorrer no tempo.

Um indicativo que muitas equipes julgam interessante é o lead time esperado, apontado no gráfico da figura 17 com uma linha vermelha. Nesse caso, a equipe Royal Flush definiu como lead time esperado 23 dias. Dessa forma as equipes podem analisar quanto as tarefas se aproximam ou distanciam da meta esperada.

Figura 9: Gráfico de Lead time



Fonte**:** Planilha kanban da equipe Royal Flush

4.6 Relação com a OKR

A OKR (*Objectives and Key Results*) é uma estrutura utilizada no laboratório Bridge. Foi utilizado no projeto SISMOB, porém tem como objetivo ser utilizado no projeto e-SUS e em todos os demais projetos do laboratório. Como a OKR é alimentada com indicadores de planejamentos de tarefas vindo das equipes, houve um interesse no aprofundamento para uma possível utilização dos indicadores obtidos neste trabalho.

A reunião com o responsável no laboratório pela OKR foi realizada dia 26 de outubro de 2018, onde foi alcançado um entendimento muito maior sobre a grandiosidade desse sistema e seus pontos de entrada. E a partir de então foram realizadas reuniões com as equipes e palestras de alinhamento para que os colaboradores entenderem mais a fundo essa estrutura e os seus benefícios a longo prazo.

4.6.1 Sistema de gestão por OKR

O sistema de gestão por OKR no laboratório Bridge visa um maior alinhamento, transparência e priorização de seus objetivos como organização. O sistema é dividido em três pilares: estratégico, tático e operacional.

Os objetivos estratégicos se realizam com ações de longo prazo. As ações de médio prazo dizem respeito aos objetivos táticos e as ações de curto prazo visam alcançar os objetivos operacionais.

Cada pilar tem seus objetivos, e para esses objetivos serem alcançados é preciso dos *key results* designados para cada um. Para um objetivo pode haver um ou mais *key results.* Os objetivos táticos estão ligados ao Kanban, pois cada equipe, primeiramente separado e depois em conjuntos, fornecerá *key results* para alcance de ações a curto prazo, como melhoria contínua de entregas.

O sistema desenvolvido neste trabalho conclusão de curso, deve então, gerar indicadores para proporcionar a coleta de key results que possam ser utilizados para o alcance dos objetivos do sistema de OKR do laboratório. Portanto, o Kanban auxilia as equipes na melhoria contínua de seus processos, e o OKR auxilia de uma forma geral a análise de melhorias na organização como um todo.

4.7 Levantamento de requisitos

Nesta seção são apresentados os requisitos levantados durante as entrevistas com as equipes. Os requisitos funcionais se fundem em algumas histórias de usuários, completadas por propostas de protótipo de tela. E também são descritos os requisitos não-funcionais, não somente decorrentes de conversas com os integrantes das equipes, mas também resultantes de pesquisas e estudos sobre as tecnologias recentes e adequadas para o desenvolvimento da ferramenta.

4.7.1 Requisitos funcionais

Após a modelagem do fluxo do processo de cada equipe, foi identificado através de eventos intermediários onde seria possível gerar indicadores de forma automática pelo GitHub (Figura 11), e assim realizar avaliações pontuais do processo. A relação de ideias obtidas através dos *brainstorming* com as duas equipes foi filtrada, lapidado e relacionado com os pontos de avaliação dentro do processo.

As equipes utilizam de forma contínua a planilha de métricas do Kanban, portanto foi visto juntamente com as equipes ao longo das entrevistas, quais dados seriam essenciais e que facilitariam os integrantes a popular essa planilha. O resultado desse processo resultou na listagem de requisitos funcionais.

Tabela 1: Requisitos funcionais

RF01	O sistema deve exibir a quantidade de tarefas em cada coluna do projeto da equipe. Essa quantidade deve ser populada semanalmente.
RF02	O sistema deve mostrar a quantidade de tarefas concluídas em uma semana (*throughput*), assim como o cálculo da quantidade acumulada de tarefas concluídas (*throughput acumulado*).
RF03	O sistema deve apresentar a quantidade de tarefas em progresso na semana (*work in progress*), assim como a quantidade total (tanto tarefas em progresso quanto concluídas).
RF04	O sistema deve apresentar o tempo que uma tarefa levou para ser completada (*lead time*), e também o lead time médio das tarefas.
RF05	O sistema deve mostrar a porcentagem de tarefas que ultrapassaram o *lead time* médio.
RF06	O sistema deve obter o valor médio de tarefas em progresso de uma equipe (*work in progres*s médio).
RF07	O sistema deve calcular a porcentagem do módulo pronto, através de tarefas finalizadas com a label referente ao módulo.

Fonte: Elaborado pela autora

Tabela 2: Histórias de usuário

História	Descrição	RF referentes
HU01	Como um integrante de uma equipe ágil, eu quero visualizar as informações semanais disponíveis sobre as tarefas para obter os dados de entrada para a planilha de análise de métricas da equipe.	**RF01, RF02, RF03, RF06**
HU02	Como um integrante da equipe ágil, eu quero visualizar o gráfico de *throughput* das tarefas, para auxiliar na melhoria de estimativas de entrega da equipe.	**RF02**
HU03	Como um integrante da equipe ágil, eu quero observar o gráfico de *lead time* das tarefas, para obter métricas que auxiliam na estimativa de tempo de tarefas.	**RF06, RF07**
HU04	Como um integrante da equipe ágil, eu quero visualizar o gráfico CFD do projeto (Cumulative Flow Diagram), para observar o progresso das tarefas e a identificação de gargalos no processo.	**RF01, RF03**
HU05	Como um integrante da equipe ágil, eu quero saber a quantidade do módulo pronto, a fim de obter métricas que auxiliem na estimativa de entrega e sirvam de insumo para planilhas utilizadas para análise na equipe e na organização.	**RF07**

Fonte: Elaborado pela autora

4.7.2 Requisitos não funcionais

Após um estudo sobre as ferramentas existentes de integração ao GitHub, foram levantados algumas questões técnicas essenciais para o desenvolvimento da ferramenta, como linguagem, API e biblioteca.

Também nas reuniões realizadas com as equipes surgiram sugestões que podem tornar a utilização da ferramenta mais prática e dinâmica, como o desenvolvimento de um GitHub app, ou seja, o sistema se tornar uma própria extensão da plataforma GitHub.

Tabela 7: Requisitos não funcionais

RNF01	O sistema deve ser web. Compatível com dispositivos *desktops,* nos navegadores *Google Chrome* e *Mozilla FIrefox.*
RNF02	O sistema deve ser integrado à plataforma GitHub, utilizando a *GitHub REST API v3.*
RNF03	O sistema deve ter uma estrutura de banco de dados para armazenar os valores. O banco deve ser PostgreSQL.

RNF04	O sistema deve utilizar o interpretador NodeJS v10.13.
RNF05	O sistema deve ser desenvolvido em JavaScript.
RNF06	O sistema deve utilizar o *express*, um *framework* de servidor padrão para o Node.js.

