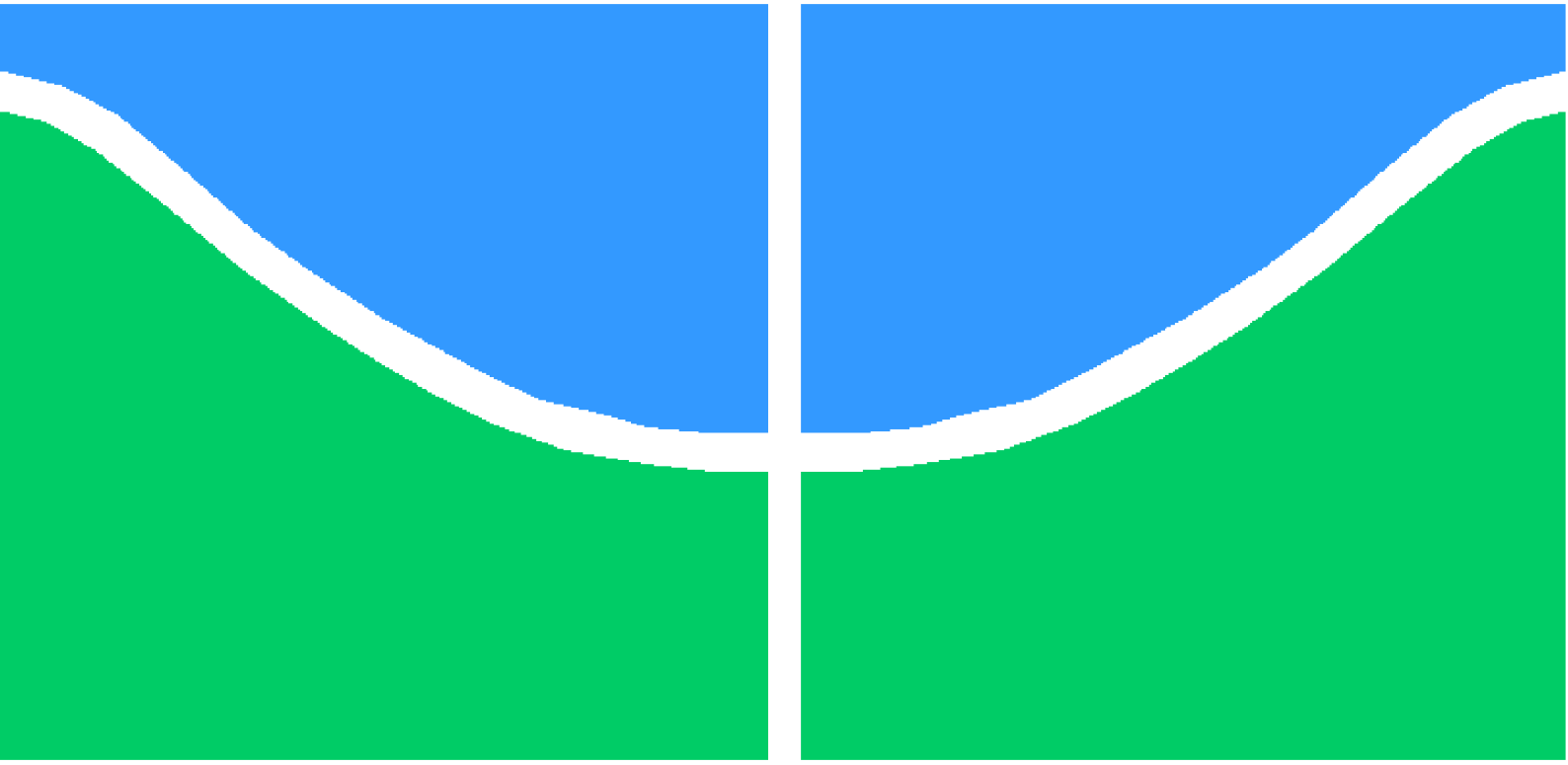

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Construção de Uma Ferramenta Flexível de Base Histórica de Medição

Autor: Karine Santos Valença
Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF
2018

Karine Santos Valença

Construção de Uma Ferramenta Flexível de Base Histórica de Medição

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF

2018

Karine Santos Valença
Construção de Uma Ferramenta Flexível de Base Histórica de Medição/ Karine Santos Valença. – Brasília, DF, 2018-
85 p. : il. (algumas color.) ; 30 cm.

Orientador: Professora Dra. Edna Dias Canedo

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.
Construção de Uma Ferramenta Flexível de Base Histórica de Medição

CDU 02:141:005.6

Karine Santos Valença

Construção de Uma Ferramenta Flexível de Base Histórica de Medição

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 13 de Julho de 2018:

Professora Dra. Edna Dias Canedo
Orientadora

Professora Msc. Cristiane Soares Ramos
Convidado 1

Professor Msc. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2018

Dedico este trabalho à minha mãe, que nunca mediu esforços para me ajudar nesta jornada.

Agradecimentos

Primeiramente, quero agradecer à minha família, que sempre apoiaram meus estudos e tiveram paciência comigo durante os momentos mais difíceis. Em especial, à minha mãe, que durante toda a graduação, se prontificou para fazer minhas marmitas, sempre me buscou e me levou à parada de ônibus, e sempre esteve atenta à minha saúde.

Ao meu namorado, Filippe, que me abraçou e me consolou quando eu quis desistir e me deu forças para continuar.

Aos professores da FGA, que compartilharam seu conhecimento, mesmo diante das situações problemáticas do campus, e me incentivaram a buscar conhecimento cada vez mais. Em especial, à professora Edna Dias Canedo, que me iniciou no mundo científico e da pesquisa.

Aos meus grandes amigos, Felipe, Lucas, Rebecca, Pedro e Victor, que me deram conselhos e me divertiram mesmo quando eu estava mal-humorada.

Karine Santos Valença

"Mas, hey mãe!