Fonte: Elaborado pela autora

4.8 Arquitetura geral da ferramenta

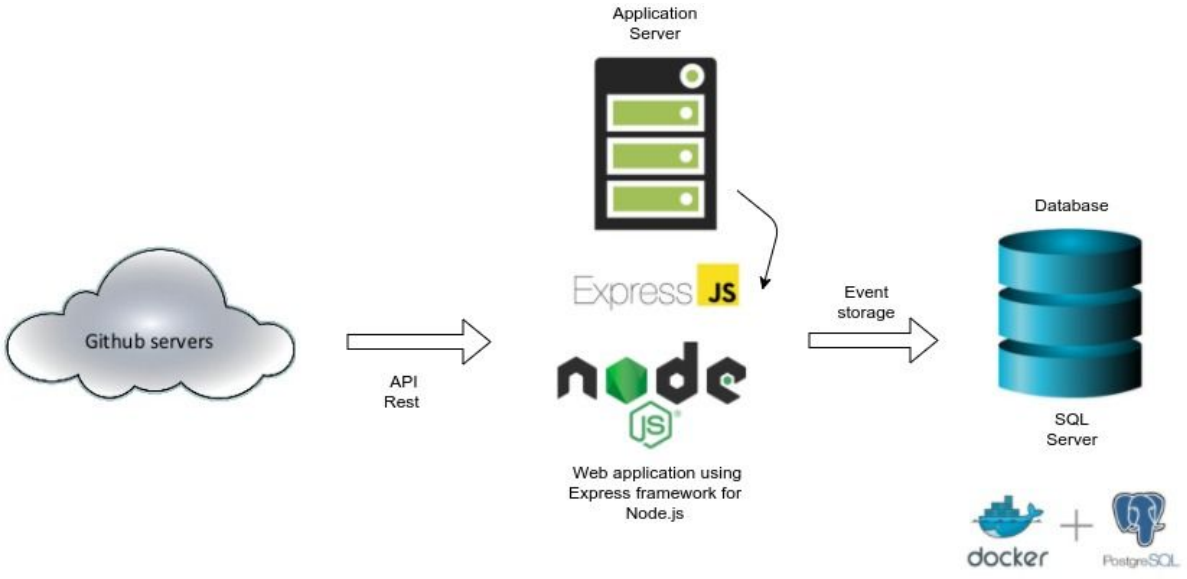
A ferramenta é uma aplicação Web criada com Express, um framework que utiliza Node.js para construções de sistemas. Portanto essa aplicação utilizará JavaScript tanto na parte *backend* quanto na parte *frontend*.

A comunicação da aplicação com o GitHub é realizada através da GitHub API v3, uma API em REST. Onde através de chamadas POST da aplicação, a API retorna os valores requeridos em uma estrutura JSON.

Os valores serão salvos em um banco de dados, este criado com auxílio da ferramenta docker, para salvar todos os estados do banco de dados. O banco de dados é PostgreSQL, ou seja, um banco de dados estrutural, portanto os dados que vem da requisição em JSON terão de ser tratados antes de serem salvos.

Os dados serão salvos no banco através de rotinas, e a cada vez que o usuário utilizar a aplicação, será feito consultas no banco para trazer os dados filtrados.

Figura 10: Arquitetura geral do sistema



Fonte: Elaborado pela autora

5 Desenvolvimento

Neste capítulo é apresentado o desenvolvimento da ferramenta que obtém indicadores automatizados através da plataforma GitHub, chamada de *AppIssue* (uma junção de *application* com *issues*).

É apresentada a descrição de todos os passos utilizados, as principais decisões de projeto, as dificuldades encontradas e as superadas, as tecnologias utilizadas e também exemplo de funcionalidades implementadas.

5.1 Configuração do ambiente

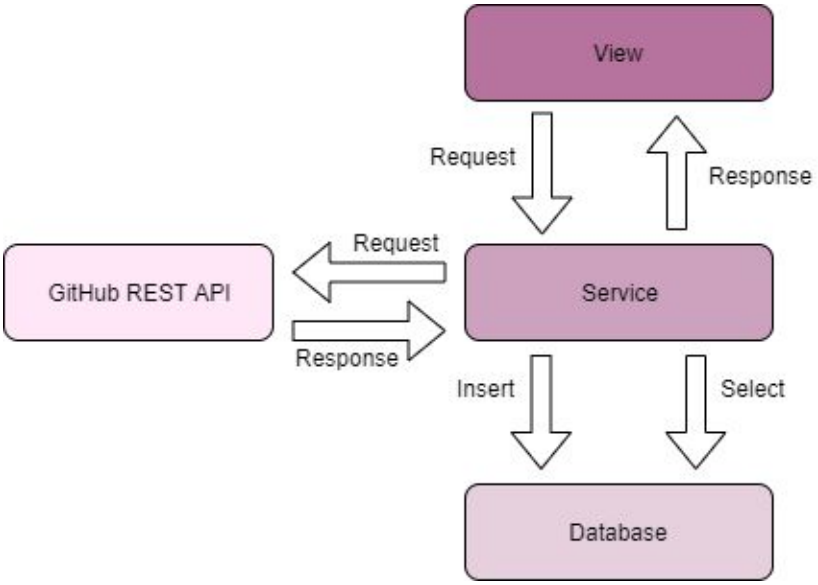
A IDE escolhida para desenvolver a aplicação foi o Visual Studio Code[15]. Primeiramente foi criado uma base para uma aplicação Node.js utilizando Express, do mesmo modo que no primeiro teste. Depois foram instaladas todas as dependências e feita a configuração da aplicação para utilização da biblioteca React[16].

Para configuração do banco de dados, foi utilizado o Docker[17], no qual, por meio de alguns comandos de configuração foi criado o container e a imagem de um banco de dados postgres.

Na camada view, onde são implementadas as telas do sistema, foi utilizado TypeScript - JavaScript com alguns recursos a mais como tipagem. Essa escolha foi dada em função de que os componentes do Design System *bold* são desenvolvidos dessa forma.

Na camada service é onde é realizada a comunicação com a API REST v3 do GitHub, e também é onde são inseridas e consultadas as informações no banco de dados. Para entender melhor essas três camadas a figura 21 mostra uma visão geral da arquitetura da aplicação.

Figura 11: Arquitetura geral das camadas da aplicação



Fonte: Elaborado pela autora

[15] https://code.visualstudio.com/
[16] https://pt-br.reactjs.org/
[17] https://www.docker.com/

Nas próximas seções é detalhado cada uma das camadas desenvolvidas: comunicação com a GitHub API, Banco de dados, Serviço e View.

5.3 GitHub API

O GitHub fornece uma interface para desenvolvedores que desejam desenvolver aplicativos integrados com a plataforma, chamada de API REST que está na versão 3. Por padrão, todas as solicitações para *https://api.github.com* recebem a versão v3 da API REST.

Neste trabalho são realizadas requisições do tipo GET para a API do GitHub. Para cada evento será enviada a requisição para um endereço específico, podendo conter header de Accept, conforme descrito na documentação da API do GitHub.

A fim de testar as requisições para API REST e observar os resultados, sem precisar fazer toda a implementação logo de início, foi utilizado o software *insomnia*[18].

5.3.2 Implementação das requisições

Na aplicação foi implementado um método genérico que sintetiza os passos feitos anteriormente. Como parâmetro é passado o endereço da requisição GET. É definido o username, o token (password), os headers (foram adicionados todos os headers que seriam necessários de uma vez só) e o tipo da autorização.

Para realizar as requisições é utilizado o axios[19], dessa forma o método chama uma função get através do axios, com a url passada por parâmetro e os outros parâmetros definidos.