Alguma coisa ficou pra trás

Antigamente eu sabia exatamente o que fazer"

(Engenheiros do Hawaii)

Resumo

A medição é uma área importante na engenharia de software, uma vez que ela permite que as organizações consigam fazer estimativas confiáveis sobre prazo, custo e qualidade. Desenvolver um processo de medição bem definido, baseado nos objetivos de negócio, fornece uma fonte de informação confiável para tomada de decisões nos projetos. Problemas relacionados à criação e à manutenção de uma base histórica de medição são recorrentes nas organizações que trabalham com desenvolvimento de software, uma vez que não existem muitas ferramentas que mantém a base histórica das métricas e que existe falta de flexibilidade na criação e edição das métricas na maioria das ferramentas medição disponíveis. Diante deste problema, este trabalho tem como objetivo desenvolver uma ferramenta de base história de medição flexível. A metodologia de pesquisa utilizada foi a revisão sistemática de literatura. Além disso, a metodologia de desenvolvimento foi uma versão simplificada do *Scrum*. Os resultados da revisão sistemática apresentam as ferramentas de métricas listadas na literatura, quais são as suas funcionalidades, suas vantagens e também desvantagens. Foi obtido também, as principais métricas utilizadas por essas ferramentas. A ferramenta desenvolvida trabalhou boas práticas de programação, buscou utilizar das forças que as ferramentas estudadas mostraram e preencheu as lacunas que essas ferramentas apresentam. Além disso, a ferramenta foi validada por iterações de Interação Humano Computador e em ambiente real de desenvolvimento de software. Os resultados dessa validação foram aplicados em melhorias para o sistema. Assim, o objetivo do estudo foi atingido, uma vez que a ferramenta de base histórica de medições foi construída e validada utilizando boas técnicas de desenvolvimento de software.

Palavras-chaves: Medição, Métricas de Software, Base Histórica de Medição.

Abstract

Measurement is an important area in software engineering once it enables organizations to do reliable estimates of time, cost, and quality. Developing a well-defined measurement process, based on business objectives, provides a reliable source of information for project decision-making. Problems related to the creation and maintenance of a historical measurement base are recurrent in organizations that work with software development since there are few tools that maintain the historical base of the metrics and that there is a lack of flexibility in the creation and edition of the metrics in the majority measuring tools available. Faced with this problem, this work aims to develop a flexible measurement historical base tool. The research methodology used was the systematic review of the literature. In addition, the development methodology was a simplified version of Scrum. The results of the systematic review present metrics tools listed in the literature, what are their functionalities, their advantages and also disadvantages. The main metrics used by these tools were also obtained. The tool developed used good programming practices, sought to use the forces that the tools studied showed and filled the gaps that these tools present. In addition, the tool was validated by Human Computer Interaction iterations and in real software development environment. The results of this validation were applied in improvements to the system. Thus, the objective of the study was reached, since the measurement historical base tool was constructed and validated using good software development techniques.

Key-words: Measurement, Software metrics, Measurement Historical Base.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CBO	*Coupling Between Objects*
CLOC	*Comment lines of Program Text*
DIT	*Depth of Inheritance Tree*
IHC	*Interação Humano Computador*
LCOM	*Lack of COhesion of Methods*
LOC	*Lines of Code*
MVC	*Model-View-Controller*
NLOC	*Non-commented Line of Code*
NOC	*Number of Children*
OCL	*Object Constraint Language*
QP	Questão de Pesquisa
RFC	*Response for Class*
RSL	Revisão Sistemática de Literatura
TCC	Trabalho de Conclusão de Curso
UML	*Unified Modeling Language*
WMC	*Weighted Methods per Class*

Sumário

1 Introdução

1.1 Contexto

A habilidade de fazer estimativas e predições permite às organizações que desenvolvem software obterem maior sucesso, visto que elas conseguirão realizar planos de projeto alcançáveis tanto em termos de tempo, quanto de qualidade. A medição, quando realizada de maneira efetiva, fornece conhecimento suficiente para as organizações de software realizarem as estimativas de maneira confiável e a detectar problemas antecipadamente, permitindo, assim, controle de custos, redução dos riscos e aumento da qualidade (FLORAC; PARK; CARLETON, 1997).

Segundo (FENTON; PFLEEGER, 1997), medição é "o processo pelo qual números e símbolos são atribuídos a atributos de entidades no mundo real de modo a descrevê-los de acordo com regras claramente definidas". Desta forma, a medição na engenharia de software visa caracterizar os elementos intrínsecos ao desenvolvimento de software de modo que esses elementos sejam mensuráveis.

O processo de medição visa realizar a coleta e análise de dados, e reportar esses dados de forma que possa apoiar a gestão dos processos de uma organização (ISO/IEC/IEEE, 2008). Assim, ao realizar processos de medição bem sucedidos, a organização será capaz de fazer predições sobre prazos, custos e recursos necessários para entregar seus produtos.

As organizações não devem realizar as medições apenas por fazer medições, uma vez que é uma atividade que gera custos para a empresa. Para que a medição seja viável para a organização, ela deve ser planejada de acordo com os objetivos de negócio da organização, fornecendo informações que serão utilizadas para as tomadas de decisão (FLORAC; PARK; CARLETON, 1997).

1.2 Justificativa

As organizações utilizam diferentes métricas para avaliar seus produtos e processos. Essas métricas podem ser de tamanho, de produtividade, de qualidade, de complexidade, e várias outras (FENTON; PFLEEGER, 1997) (KAN, 2002). Assim, para fazer essa avaliação, é necessário que essas métricas estejam registradas historicamente em um local confiável, e preferencialmente, de fácil acesso. Além disso, pode ser necessário, também, o uso de métricas mais complexas ou até mesmo métricas customizadas específicas para a necessidade da organização.

Por exemplo, a informação sobre a quantidade de problemas por usuários por mês

pode ser útil. Essa métrica é complexa, pois necessita das informações sobre os problemas que os clientes relataram por um período de tempo e o total de licenças durante esse período, sendo que o total de licenças por período leva utiliza as informações sobre número total de licenças instaladas do software e o número de meses no período de cálculo (KAN, 2002).

Diante da necessidade do uso dessas métricas mais complexas, ou até mesmo métricas customizadas, é interessante que a base histórica de medição seja flexível, a ponto de permitir que o próprio usuário defina suas métricas. Além disso, uma base flexível vai permitir mudanças nas métricas já registradas. Por exemplo, a métrica anteriormente apresentada, quantidade de problemas por usuários por mês, poderia ser alterada para quantidade de problemas por usuários ano, caso a organização sentisse necessidade. Com uma base flexível de medições, seria necessário apenas a alteração de uma das medidas envolvidas na métrica, permitindo maior manutenção da base histórica.

É interessante também que o registro, o cálculo e a apresentação dessas métricas seja feita de forma automatizada, uma vez que a automatização permitirá maior rapidez nas atividades de coleta e análise dos dados (FENTON; PFLEEGER, 1997).