Figura 12: Método geral de requisição à API

```
async function getGitURL(url) {
  let data;
  const username = "";
  const password = "";
  const headers = {
    Accept:
      "application/vnd.github.inertia-preview+json, application/vnd.github.symmetra-preview+json, " +
      "application/vnd.github.machine-man-preview, application/vnd.github.starfox-preview+json",
    Authorization: "Basic " + Base64.encode(username + ":" + password)
  };
  try {
    const response = await axios.get(url, { params: {}, headers });
    data = response.data;
  } catch (error) {
    console.error(error);
  }
```

Fonte: Elaborado pela autora

[18] https://insomnia.rest/
[19] https://github.com/axios/axios

5.3.3 Métodos utilizados

A fim de obter diferentes informações sobre o repositório, deve-se aplicar uma url específica no método GET da requisição. Esses endereços estão definidos na documentação da API REST GitHub v3. Para esta aplicação foram utilizadas as seguintes url abaixo:

Para obter cada board (equipe) do repositório:

"https://api.github.com/repos/laboratoriobridge/pec/projects"

Para cada coluna de cada equipe:

"https://api.github.com/projects/" + id da equipe + *"/columns"*

Os cards (issues) de cada coluna da equipe:

"https://api.github.com/projects/columns/" + id da coluna + *"/cards"*

Lista de eventos ocorridos no repositório, separados por página:

"https://api.github.com/repos/laboratoriobridge/pec/issues/events?page=" + nº da página

5.4 Banco de dados

O banco de dados utilizado para desenvolver a aplicação foi o PostgreSQL. Para realizar a configuração inicial foi utilizado o Docker. Após essa configuração, foi realizada a conexão do banco de dados no DBeaver[20], ferramenta onde é feita as migrações do banco, assim como testes, consultas e alterações.

Depois de criado o banco localmente, foi feita a conexão desse banco com a aplicação. Foi utilizada a função *client* do JavaScript para realizar essa integração, assim como está descrito abaixo. No código abaixo, primeiro foram definidas as informações básicas como a porta, *host*, *user/password* e o nome do banco. Na segunda parte, é feita a chamada connect para o *client*, concluindo a conexão com o banco.

Figura 13: Função conexão com o banco

```
const client = new Client({
  host: "localhost",
  port: 5436,
  user: "root",
  password: "",
  database: "appissue"
});

client.connect(err => {
  if (err) {
    console.error("connection error", err.stack);
  } else {
    console.log("connected");
  }
});
```

Fonte: Elaborado pela autora

[20] https://dbeaver.io/

Foram então criadas as tabelas e as colunas no banco de acordo com cada dado coletado pelas requisições. A principal tabela na modelagem é a tabela “eventos”, onde é salvo todos os eventos do repositório, relacionando com cards, issues, colunas e board.

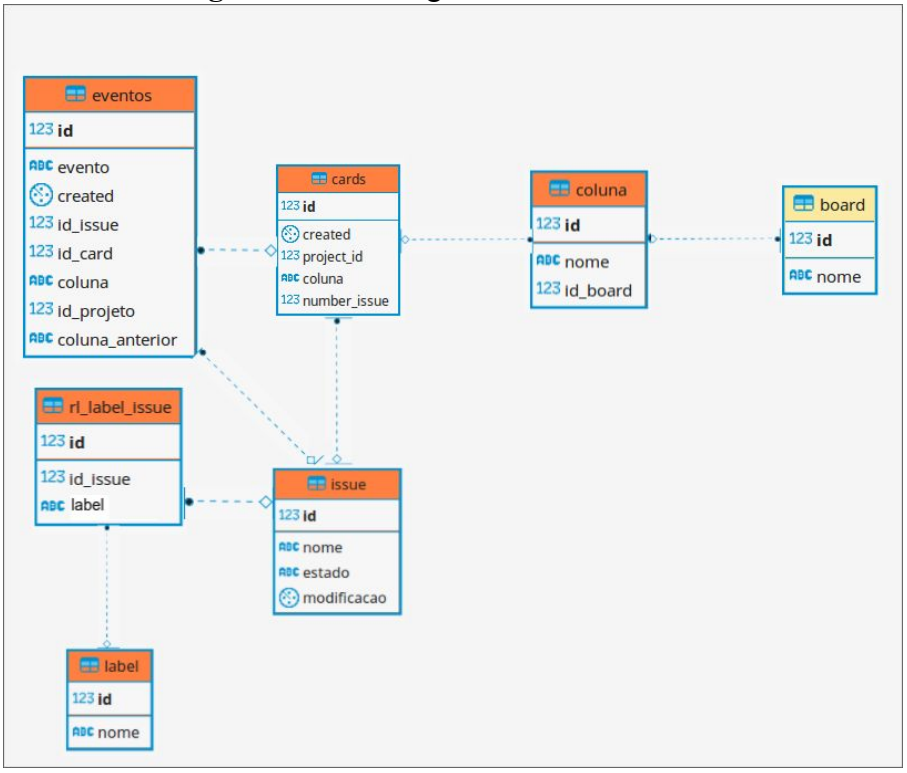
Existe uma tabela para “cards” e outra para “issues” pelo motivo que o card sempre está em um board, portanto sempre está atrelado à uma equipe, o que não acontece em uma issue que não vira um card.

Também foram criado as tabelas: “board”, com todas as equipes, a tabela “coluna”, que contém todas as colunas de cada board, e a tabela “label”, onde está todas as labels do repositório.

A tabela “rl_label_issue” relaciona a issue com as labels, ou seja, por esta tabela é possível saber quais labels estão atreladas em cada issue.

A modelagem completa do banco de dados utilizado por essa ferramenta pode ser observado na figura abaixo.

Figura 14: Modelagem do banco de dados



Fonte: Elaborado pela autora

5.5 Serviço

Na camada de serviço, onde a classe principal é a server.js, é onde são realizadas as operações com o banco de dados. Alguns dos métodos implementados nessa classe já foram explicados anteriormente, como a requisição para a API REST do GitHub e a conexão com o banco de dados.

Além desses métodos, o resultado de cada requisição específica é inserido no banco de dados. E também, o service realiza consultas ao banco de dados toda vez que a cama view faz uma requisição.

5.5.1 Inserção de dados

As funções que inserem dados, sempre utilizam o retorno da resposta do método de requisição à API REST do GitHub. Depois disso, para cada item coletado, é feita a inserção no banco de dados, passando um ou mais parâmetros no meio da query.

No trecho de código abaixo, é inserido no banco todas as colunas de cada projeto (passado por parâmetro) do repositório. É utilizado o método *client.query()* para fazer a integração do trecho sql com o banco de dados.

Figura 15: Método de inserção de colunas

```
async function insertColumns(projectId) {
  const colunas = await getColunas(projectId);
  console.log(colunas);
  colunas.map(item => {
    var sqlSelect = "SELECT board.id FROM board WHERE id = $3";
    var sql =
      "INSERT INTO coluna (id, nome, id_board) VALUES ($1, $2, (" +
      sqlSelect +
      "))";
    client.query(sql, [item.id, item.name, projectId], function(err, result) {
      try {
        if (err) throw err;
        console.log("Number of records inserted: " + result.affectedRows);
      } catch (error) {
        console.log(error);
      }
    });
  });
}
```

Fonte: Trecho de código elaborado pela autora

5.5.2 Consulta de dados

Toda vez que a view faz uma requisição de dados para uma url (explicado mais detalhadamente na próxima seção sobre a camada *view*), o método app.get() no *service* com a url referente é invocado.

Esse método é então executado, podendo conter ou não parâmetros (chamados de request) trazidos pela view. A resposta da query é enviado de volta para a view através do *response.send()*.

No trecho de código abaixo, ao receber uma requisição da view com o endereço "/col", o método *colunasExpected()* que recebe o id da equipe é executado. É realizado o select passado como parâmetro o resultado do request (id da equipe), e o r*esponse.send()* envia o retorno do select para a view.

Figura 16: Método de consulta de colunas de uma equipe

```
async function colunasExpected(request, response) {
  var sql = "SELECT coluna.nome FROM coluna WHERE coluna.id_board=$1";
  client.query(sql, [request.query.id], function(err, result) {
    try {
      if (err) throw err;
    } catch (error) {
      console.log(error);
    }
    response.send({ result });
  });
}

app.get("/col", colunasExpected);
```

Fonte: Trecho de código elaborado pela autora

Contudo, algumas consultas se tornaram um pouco complexas devido ao fato das regras do Kanban e também pelo fato que impactou no sistema como um todo: cada equipe tem um número e nome de colunas diferentes.