1.3 Problema

A inflexibilidade das ferramentas de medições impede que as organizações coletem os dados necessários de maneira adequada. Durante o processo de planejamento de medições, na etapa de "Selecionar e definir métricas"(FLORAC; PARK; CARLETON, 1997), podem ser identificadas métricas necessárias para atender os objetivos de negócio da organização. Porém, a ferramenta utilizada pela a organização pode não abranger a métrica escolhida e além disso, ser inflexível, não permitindo criação de novas métrica.

Desta forma, a coleta da métrica seria inviabilizada, perdendo informações importantes para avaliação do processo, ou então haveria uma mudança na ferramenta, perdendo tempo de migração e aprendizado, e até mesmo, perda dos dados históricos.

Além disso, foi observado que boa parte das ferramentas de medição não mantém a base histórica dos seus dados. Assim, elas apenas realizam medição, porém cabe ao usuário manter esse dado em algum banco de dados de métricas externo à ferramenta.

Este trabalho apresenta uma solução de base histórica de medição flexível. Sendo assim, a ferramenta criada deverá permitir a criação, edição e manutenção das métricas em base histórica. A ferramenta também gerará gráficos que auxiliarão na visualização e compreensão dos dados históricos. Além disso, a ferramenta também permitirá a importação de dados, evitando retrabalho caso o usuário esteja migrando para a ferramenta e automatizando parte do processo de criação de medidas. A ferramenta não busca realizar

o processo de medição, por meio de análises de código, por exemplo, mas sim fornecer um local estável para o armazenamento e análise desses dados historicamente.

1.4 Questões de pesquisa

Visando entender o estado atual das ferramentas de medição, e assim conseguir propor uma ferramenta que visa resolver o problema na flexibilidade das métricas e na manutenção da base histórica, foram levantadas quatro questões de pesquisa (QP) para serem respondidas. Essas questões foram:

QP1. Quais são as ferramentas de medições encontradas na literatura?

QP2. Quais as funcionalidades as ferramentas de medições apresentam?

QP3. Quais as vantagens e desvantagens das ferramentas de medições?

QP4. Quais as métricas identificadas e utilizadas pelas ferramentas de medições?

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral desse trabalho é desenvolver uma ferramenta de base histórica de medição flexível. Essa ferramenta deve permitir às organizações que desenvolvem software definirem suas próprias métricas e manter sua base histórica de métricas.

1.5.2 Objetivos Específicos

Visando atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

- Fazer uma revisão de literatura relacionada a área de medição de software;
- Fazer uma revisão de literatura relacionada sobre o estado atual das ferramentas de métricas e medições;
- Implementar uma ferramenta de base histórica de medição;
- Validar a ferramenta proposta.

1.6 Metodologia de Pesquisa

A metodologia de pesquisa utilizada neste trabalho foi a Revisão Sistemática de Literatura (RSL). Essa metodologia visa identificar e analisar as pesquisas relevantes para

uma questão de pesquisa, por meio da definição de um protocolo de revisão, estratégia de busca e critérios de inclusão e exclusão (KITCHENHAM; CHARTERS, 2007).

1.7 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta o referencial teórico levantado sobre medição. O Capítulo 3 apresenta a metodologia de pesquisa que foi utilizada para responder às perguntas do trabalho. O Capítulo 4 apresenta os resultados obtidos na revisão sistemática de literatura. O Capítulo 5 apresenta o desenvolvimento da ferramenta, linguagem de programação e arquitetura do sistema, as funcionalidades e a validação da ferramenta. E finalmente, o capítulo 6 apresenta as considerações finais, aprendizados e trabalhos futuros.

2 Referencial Teórico

Neste Capítulo, será apresentada a base teórica que será necessária para a compreensão do trabalho. A Seção 2.1 apresenta os principais conceitos sobre medição e definições importantes. A Seção 2.2 mostra as abordagens que podem ser utilizadas para realizar a medição de software.

2.1 Medição de software

Existem várias definições para o que é medição. Uma das definições clássicas é apresentada por (FINKELSTEIN; LEANING, 1984) e diz que "Medição é a atribuição de números a propriedades de objetos ou eventos no mundo real por meio de uma operação empírica objetiva de modo a descrevê-los". Outra definição clássica para medição é apresentada por (FENTON; PFLEEGER, 1997), que classifica a medição como "O processo pelo o qual números e símbolos são atribuídos a atributos de entidades no mundo real de modo a descrevê-las de acordo com regras claramente definidas". Com base nesses conceitos básicos, podemos dizer que a medição visa caracterizar numericamente os atributos dos objetos e dos eventos do mundo real.

Dentro da engenharia de software, podemos dizer que a medição de software é um processo contínuo, onde os dados dos processo do desenvolvimento de software são definidos, coletados e analisados, com o objetivo de fornecer informação para melhoria dos processos e produtos de software (SOLINGEN; BERGHOUT, 1999).

O processo de medição, segundo a definição da (ISO/IEC/IEEE, 2008) visa "coletar, analisar e reportar dados objetivos e informações para apoiar a gestão efetiva e demonstrar a qualidade dos produtos, serviços e processos". Este processo, apresenta vários termos específicos. Na seções a seguir, são apresentados os principais termos utilizados nesta área.

2.1.1 Métricas

A medição no processo de software é realizado utilizando métricas de software (SOLINGEN; BERGHOUT, 1999). Matematicamente, uma métrica é a regra utilizada para descrever a distância entre dois pontos (FENTON; PFLEEGER, 1997). Sendo assim, uma métrica é composta por várias medidas, uma vez que a medida representa um desses pontos e métrica contém todos os pontos, permitindo a análise desses valores em conjunto.

As métricas de software possuem diversas classificações. Elas podem ser classificadas em métricas produto ou processo, como métricas diretas ou indiretas, e até mesmo

métricas objetivas ou subjetivas.

2.1.1.1 Métrica de Produto

Uma métrica de produto corresponde ao conjunto de medidas que medem um produto resultante do processo de desenvolvimento de software (SOLINGEN; BERGHOUT, 1999).

Um exemplo de métrica de produto é a complexidade de um código.

2.1.1.2 Métrica de Processo

Uma métrica de processo corresponde ao conjunto de medidas que medem as características do processo de desenvolvimento de software (SOLINGEN; BERGHOUT, 1999).

Como exemplo de métrica de processo, pode-se citar a produtividade do time.

2.1.1.3 Métrica direta

Uma métrica direta de um atributo envolve nenhum outro atributo ou entidade (FENTON; PFLEEGER, 1997) (FENTON, 1994).

Um exemplo de métrica direta é quantidade de linhas de código, uma vez que esta métrica não depende de nenhum atributo para responder à informação requisitada (FENTON; PFLEEGER, 1997).

2.1.1.4 Métrica indireta

Uma métrica indireta de um atributo envolve a medição de um ou mais atributos (FENTON, 1994).

Um exemplo de métrica indireta é densidade de defeitos do módulo, pois esta métrica necessita do número de defeitos e do tamanho do módulo sendo medido (FENTON; PFLEEGER, 1997).

2.1.1.5 Métrica Objetiva

Uma métrica objetiva é feita pela contagem de características de um modo objetivo (SOLINGEN; BERGHOUT, 1999).

Como exemplo de métrica objetiva, pode-se citar número de falhas encontradas.

2.1.1.6 Métrica Subjetiva

Uma métrica subjetiva é uma métrica que envolve o julgamento humano, dessa forma, sendo subjetiva (SOLINGEN; BERGHOUT, 1999).

Como exemplo de métrica subjetiva, pode-se citar conformidade com padrões de projeto.

2.1.2 Medida

Na Seção 2.1.1 foi citado que uma métrica é composta por várias medidas. Uma medida corresponde ao número atribuído a uma entidade, como resultado de uma medição, visando caracterizar um atributo desta entidade (FENTON; PFLEEGER, 1997) (ISO/IEC/IEEE, 2017).

2.1.3 Indicador

Um indicador corresponde a uma medida que fornece uma estimativa ou uma avaliação (ISO/IEC/IEEE, 2017).

A equipe pode definir indicadores para avaliar uma métrica, por exemplo, se uma métrica de qualidade de código, como complexidade ciclomática, está acima do que foi determinado pelo indicador, isso é um sinal de que o código necessita de refatoração (SHIN; MENEELY; WILLIAMS; OSBORNE, 2011). Ou seja, os indicatores são utilizados como objeto para tomada de decisões.

2.1.4 Entidade

Corresponde ao objeto ou evento do mundo real que será caracterizado pela medição dos seus atributos (ISO/IEC/IEEE, 2017). Em engenharia de software, existem 3 classes de interesse: Processo, Produto e Recursos.

As entidades de processo são as atividades relacionadas ao software, geralmente associadas com uma escala de tempo. As entidades de produto são entregáveis que surgem dos processos. As entidades de recursos são os itens de entrada dos processos (FENTON; PFLEEGER, 1997).

2.1.5 Atributos

Um atributo corresponde a uma característica de uma entidade que pode ser diferenciada quantitativamente ou qualitativamente (ISO/IEC/IEEE, 2017). Os atributos de uma entidade podem ser classificados em 2 tipos: atributos internos e atributos externos.

2.1.5.1 Atributos Internos

Os atributos internos de uma entidade são aqueles que podem ser medidos ao examinar a entidade em si, separada do seu comportamento, por exemplo, a quantidade de linhas de código de uma classe (FENTON; PFLEEGER, 1997).

2.1.5.2 Atributos Externos

Os atributos externos medem como a entidade se relaciona com outras entidades, focando a medição no comportamento daquela entidade, como produtividade, por exemplo (FENTON; PFLEEGER, 1997).

2.1.6 Escala de medição

Uma escala corresponde a um conjunto ordenado de valores ou de categorias, que realiza um mapeamento para um atributo (ISO/IEC/IEEE, 2017). Uma métrica possui uma escala de medição correspondente, assim suas medidas são mensuradas nessa escala de medição. Uma escala pode ser classificada em 5 tipos principais (FENTON; PFLEEGER, 1997) (FENTON, 1994):

- Nominal: Neste tipo de escala não existe noção de ordenamento entre as classes que classificam o atributo. Além disso, não existe noção de magnitude. Os valores de medição são basicamente categóricos.
- Ordinal: Neste tipo de escala existe a noção de ordenamento, dessa forma os valores de medição significam posição. Operações matemáticas nesse tipo de escala não fazem sentido.
- Intervalar: Neste tipo de escala existe a noção de ordenamento. Os valores de medição mantém distâncias iguais, mas não proporções. Operações matemáticas do tipo soma e subtração são aceitas, mas multiplicação e divisão não são.
- Ratio: Neste tipo de escala existe a noção de ordenamento e os intervalos entre os valores, além disso a proporção também é mantida. Existe o valor zero, que corresponde a nenhum atributo. Todas as operações aritméticas são permitidas.
- Absoluta: Neste tipo de escala, a medição é feita apenas pela contagem do número de elementos da entidade. Nenhum tipo de operação aritmética faz sentido.

2.1.7 Unidade de medida

Unidade de medida corresponde à uma quantidade, definida por convenção, permitindo a comparação entre entidades representadas em unidades de medidas iguais (ISO/IEC/IEEE, 2017).

As medidas dentro da mesma métrica, possuem a mesma unidade de medida. Dessa forma, é possível comparar medidas dentro de uma mesma métrica. Por exemplo, a porcentagem de cobertura de testes no código, ao longo das *sprints*.

2.1.8 Função de medição

A função de medição corresponde ao cálculo utilizado para combinar duas medidas e formar uma medida indireta (ISO/IEC/IEEE, 2017).

2.1.9 Operador

Um operador corresponde à uma entidade que realiza a operação em uma função de medição (ISO/IEC/IEEE, 2017).

A operação de divisão em uma medida indireta, como densidade de defeitos, pode ser um exemplo de operador (FENTON; PFLEEGER, 1997). Outras operações matemáticas, como soma, subtração e multiplicação, também são operadores.

2.1.10 Base de experiência de medição

Uma base de experiência de medição corresponde a um banco de dados que armazena os resultados de medições e lições aprendidas anteriormente (ISO/IEC/IEEE, 2017). Neste trabalho, será utilizado o termo base histórica de medição, que se assemelha bastante à base de experiência de medição, porém, não registra lições aprendidas, apenas os dados histórico das métricas.

2.2 Abordagens de Medição de Software

Existem diferentes abordagens que podem ser utilizadas para realizar a medição dentro dos processos de Engenharia de Software. Essa abordagens fornecem metodologias que facilitam no processo de medição de software.