A consulta SQL abaixo retorna a quantidade de issue em cada coluna de cada equipe em um determinado período. É a consulta que retorna as informações bases para um gráfico CFD ou então para as análises de métricas semanais que as equipes realizam. O parâmetro $1 é o id da equipe, e o $2 e $3 são as datas de início e data de fim do período, respectivamente. Essas datas são definidas pelo usuário na *view*.

Figura 17: Consulta issues em cada coluna por período

```
select
    coluna,
    count(coluna)
from
    (
    select
        max(id) as id,
        id_issue
    from
        eventos as e
    where
        created between to_date( $2, 'dd/mm/yyyy') and to_date($3, 'dd/mm/yyyy')
        and id_projeto = $1
        and evento not in( 'closed',
        'reopened',
        'lebeled',
        'unlabeled',
        'removed_from_project')
        and id_issue not in(
        select
            distinct ev.id_issue
        from
            eventos as ev
        join cards on
            cards.number_issue = ev.id_issue
        where
            evento like 'closed'
            and ev.created < to_date($2, 'dd/mm/yyyy')
            and cards.project_id = $1)
    group by
        e.id_issue) as sub
join eventos on
    sub.id = eventos.id
group by
    coluna
```

Fonte: Trecho de código elaborado pela autora

5.6 *View*

Como mencionado anteriormente, o frontend da aplicação é feito na linguagem *Typescript*, uma especialização do JavaScript com tipagem de objetos. A aplicação também é desenvolvida utilizando a tecnologia *React*, pois é uma das bibliotecas atualmente mais utilizadas, utilizada pelo *Facebook, Instagram, Netflix* e outras grandes plataformas.

As classes na view são dividas por funcionalidade, e a classe App é a principal, é essa classe que o index.js chama. A partir da classe App, as outras funcionalidades são chamadas passando os parâmetros necessários de cama uma.

A comunicação da view com o backend, a fim de obter os valores para mostrar na tela, se dá da seguinte maneira: é utilizado um método *axios.get()* passando por parâmetro a url em que o método no backend será invocado, como foi explicado anteriormente. A resposta do método invocado no service é então colocada em uma *promise* após a chamada

then() do método na view. É utilizado o *useEffect()* para a requisição ocorrer só uma vez e não ficar replicando os dados infinitamente. Também é possível passar um argumento no final do *useEffect()* para este ser executado novamente quando esse valor mudar.

No trecho de código abaixo, é feita a chamada da view para o backend para trazer todas as equipes. Como essa funcionalidade é chamada uma vez apenas, quando a tela é renderizada pela primeira vez, não é necessário passar argumentos após o *useEffect()*.

Figura 18: Função na view que faz a requisição das equipes para o *service*

```
export default function Equipes() {
  const [projetos, setProjetos] = useState<Projeto[]>();
  const [statusSelecionado, setstatusSelecionado] = useState<Projeto>();

  useEffect(() => {
    axios.get("/equipes").then(resp => {
      setProjetos(resp.data.res.rows);
    });
  }, []);
```

Fonte: Trecho de código elaborado pela autora

Já no trecho abaixo, a view faz a requisição de todas issues abertas para o *service*. E a cada vez que o usuário muda a equipe (também chamada de *project* ou projeto como é visualizada no GitHub) esses dados precisam ser atualizados, por esse motivo é passado o argumento no final da função *useEffect()*.

Figura 19: Função na view que faz a requisição das issues abertas por equipe para o *service*

```
export const Abertas = (props: AbertasProps) => {
  const [abertas, setAbertas] = useState<Abertas[]>();
  const { classes } = useStyles(createStyles);

  useEffect(() => {
    axios.get("/abertas", { params: { id: props.projeto.id } }).then(resp => {
      console.log("abertas: ", resp.data.result.rows);
      setAbertas(resp.data.result.rows);
    });
  }, [props.projeto]);
```

Fonte: Trecho de código elaborado pela autora

5.6.1 *Design System*

O laboratório Bridge possui seu próprio Design System, o *bold*, que é público e pode ser importado para qualquer projeto em React. O Design System já é utilizado no redesign do projeto e-SUS e no sistema de gestão de pessoas interno do laboratório.

Todos os componentes utilizados para o desenvolvimento da aplicação foram importados da biblioteca do *bold,* que são especificados com exemplos no próprio site do Design System (*https://bold.bridge.ufsc.br/*).

5.6.2 Gráficos

A biblioteca utilizada para a geração dos gráficos na ferramenta, foi a biblioteca do Google Charts[21] para aplicações em React.

[21] https://www.npmjs.com/package/react-google-charts

Para utilizar essa biblioteca, é necessário importar e instalar o *package* no projeto para ter acesso aos componentes definidos como <Chart />. Também é necessário definir os dados para preencher o gráfico.

No trecho de código abaixo, é apresentado a função que preenche um gráfico de *issues* por *lead time,* juntamente com a média desses dias. Essa função percorre todas as issues fechadas e preenche o *array* de dados que possui o id da issue e os dias calculados de *lead time.*

Figura 20: Função que preenche os dados do gráfico de *lead time*

```
function fillData(fechadas: Fechadas[]) {
  const data: any[] = [];
  let acum = 0;
  let i = 0;
  for (; i < fechadas.length; i++) {
    let item = fechadas[i];
    const days = calculaLeadTime(item);
    acum += days;
    data.push([
      "Issue nº: " + item.id + " - " + item.nome,
      days,
      days !== 0 ? acum / (i + 1) : days
    ]);
  }
  return [["Issues", "Dias", "Media"], ...data];
}
```

Fonte: Trecho de código elaborado pela autora

5.7 Funcionalidades

Nesta seção é apresentado o fluxo de utilização da ferramenta, com imagens da ferramenta desenvolvida. São mostradas as funcionalidades na versão atual da ferramenta, pretende-se atualizar essas imagens até a entrega final, com a versão mais estável da ferramenta.

1: Selecionar a equipe ágil que pretende-se obter as métricas

Figura 21: Combo de seleção da equipe

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

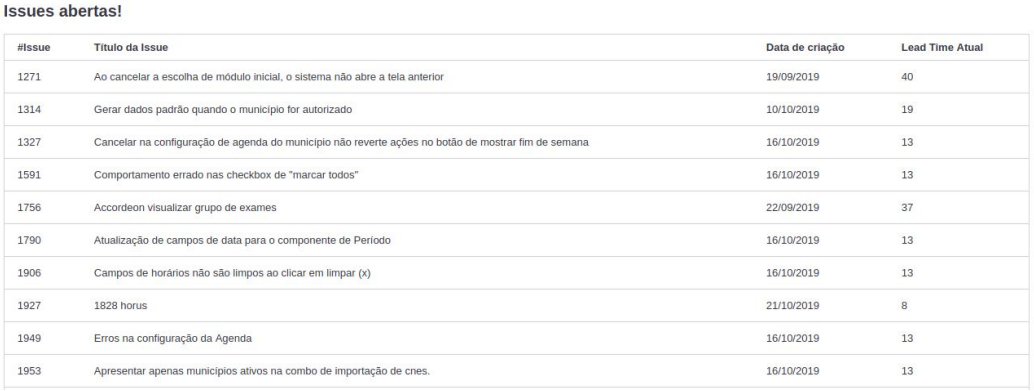
2: Selecionar o período em que pretende-se obter informações sobre os estados das tarefas em cada coluna do *board* da equipe (referente à **HU01**).

Figura 22: Quantidade de tarefas em cada coluna filtrada por período

Fonte: *Screenshot* da ferramenta desenvolvida pela autora

3: Visualização do *lead time* das issues abertas no *board* da equipe (referente à **HU03**).