Neste trabalho, serão estudadas 3 abordagens para medição. Na Seção 2.2.1 é apresentada a abordagem de medição Goal-Question-Metric (GQM). Na Seção 2.2.2 é apresentada a abordagem de medição sugerida pelo Modelo Integrado de Maturidade em Capacitação (CMMI). Finalmente, na Seção 2.2.3 é apresentada a abordagem de medição apresentada pelo Pratical Software Measurement (PSM).

2.2.1 Goal-Question-Metric

O GQM é uma abordagem de medição de software orientada a objetivos, desenvolvida por Basili e Weiss. Nessa abordagem, o objetivo definido gera questões para o projeto, e as questões, por sua vez, geram várias métricas que visam responder essas questões (SOLINGEN; BERGHOUT, 1999).

Essa abordagem é constituída de 4 fases principais (SOLINGEN; BERGHOUT, 1999):

1. Fase de Planejamento: Nesta fase, o projeto de medição é selecionado. Um plano de projeto deve ser construído, onde o projeto deve ser definido, caracterizado e planejado.

2. Fase de Definição: Nesta fase, os objetivos, questões e métricas são definidos e documentados.

3. Fase de Coleta: Nesta fase, os dados reais do projeto são coletados.

4. Fase de Interpretação: Os dados que foram coletados são analisados em relação às métricas definidas, respondendo, assim às questões e fazendo a avaliação se o objetivo da medição foi atingido.

Nesta abordagem, a fase de definição é feita utilizando uma perspectiva *top-down*, ou seja, começa com uma abstração maior e vai diminuindo ao longo dos níveis. Na fase de interpretação, a perspectiva é *bottom-up*, ou seja, o entendimento sobre o contexto vai crescendo ao longo dos níveis (SOLINGEN; BERGHOUT, 1999)

2.2.2 Modelo Integrado de Maturidade em Capacitação

O CMMI é um guia que contém elementos essenciais para melhorar os processos de software (CHRISSIS; KONRAD; SHRUM, 2011). Dentro do CMMI, existem várias áreas para apoiar a gestão dos processos, e uma delas é a área de Medição e Análise.

A Medição e Análise é uma área de suporte no nível de maturidade 2, que tem como proposta desenvolver a capacidade de medição, apoiando, assim, a necessidade de informação da gestão (CHRISSIS; KONRAD; SHRUM, 2011). Contém as seguintes atividades:

1. Definir os objetivos da medição e análise, de forma que eles estejam alinhados com as necessidades de informação e objetivos do projeto e do negócio.

2. Especificar as métricas, técnicas de análise e métodos para a coleta e armazenamento de dados, além de será feito o reporte das atividades e os *feedbacks*.

3. Implementar as técnicas e métodos para a coleta, reporte e *feedback* dos dados.

4. Fornecer os resultados de maneira objetiva, a fim de que eles possam ser utilizados para as tomadas de decisões e ações corretivas.

Ao implementar o processo de Medição e Análise proposto pelo CMMI, os seguintes resultados são esperados (CHRISSIS; KONRAD; SHRUM, 2011):

- Planejamento e estimativas objetivas;

- Acompanhamento do progresso real e comparação com os planejamento estabelecido;
- Identificação e resolução de problemas relacionados ao processo;
- Criação de uma base sólida para a incorporação da medição e análise em novos processos.

2.2.3 Pratical Software Measurement

O PSM corresponde à um guia sobre como utilizar medições para gerenciar e melhorar os processos de software (FLORAC; PARK; CARLETON, 1997). Dentro do PSM, é sugerido um modelo básico para gerenciamento e melhoria dos processos software. As seguintes atividades são propostas (FLORAC; PARK; CARLETON, 1997):

- Definir o processo: Identificação dos elementos que contribuem para o processo e quais os objetivos desse processo. Alcance do entendimento do processo e seus objetivos por parte dos envolvidos no processo.
- Planejar medidas: Problemas no processo são identificados. As medidas de software devem ser selecionadas. Métodos para coleta e análise das medidas devem ser fornecidos.
- Executar o processo: Execução do processo de software. As medidas devem ser coletadas durante e ao final de cada processo.
- Aplicar as medidas: Armazenamento e análise das medidas que foram coletadas durante a execução do processo.
- Controlar o processo: Execução de ações caso o processo varie de modo imprevisível, visando retornar o processo para seu nível natural.
- Melhorar o processo: Aplicação de ações de melhoria, baseado nos resultados das análises das métricas que foram coletadas no processo de software.

3 Metodologia

Neste Capítulo será discutido a metodologia utilizada para o desenvolvimento deste trabalho. A Seção 3.1 apresenta metodologia de pesquisa que foi utilizada para a execução da revisão bibliográfica.

3.1 Metodologia de Pesquisa

A metodologia de pesquisa utilizada neste trabalho utilizou vários princípios da revisão sistemática de literatura (RSL). A RSL é uma forma de estudo secundário que visa identificar e analisar as pesquisas relevantes para uma questão de pesquisa. Dentre as suas características, podemos destacar como principais (KITCHENHAM; CHARTERS, 2007):

- Possui um protocolo de revisão que é definido no começo da pesquisa. Esse protocolo define as questões de pesquisa a serem abordados e quais métodos são utilizados durante a pesquisa.
- Propõe a criação de uma estratégia de busca bem documentada. Essa estratégia deve ser boa o suficiente para conseguir a maior quantidade de literatura relevante para o assunto.
- Utiliza de critérios de inclusão e exclusão para avaliar os estudos primários encontrados.

Os principais motivos para escolher uma RSL como metodologia de pesquisa são para identificar lacunas nos estudos atuais, reunir as evidências existentes sobre algum tema e/ou construir conhecimento para propor novas pesquisas (KITCHENHAM; CHARTERS, 2007). Como este trabalho tem por objetivo a criação de uma ferramenta de base histórica de medição flexível, é interessante identificar as ferramentas atuais e também construir uma base sólida de conhecimento para propor essa tecnologia. Por esses motivos, a RSL foi considerada adequada para este trabalho.

A revisão sistemática utilizada neste trabalho foi composta de 3 fases principais (KITCHENHAM; CHARTERS, 2007), são elas:

- **Planejar a revisão:** Nesta fase, foram levantadas as questões de pesquisa. Além disso, o protocolo de revisão foi desenvolvido.

- **Conduzir a revisão:** Nesta fase, o protocolo de revisão definido na etapa anterior foi aplicado rigorosamente.
- **Reportar a revisão:** Nesta fase, foram reportados os resultados da revisão sistemática.