Figura 23: Tabela de *issues* no estado aberta



Fonte: *Screenshot* da ferramenta desenvolvida pela autora

4: Visualização do *lead time* das issues fechadas no *board* da equipe (referente à **HU03**).

Figura 24: Tabela de *issues* no estado fechada

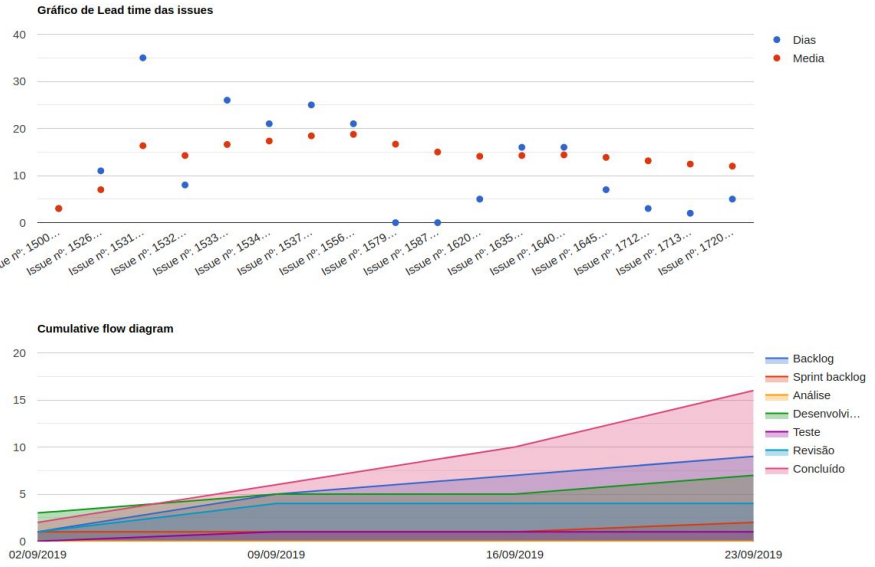
Fonte: *Screenshot* da ferramenta desenvolvida pela autora

5: Visualização dos gráficos:

- Gráfico de lead time com as issues fechadas representadas em azul e a média acumulada representada em vermelho (referente à **HU03**).
- Gráfico *Cumulative Flow Diagram* representando a quantidade de tarefas acumuladas separadas por colunas no *board* da equipe (referente à **HU04**).

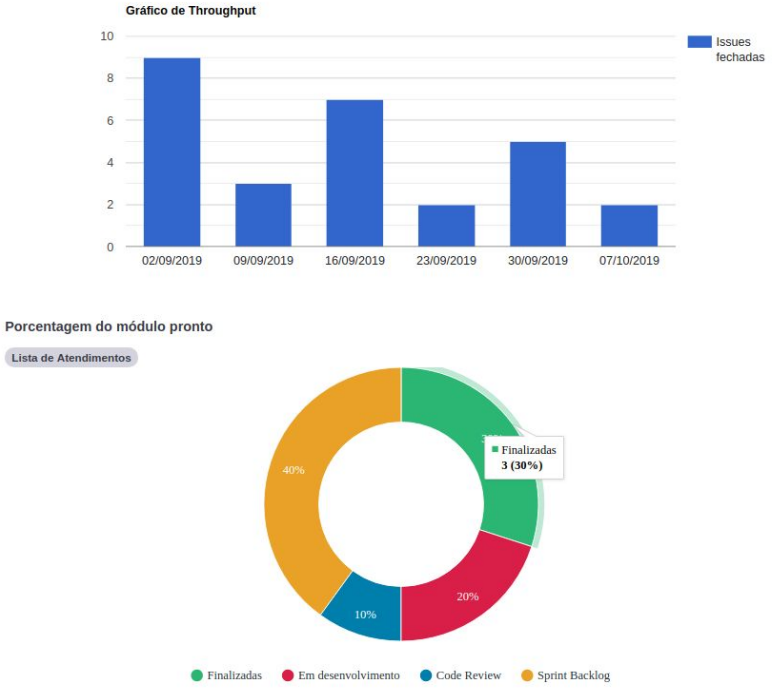
- Gráfico de Throughput, mostrando a quantidade de issues fechadas por semana (referente à **HU02**).
- Gráfico de Porcentagem do módulo pronto, apresentando quantas tarefas pertencentes àquela label estão em cada estado (referente à **HU05**).

Figura 25: Gráficos de *lead time* e CFD



Fonte: *Screenshot* da ferramenta desenvolvida pela autora

Figura 26 - Gráficos de Throughput e Porcentagem do módulo pronto



Fonte: *Screenshot* da ferramenta desenvolvida pela autora

6. Avaliação

Este capítulo apresenta a avaliação da utilidade e da funcionalidade da ferramenta desenvolvida, com enfoque no ambiente de desenvolvimento ágil. A avaliação engloba participantes de equipes ágeis do laboratório Bridge, e também a gerência de desenvolvimento de projetos.

Primeiramente são definidos os objetivos da avaliação, e na sequência são apresentadas as motivações para a escolha das técnicas de avaliação adotadas. Logo após, são apresentadas a execução da avaliação e a análise dos resultados.

6.1 Objetivos

A avaliação tem como principal objetivo:

> Avaliar a funcionalidade e utilidade da ferramenta de coleta de métricas ágeis integrada à plataforma GitHub, sob o ponto de vista da gerência de desenvolvimento e membros de equipes ágeis.

Desta forma, em termos de funcionalidade e utilidade, pretende-se avaliar a qualidade do sistema. A funcionalidade estabelecida precisa ser satisfeita pelo software quando é usado em condições específicas, devendo ser adequada, precisa e segura (ISO/IEC 29110-5-1-2, 2011). Conforme a ISO/IEC/IEEE 24765 (2017), um ferramenta de software possui utilidade quando ela é projetada para executar alguma função de suporte frequentemente usada.

Inicialmente, é avaliada a utilidade da ferramenta num contexto geral em ambientes de desenvolvimento ágil, e como ela poderia agregar na visualização de melhoria de processos e planejamento das equipes.

Após essa etapa, é avaliada a funcionalidade, focando nas informações que são trazidas e realizando uma verificação englobando os requisitos funcionais. A última etapa consiste em considerações do usuário, como críticas e sugestões.

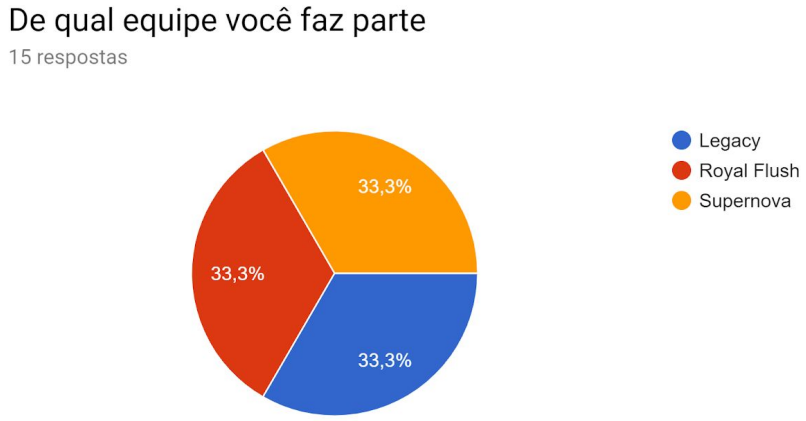
6.2 Análise dos resultados

Nas seções a seguir é apresentada a análise dos resultados obtidos pela aplicação no *survey*, separados por equipes ágeis do laboratório e gerência de desenvolvimento.

6.2.1 Equipes ágeis

A avaliação feita com três equipes ágeis contou com a participação de quinze usuários no total (Figura 38), sendo cinco participantes de cada equipe. Os usuários assumem diferentes funções dentro das equipes, como análise, desenvolvimento de software e teste de qualidade.