3.1.1 Questões de pesquisa

Visando atingir o objetivo do estudo, foram definidas quatro questões de pesquisa. Estas questões visam auxiliar a ter um panorama atual das pesquisas realizadas na área de ferramentas de medições. As questões de pesquisa (QP) levantadas foram:

QP1. Quais são as ferramentas de medições encontradas na literatura?

QP2. Quais as funcionalidades as ferramentas de medições apresentam?

QP3. Quais as vantagens e desvantagens das ferramentas de medições?

QP4. Quais as métricas identificadas e utilizadas pelas ferramentas de medições?

3.1.2 Protocolo de revisão

Esta revisão sistemática seguiu rigidamente um protocolo de revisão. A Figura 1 apresenta as atividades desse protocolo de revisão.

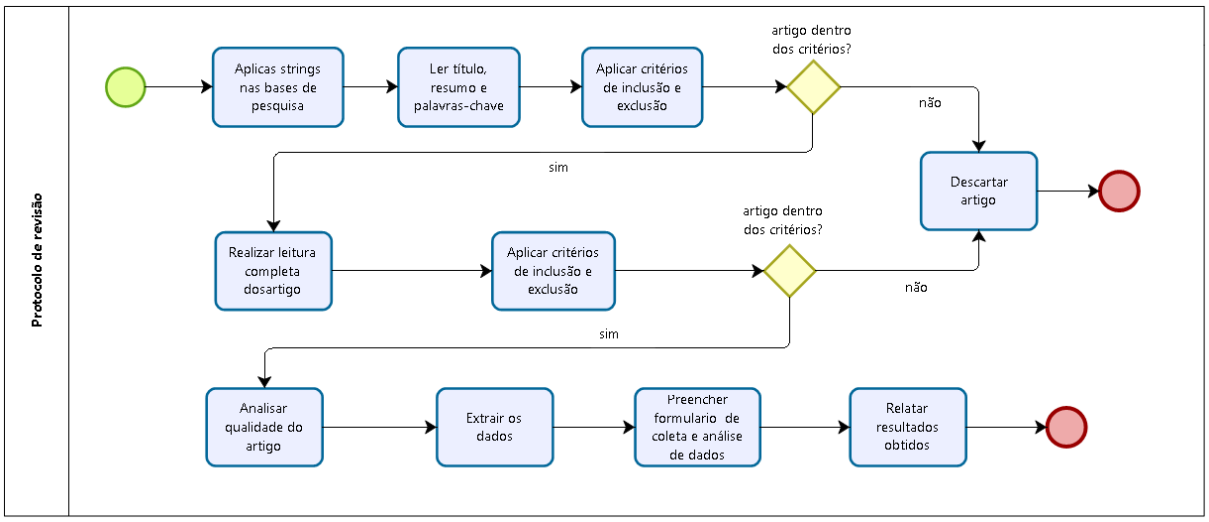

Figura 1 – Procoloco de revisão

Fonte: Autora

A atividade "Aplicar string de busca nas bases de pesquisa"consiste de aplicar a *string* de busca que foi definida nas bases de pesquisas, a Seção 3.1.3 detalha melhor essa atividade.

A atividade "Ler título, resumo e palavras-chave"consiste em realizar a leitura destas informações dos artigos, a fim de verificar se ele se enquadra dentro dos critérios de inclusão e exclusão.

A atividade "Aplicar critérios de inclusão e exclusão"consiste em observar os critérios de inclusão e exclusão e verificar se o artigo se encaixa nestes critérios, caso não, o artigo deve ser descartado. A Seção 3.1.4 detalha melhor os critérios de inclusão e exclusão.

A atividade "Realizar leitura completa do artigo"permite identificar se o artigo responde à questão de pesquisa e se está dentro dos critérios de inclusão e exclusão. Caso o artigo não responda à questão de pesquisa, ele deve ser descartado.

A atividade "Analisar qualidade do artigo"consiste em analisar o artigo segundo os critérios de qualidade definidos. A Seção 3.1.6 detalha melhor como foi feita essa análise e quais critérios foram utilizados.

A atividade "Extrair os dados"realiza a coleta dos dados que respondem à questão de pesquisa, encontrados nos artigos.

A atividade "Preencher formulário de coleta e análise de dados"visa organizar sistematicamente todos os dados obtidos durante a revisão sistemática e fazer um relato organizado dos dados. A Seção 3.1.5 apresenta o resultado desta atividade.

A atividade "Relatar resultados obtidos"consiste em responder às questões de pesquisa definidas com base nos artigos encontrados durante o procedimento de revisão. O Capítulo 4 apresenta o resultado desta atividade.

3.1.3 Estratégia de busca

Para realizar a busca na bases de pesquisa, foi construída uma *string* de busca. A *string* foi definida utilizando a abordagem PICO (*Population, Intervation, Comparation, Outcomes*), que auxilia a melhor enquadrar as questões de pesquisa. Nos itens abaixo estão explicados cada uma das letras da abordagem PICO (KITCHENHAM; CHARTERS, 2007):

- **População:** A população corresponde a um papel específico dentro do ciclo de vida de software, uma área de aplicação ou até mesmo a um grupo específico da indústria.
- **Intervenção:** A intervenção corresponde a metodologias, ferramentas, tecnologias ou procedimentos de software que abordam uma questão.
- **Comparação:** A comparação corresponde a metodologias, ferramentas, tecnologias ou procedimentos de software na qual a intervenção está sendo comparada. Nem sempre é obrigatório ter comparação.

- **Resultados:** Os resultados devem relatar os fatores de importância relevantes para os profissionais de software.

Baseado nisso, para responder às questões de pesquisa, o PICO definido foi:

- **População:** Área de medição de software.
- **Intervenção:** Ferramenta de coleta de métricas.
- **Comparação:** *Não se aplica.*
- **Resultados:** Funcionalidade, Vantagens, Desvantagens.

Após executar a abordagem PICO, obteve-se a seguinte *string* de busca:

(TITLE-ABS-KEY (("software measurement"OR "software metrics")) AND TITLE-ABS-KEY (("metric collection tool"OR "metric application"OR "metric tool"OR "metrics tool"OR "metrics collection tool"OR "software tool"OR "metrics tools"OR "measurement tools" OR "measurement tool")))

Com a *string*, buscas automatizadas foram feitas na base de pesquisa SCOPUS, visto que a *SCOPUS* indexa diferentes bases de pesquisa, incluindo a *IEEE Xplore*, a *ACM Digital Library* e *Science Direct*, além de outras bases relevantes. Além disso, também foram feitas buscas automatizadas, especificamente, na base *Science Direct*, uma vez que notou-se que a indexação da *SCOPUS* não estava retornando todos os resultados para essa base em específico.