Figura 27: Participantes por equipe que responderam o questionário



Fonte: Resultado do Google forms elaborado pela autora

Nas seções abaixo 6.4.1.1 e 6.4.1.2 será feita uma análise das respostas separadas por características. As respostas são classificadas de 1 a 5, sendo o 1 "Discordo totalmente" até 5 "Concordo totalmente".

6.2.1.1 Sobre a utilidade

Pergunta 1: Você considera a ferramenta útil para o autogerenciamento de sua equipe no contexto de desenvolvimento ágil?

No contexto de desenvolvimento ágil, 93,3% dos usuários concordam totalmente (nível 5) que a ferramenta é útil para o autogerenciamento de sua equipe, enquanto 6,7% escolheram 4. Na tabela 9 a seguir apresentados os comentários dos usuários sobre a pergunta 1.

Tabela 3: Comentários sobre a pergunta 1 do questionário de equipes ágeis

Utilizamos o Kanban dentro de nossa equipe, porém fazemos a coleta de métricas tudo manualmente. A ferramenta automatiza alguns processos da equipe
Ao coletar e exibir dados relevantes de forma rápida, a ferramenta demonstrou ser capaz de tornar o nosso processo de autogerenciamento mais eficiente.
Poupa tempo ter uma ferramenta que já retira essas informações diretamente do github
Achei a funcionalidade de ver o histórico do CFD muito interessante. Acho importante pensar no cenário dos times usarem um project a nível de organização, não de repositório (por exemplo, ao invés do project ficar no repositório PEC, ele fica na organização laboratório bridge). Alguns times compilam os dados de sua sprint considerando todos os repos que trabalharam.
A ferramenta é muito útil para a equipe, pois ela dá um apoio no que diz respeito à coleta e síntese das principais informações sobre as tarefas da equipe

É útil para gerar indicadores do desenvolvimento das tarefas

Fonte: Respostas obtidas do Google forms

Pergunta 2: Você acha que a ferramenta na sua forma atual é prática para a sua equipe em seu dia-a-dia?

Em relação ao uso no dia-a-dia de uma equipe ágil, 80% responderam que concordam totalmente que a ferramenta é prática, enquanto 20% (nível 4) concordam com a praticidade da aplicação.

Pergunta 3: Você acha que a ferramenta oferece informações suficientes para apoio ao desenvolvimento do método Kanban dentro da sua equipe?

Sobre o apoio ao desenvolvimento do método Kanban em sua equipe, 40% concordam totalmente que a ferramenta fornece informações suficientes, enquanto 53,3% concordam e 6,7% nem concorda e nem discorda . Na tabela 10 abaixo é apresentado os comentários dos usuários sobre a pergunta 3.

Tabela 4: Comentários sobre a pergunta 3 do questionário de equipes ágeis

Nossa equipe está bem madura no Kanban, estamos coletando mais dados que a ferramenta ainda não tem.
Adicionar um valor do working in progress pra semana filtrada e também um gráfico para avaliar o coeficiente de variação também são informações legais para o Kanban, mas acho que as principais já existem que é o cfd e o gráfico de lead time
Poderia ter a data de criação e de fechamento da issue na lista de issues fechadas.
Ele se complementa com a planilha. Inclusive acho interessante essa ferramenta absorver a planilha e ser integrada no Meu Bridge.
Ainda faltam algumas poucas informações, porém ela facilita muito o trabalho da equipe.

Fonte: Respostas obtidas do Google forms

Pergunta 4: O conjunto de informações fornecidas pelo sistema é suficiente para realizar o planejamento, como estimativa de entregas e partição de tarefas na sua equipe?

Com relação ao planejamento, 20% dos usuários concordam totalmente que o conjunto de informações fornecidas é suficiente, 73,3% concordam e 6,7% nem concordam nem discordam.

A partir de uma análise mais detalhada das respostas apresentadas acima é possível ter algumas ideias sobre a utilidade do sistema:

- A ferramenta pode ser útil no ambiente de desenvolvimento, pois maioria dos usuários concordam com isso. Principalmente no que diz respeito a autogerenciamento, planejamento e observações de possíveis melhorias no fluxo, relacionados às perguntas 1 e 4.

- Os usuários acreditam que a ferramenta seria útil na implantação do método Kanban na equipe, portanto é possível inferir que a ferramenta tem alguma utilidade nesse contexto. Mesmo alguns usuários comentando que necessitam de mais informações do que as que estão disponíveis, obteve-se alguns comentários positivos no que diz respeito a coleta de informações para o desenvolvimento do método Kanban, como pode-se observar nas tabelas 9 e 10.
- É possível que a ferramenta seja útil e prática, pois além da utilidade, a maioria dos usuários concorda totalmente na praticidade da ferramenta, como é possível perceber nas respostas da pergunta 2.

6.4.1.2 Sobre a funcionalidade

Pergunta 5: De acordo com as diretrizes do método Kanban utilizado na sua equipe ágil, você acha que as informações disponibilizadas são suficientes?

Com relação às diretrizes do método Kanban, 20% dos usuários concordam totalmente que as informações disponibilizadas são suficientes, 73,3% concordam e 6,7% nem concordam nem discordam.

Pergunta 6: Você considera que o sistema disponibiliza informações suficientes para a aferição de quantidade de tarefas em cada coluna do *board*?

Sobre as tarefas em cada coluna do *board* da equipe, 80% concorda totalmente que as informações disponibilizadas são suficientes, 13,3% apenas concorda e 6,7% nem concorda e nem discorda.

Pergunta 7: Você considera que a ferramenta possui funcionalidades suficientes para automatizar um processo que hoje é manual?

60% dos usuários concordam totalmente que com as funcionalidades da ferramenta é possível automatizar algum processo manual, enquanto 40% concordam. Os comentários abaixo na tabela 11 dizem respeito à quais processos manuais a ferramenta poderia auxiliar.

Tabela 4: Comentários sobre a pergunta 7 do questionário de equipes ágeis

Contagem do Lead time, Contagem de quantas tarefas tem em cada coluna
Coleta do lead time médio da equipe
A ferramenta nos facilita a visibilidade do lead time de cada issue, que dentro de minha equipe é coletado semanalmente, toda segunda-feira, de forma manual. Fazemos essa coleta para avaliar as issues individualmente e verificar se alguma passou do tempo médio da equipe, para tentar identificar problemas. Nesse ponto a ferramenta seria um facilitador para a equipe. O gráfico do CFD também permite a detecção de anomalias no nosso Kanban e observar os gargalos de certas áreas, como desenvolvimento e testes.

Melhoria do processo em geral; Identificação de gargalos; Geração automática dos gráficos de lead time e CFD.
A construção do gráfico de lead time e o CFD
O cálculo do CFD e do lead time.
Atualmente alguém precisa entrar no histórico de issues do GitHub e verificar quais tarefas foram fechadas e quanto tempo foi necessário para fechá-las, com a ferramenta, este processo fica automático
Preenchimento semanal da planilha do Kanban utilizada pela equipe.
Passar os dados do board na mão
Obtenção dos dados das issues para o Kanban

Fonte: Respostas obtidas do Google forms

Pergunta 8: Você acha que a ferramenta possui informações suficientes para a observação do *lead time* das tarefas?

Com relação ao *lead time* das tarefas, 80% concorda totalmente que a ferramenta possui informações suficiente, enquanto 20% apenas concorda.

Pergunta 9: Você considera que a ferramenta indica corretamente as tarefas que se encontram no estado de *work in progress*?