3.1.4 Critérios de inclusão e exclusão

Com o intuito de selecionar apenas os artigos mais relevantes para o estudo, foi necessário aplicar critérios de inclusão e exclusão. Os critérios de inclusão aplicados para esse estudo são:

- Artigos que contenham título, resumo e palavras-chave relacionados com a questão de pesquisa.
- Artigos em inglês.
- Artigos publicados entre 2007 e 2017.

Os critérios de exclusão aplicados para esse estudo são:

- Artigos duplicados.
- Artigos que não contribuem para responder à questão de pesquisa.
- Artigos incompletos (publicados como *Short Paper*).

3.1.5 Coleta e análise dos dados

Após obter os artigos que atendem aos critérios de inclusão e exclusão, foi realizada a coleta e análise dos dados. A fim de auxiliar no procedimento da revisão sistemática, a ferramenta StArt[1] foi utilizada. Essa ferramenta permite os resultados da pesquisa sejam armazenados e que todo o procedimento de inclusão e exclusão de estudos seja realizado. Além disso, a ferramenta também disponibiliza gráficos que auxiliam na análise dos resultados da revisão sistemática.

3.1.5.1 Primeira etapa da coleta de dados

A primeira etapa do processo de seleção dos estudos consistiu na execução da estratégia de busca apresentada na Seção 3.1.3, ou seja, inserir a *string* nas bases de pesquisa definidas. Desta forma, 215 artigos foram encontrados. A Figura 2 mostra a quantidade de artigos encontrados em cada uma das bases de pesquisa.

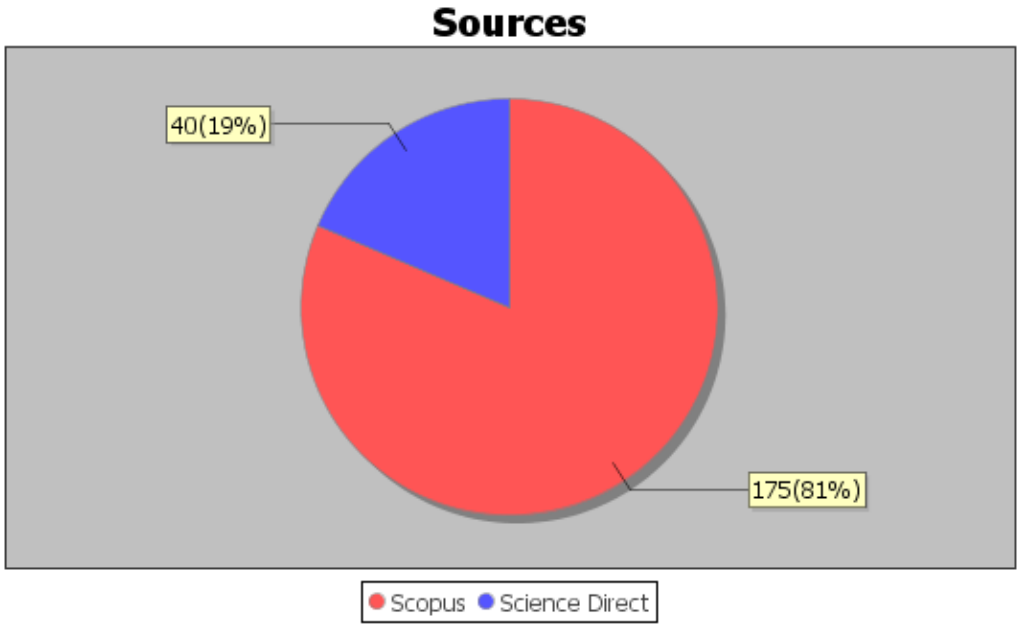

Figura 2 – Quantidade de artigos encontrados em cada uma das bases

Fonte: Ferramenta StArt

A maior parte dos artigos, 81% foram obtidos por meio da base *Scopus* e 19% dos artigos foram retirados da base *Science Direct*

3.1.5.2 Segunda etapa da coleta de dados

A segunda etapa da coleta consistiu na leitura do título, do resumo e das palavras-chave para verificar se o artigo estava dentro dos critérios de inclusão e exclusão definidos.

[1] http://lapes.dc.ufscar.br/tools/start_tool

Conforme apresentado na Figura 3, dos 215 artigos encontrados na etapa 1, 78% desses artigos foram rejeitados, 19% foram aceitos e 3% foi duplicado.

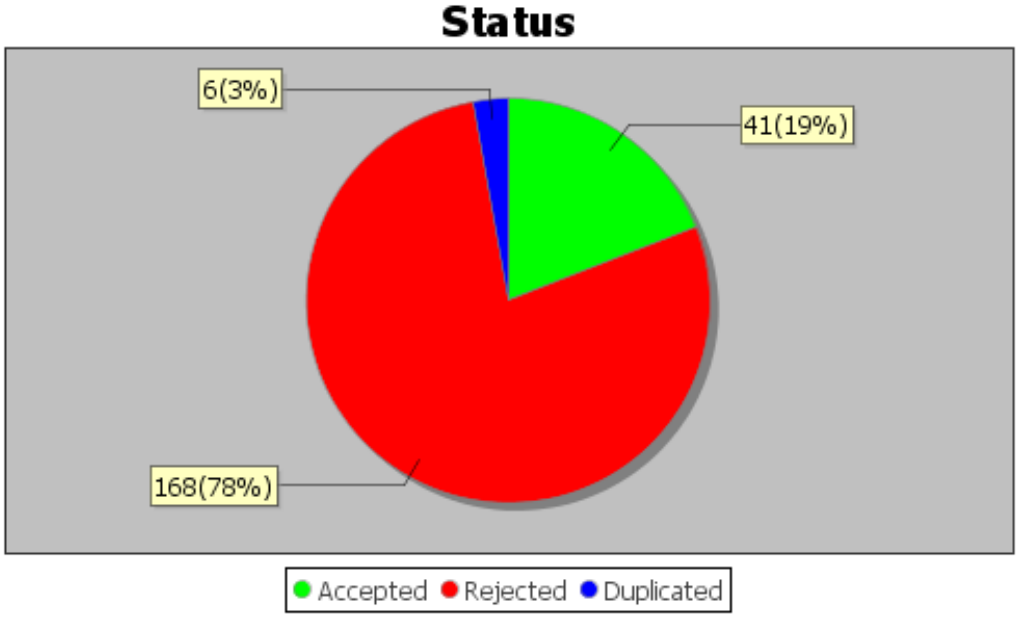

Figura 3 – *Status* dos artigos após critérios de inclusão

Fonte: Ferramenta StArt

A grande taxa de rejeição deu-se, principalmente, pelo fato dos artigos não atenderem ao critério de inclusão "Artigos publicados entre 2007 e 2017".