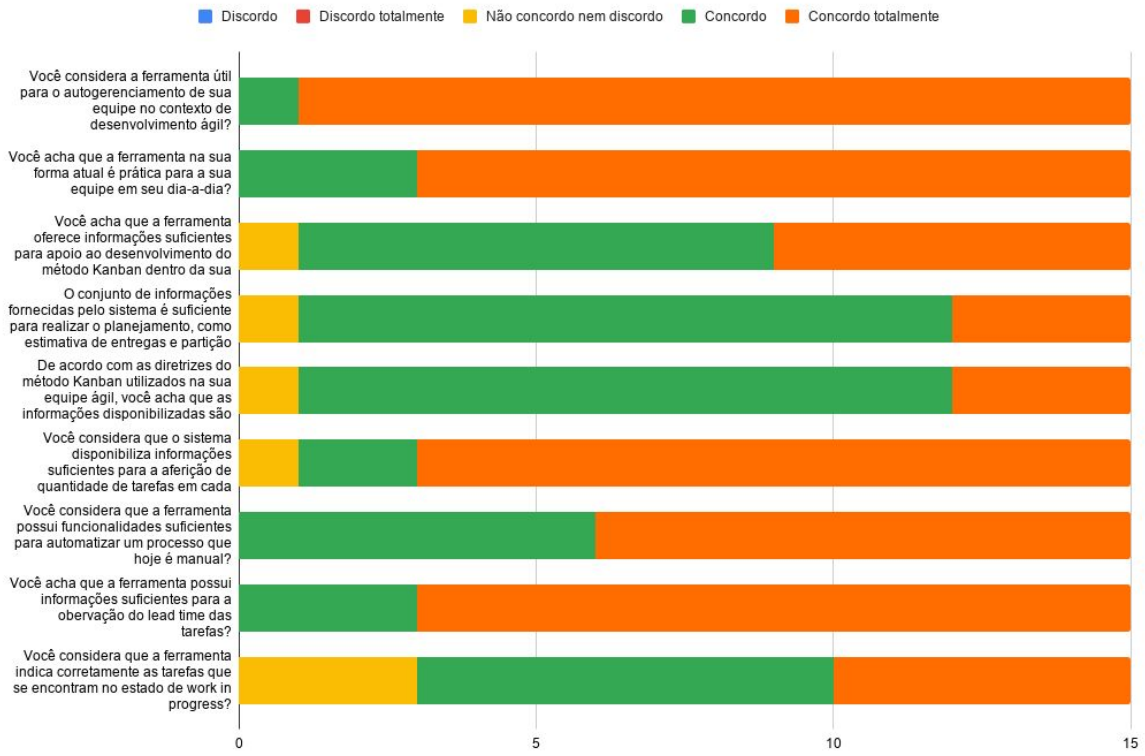
Com relação ao estado das tarefas, 33,3% concorda totalmente que a ferramenta indica as tarefas que estão em *work in progress*, 46,7% apenas concorda e 20% nem concorda e nem discorda.

A partir de uma análise mais detalhada das respostas sobre a funcionalidade da ferramenta, é possível ter algumas ideias sobre o sistema:

- A maioria dos usuários concorda que a ferramenta possui funcionalidades que permitem a aferição de métricas como *lead time,* tarefas em *work in progress* e quantidade de cada tarefa em cada coluna do *board.* Conforme pode-se observar nas respostas das perguntas 6, 8 e 9. Portanto, é possível que o sistema possa auxiliar na coleta dessas métricas ágeis.
- A aplicação pode fornecer a automatização de processos manuais, pois a maioria dos usuários concorda totalmente que a ferramenta oferece funcionalidades para isso, conforme respostas da pergunta 7. Dentre os processos citados pelos usuários (tabela 11), estão a contagem de tarefas do *board*, a geração do gráfico de cumulative flow diagram (CFD) e a contagem do *lead time*.
- É possível inferir que apesar da ferramenta obter a maior parte das métricas necessárias para a implantação do Kanban nas equipes, ainda faltam algumas informações, pois a maior parte dos usuários apenas concorda com isso, conforme respostas à pergunta 5. Algumas equipes

que tem o processos mais amadurecido sentem falta de algumas informações, como pode-se perceber na tabela 10.

Figura 28 - Respostas do questionário de avaliação



Fonte: Elaborado pela autora

6.4.2 Gerência de desenvolvimento

Com relação às três primeiras perguntas (tabela 8), o Gerente de Desenvolvimento entrevistado concorda sobre a utilidade, praticidade e coleta de métricas utilizadas no Kanban fornecidas na ferramenta.

Na resposta da quarta pergunta, o gerente nem concorda e nem discorda que as informações fornecidas sejam suficientes para realizar o planejamento como estimativa de entrega e partição de tarefas. Ele comenta em um trecho que, “Planejamento e estimativas dependem mais de uma mudança cultural significativa na equipe do que acesso às estatísticas”, além disso destaca que as informações fornecidas ajudam as equipes a saber se estão em um caminho certo.

O gerente discorda que a ferramenta seja útil para mensurar o desempenho das equipes ágeis (pergunta 5). Ele conclui que desempenho é algo relativo, pois depende da forma que as equipes fragmentam as tarefas, podendo às vezes poucas tarefas agregar mais valor que muitas.

No papel de gerente de desenvolvimento, ele concorda totalmente que a ferramenta seja útil para diagnosticar possíveis gargalos ou oportunidades de melhoria no fluxo de trabalho (pergunta 6).

Na última pergunta em aberto para críticas e sugestões, o gerente de desenvolvimento destaca que a ferramenta é uma ideia muito boa e promissora.

A partir de uma análise mais detalhada das respostas da utilidade da ferramenta a partir da gerência de desenvolvimento, é possível inferir algumas ideias:

- A ferramenta tem utilidade e é prática para equipes ágeis, assim como apoia o desenvolvimento do método Kanban, como demonstra a resposta das três primeiras perguntas.
- A ferramenta não tem como um dos objetivos medir o desempenho de equipes, por se tratar de algo que possui muitos fatores envolvidos além de coleta de dados, como pode-se observar na resposta da pergunta 5.
- Através da ferramenta, é possível identificar gargalos e melhorias no fluxo de trabalho de equipes ágeis, conforme resposta da pergunta 6.

7 Conclusão

Neste trabalho são apresentadas a análise e implementação de uma ferramenta que possibilita a coleta de métricas ágeis por meio da integração com a plataforma GitHub, com enfoque nos indicadores necessários dentro do processo de software de equipes em ambientes de desenvolvimento ágil.

Primeiramente realizou-se um estudo da literatura na área de avaliação de processos de software. Esse estudo envolveu os conceitos de métricas ágeis em ambientes de desenvolvimento ágil e também os assuntos pertinentes ao desenvolvimento do trabalho relacionados à plataforma GitHub.

Nesse contexto de avaliação automatizada de processos em equipes ágeis, o estado da arte elaborado contou com o estudo sobre os trabalhos que utilizam ferramentas para obtenção desses resultados. Foi possível obter informações sobre como modelos automatizados vem sendo utilizados na prática para realização de avaliações de processos software, e também de implantação de algumas ferramentas Web.

Na sequência, foi realizado o levantamento e análise dos requisitos, por meio de entrevistas com equipes ágeis e com colaboradores responsáveis pela implantação do método Kanban e da *Objective and Key Results* (OKR) na organização. Essas entrevistas tiveram como objetivo a compreensão de métricas ágeis necessárias na identificação de melhorias no fluxo de desenvolvimento. Como resultado dessas entrevistas, foram levantados os requisitos funcionais.

A partir dos requisitos funcionais, foram criados os protótipos de tela da ferramenta. Nessa etapa também foi concebida a modelagem do sistema, incluindo as tecnologias a serem utilizadas, linguagem de programação e a arquitetura das camadas.

Após definida a modelagem, o sistema foi implementado, coletando dados de equipes ágeis de maneira automatizada com integração à plataforma GitHub. A partir de uma versão estável, equipes ágeis e a gerência de desenvolvimento utilizaram e avaliaram a ferramenta.

Por fim, com os resultados da avaliação da utilidade e funcionalidade da ferramenta, conclui-se que os membros de equipes ágeis e da gerência mostram uma visão positiva sobre os objetivos da ferramenta. Percebe-se pelas respostas e comentários coletados, que os usuários acreditam que a ferramenta acrescenta na identificação de melhorias no fluxo da equipe, assim como possibilita uma praticidade ao obter métricas ágeis de forma automática.

Desta forma, entende-se que o objetivo geral deste trabalho de realizar a análise e implementação de métricas ágeis em equipes através de dados coletados de forma automatizada foi atingido. Além disso, a ferramenta possibilita uma visão dos indicadores

utilizados de acordo com a necessidade dos usuários dentro de ambientes de desenvolvimento ágil.

7.1 Trabalhos futuros

Os trabalhos futuros para a ferramenta de coleta de métricas ágeis com integração à plataforma GitHub foram identificados com base no desenvolvimento e resultado da avaliação da ferramenta.

- Configuração do estado inicial de início da *sprint* de cada equipe ágil. Ou seja, existência de um módulo em que o líder da equipe possa definir a partir de qual coluna uma tarefa começa a entrar em *work in progress*.
- Implementação de um módulo de autenticação a partir da conta de usuário do Google, sendo possível realizar a verificação com o e-mail da organização.
- Realizar ajustes para que o sistema funcione não apenas a nível de repositório, mas sim a nível de organização na plataforma GitHub.
- Permitir a generalização da ferramenta rodar em qualquer repositório do GitHub, desde que esse tenha *board* com informações como *issues*.

8. Referências

ALBINO, Raphael. **Métricas Ágeis - Obtenha melhores resultados em sua equipe**. Impresso e PDF: 978-85-5519-276-0. Ano publicação: 2017.

ANACLETO, Alessandra; WANGENHEIM, C. G.; SALVIANO, C. Um método de avaliação de processos de software em micro e pequenas empresas. **SBQS-Simpósio Brasileiro de Qualidade de Software. Porto Alegre**, 2005.

ANDERSON, David J.; CARMICHAEL, Andy. **Essential Kanban condensed**. Blue Hole Press, 2016. Acesso em: agosto de 2019.

BATTISTI, Iara Denise Endruweit; BATTISTI, Gerson. Métodos estatísticos. 2008. Acesso em: Outubro de 2019.

BOEG, Jesper. **Kanban em 10 passos**. Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media, 2010.

Bold Design System. Disponível em: https://bold.bridge.ufsc.br/ Acesso em: agosto de 2019.

BR, MPS. **MPS. BR–Melhoria de Processo do Software Brasileiro**. 2011.

Castro, Felipe (2015). **Agile Goal Setting with OKR - Objectives and Key Results**, InfoQ, Disponível em: https://www.infoq.com/articles/agile-goals-okr, 2015. [Acesso em: 12 de novembro de 2018]

COHN, M. **Agile Estimating and Planning**. Massachusetts: Pearson Education, 2006.

Deborah Hartmann and Robin Dymond. **Appropriate agile measurements: Using metrics and diagnostics to deliver business value**. In Agile 2006 Conference, pages 126–131, 2006

GitHub Articles (About Repository). Disponível em: https://help.github.com/articles/about-repositories/ [Acesso em: 11 de novembro de 2018]

GitHub documentation. Disponível em: https://guides.github.com/ [Acesso em: 3 de novembro de 2018]

GitHub Glossário. Disponível em: https://help.github.com/articles/github-glossary/ [Acesso em: 11 de novembro de 2018]

GitHub Issues. Disponível em: https://guides.github.com/features/issues/, atualizado em 2014 [Acesso em: 11 de novembro de 2018]

IEEE standard glossary of software engineering terminology, 1990. IEEE Std 610.12.

Inthurn, Cândida. **Qualidade & Teste de Software**. 2. ed. Florianópolis: Visual Books Editora, 2001. 108 p.

ISO/IEC/IEEE 15939:2017(E) **International Standard - Systems and software engineering Measurement process**.

ISO, I. S. O. iec/ieee international standard-systems and software engineering–vocabulary. ISO/IEC/IEEE 24765: 2017 (E), 2017.

ISO/IEC, 2008] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION. ISO/IEC 12207:2008 **Systems and software engineering — Software life cycle processes**, Geneve: ISO, 2008.

Laboratório Bridge Portfólio. Disponível em: https://bridge.ufsc.br/portfolio Acesso em: 20 de novembro de 2018.

LARMAN, C., **Agile and Iterative Development: A Manager's Guide**, Addison-Wesley Professional, 2003.

LEAL, Stéphanie. **Um deployment package de implementação dos processos do perfil básico da norma ISO/IEC 29110.** Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2016.

LUNA, Alexandre JH de O. et al. Uma Abordagem para o Gerenciamento Estratégico Ágil em Saúde utilizando PES, OKR e MAnGve. **Revista Eletrônica da Estácio Recife**, v. 3, n. 2, 2017.

Manifesto Ágil. Disponível em http://www.manifestoagil.com.br/ Acesso em: 30 de maio de 2018.

MONTONI, Mariano et al. **Uma abordagem para Condução de Iniciativas de Melhoria de Processos de Software**. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2008/018.pdf Acesso em: 15 de abril de 2018.

NIVEN, Paul R.; LAMORTE, Ben. **Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs**. John Wiley & Sons, 2016.

PEINADO, JURANDIR; AGUIAR, G. **Compreendendo o Kanban: um ensino interativo ilustrado**. Revista DaVinci. Curitiba–PR, v. 4, n. 1, p. 133-146, 2007.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software** - 8ª Edição. McGraw Hill Brasil, 2016.

RAO, Kuda Nageswara; NAIDU, G. Kavita; CHAKKA, Praneeth. A study of the agile software development methods, applicability and implications in industry. **International Journal of Software Engineering and its applications**, v. 5, n. 2, p. 35-45, 2011.

RISING, Linda; JANOFF, Norman S. The Scrum software development process for small teams. **IEEE software**, v. 17, n. 4, p. 26-32, 2000.

ROCHA, Lais MA; SILVA, Thiago Henrique P.; MORO, Mirella M. **Análise da Contribuição para Código entre Repositórios do GitHub**. 2016. Disponível em: http://homepages.dcc.ufmg.br/~mirella/pdf/2016.SBBDshort.Rocha.pdf Acesso em: 30 de maio de 2018.

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, v. 139, 2007.

SAURO, Jeff. Using Surveys to Measure the User Experience. 2015. Disponível em: https://measuringu.com/survey-ux/. Acesso em: Outubro de 2019.

SCAFF, Mariana. **Um Estudo de Caso de Melhoria de Processos do Nível G para o Nível F do Modelo MPS Para Software**. Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2016.

SCHWABER, Ken; BEEDLE, Mike. **Agile Software Development with SCRUM**. Prentice Hall, 2002.

SOMMERVILLE, Ian; ARAKAKI, Reginaldo; MELNIKOFF, Selma Shin Shimizu. **Engenharia de software** - 9ª Edição - Pearson Prentice Hall, 2011.

THUNG, Ferdian et al. Network structure of social coding in GitHub. In: **Software maintenance and reengineering (csmr), 2013 17th european conference on**. IEEE, 2013. p. 323-326.

WAZLAWICK, Raul. **Engenharia de software: conceitos e práticas**. Elsevier Brasil, 2013.

Zwirtes, Augusto. **UM MODELO DE AVALIAÇÃO SEMI-AUTOMATIZADA DE PROCESSOS DE SOFTWARE ALINHADO AO BPMN.** Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis, 2018.