3.1.5.3 Terceira etapa da coleta de dados

Na terceira etapa da coleta de dados, foi feita a leitura completa dos artigos aceitos na segunda etapa. Dessa forma, foi verificado se o artigo contribuiria para responder à questão de pesquisa. Além disso, os outros critérios de exclusão também foram aplicados.

A fim de otimizar a atividade de leitura dos artigos, eles foram classificados segundo um critério de prioridade de leitura. Este critério de prioridade foi definido após a leitura do título e do resumo dos artigos. Artigos que possuíam em seu título ou resumo as palavras "ferramentas"ou "aplicação", e "métricas"ou "medição", e listavam que iriam apresentar uma ferramenta, foram considerados como altíssima prioridade de leitura. Os artigos que incluíam estes termos, porém não listavam que uma ferramenta de métrica seria apresentada, foram considerados com alta prioridade de leitura. Artigos que incluíam pelo menos um destes termos e que o resumo apresentava resultados promissores, foram enquadrados como baixa prioridade de leitura. Finalmente, artigos que incluíam pelo menos um destes termos, porém em seu resumo não apresentava resultados promissores, foram considerados baixíssima prioridade de leitura.

A Figura 4 apresenta a prioridade de leitura para os artigos.

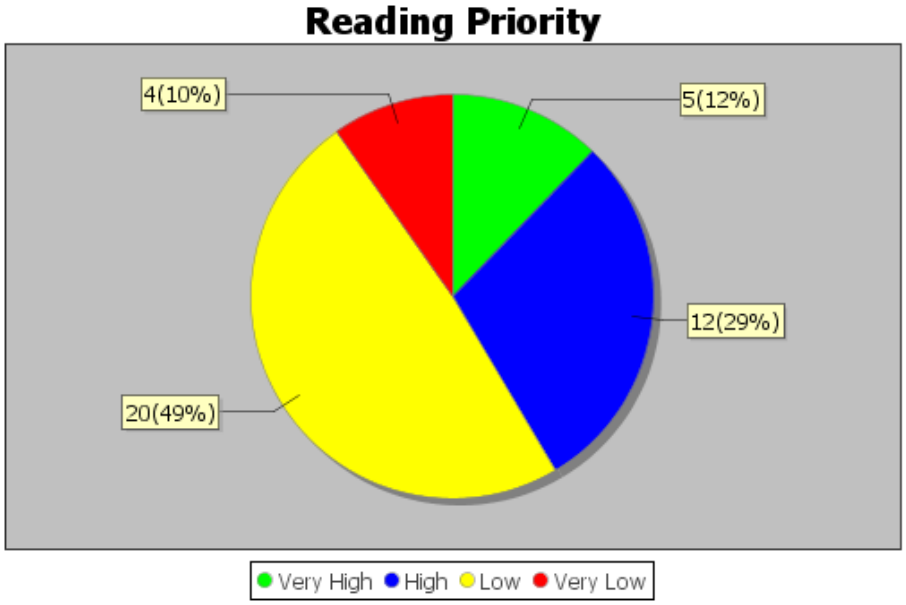

Figura 4 – Prioridade de leitura dos artigos

Fonte: Ferramenta StArt

Conforme apresentado na Figura 4, 10% dos artigos teve baixíssima prioridade de leitura, 49% dos artigos teve baixa prioridade de leitura, 29% teve alta prioridade de leitura e 12% teve altíssima prioridade de leitura.

Após a leitura completa dos artigos, 59% dos artigos foram aceitos, ou seja, 24 artigos responderam à questão de pesquisa e não apresentaram os demais critérios de exclusão. A Figura 5 apresenta este resultado.

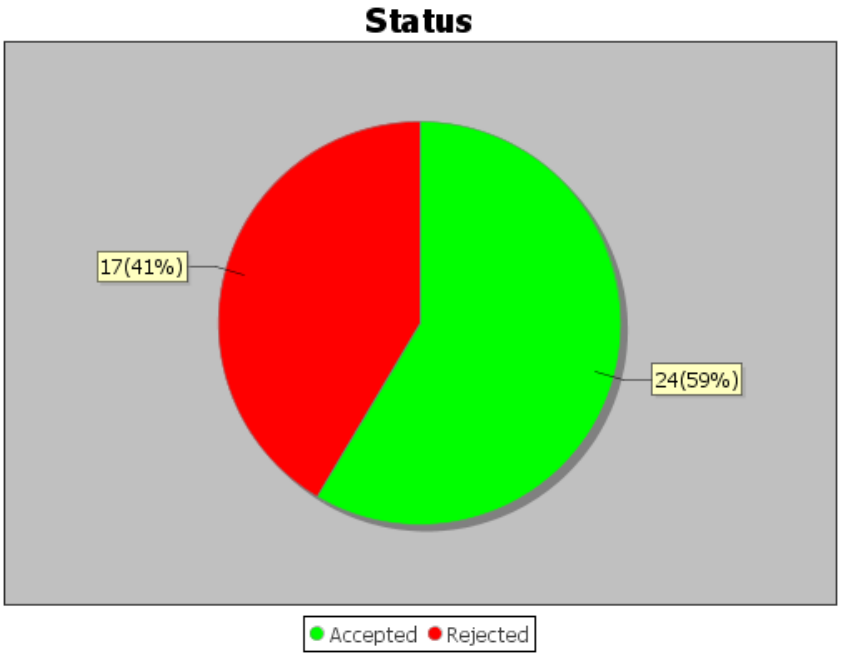

Figura 5 – *Status* dos artigos após leitura completa

Fonte: Ferramenta StArt

De acordo com a Figura 6, dos 17 artigos rejeitados, 16 artigos foram rejeitados pois não contribuíam para responder à questão de pesquisa, e 1 artigo foi rejeitado pois era um artigo incompleto.

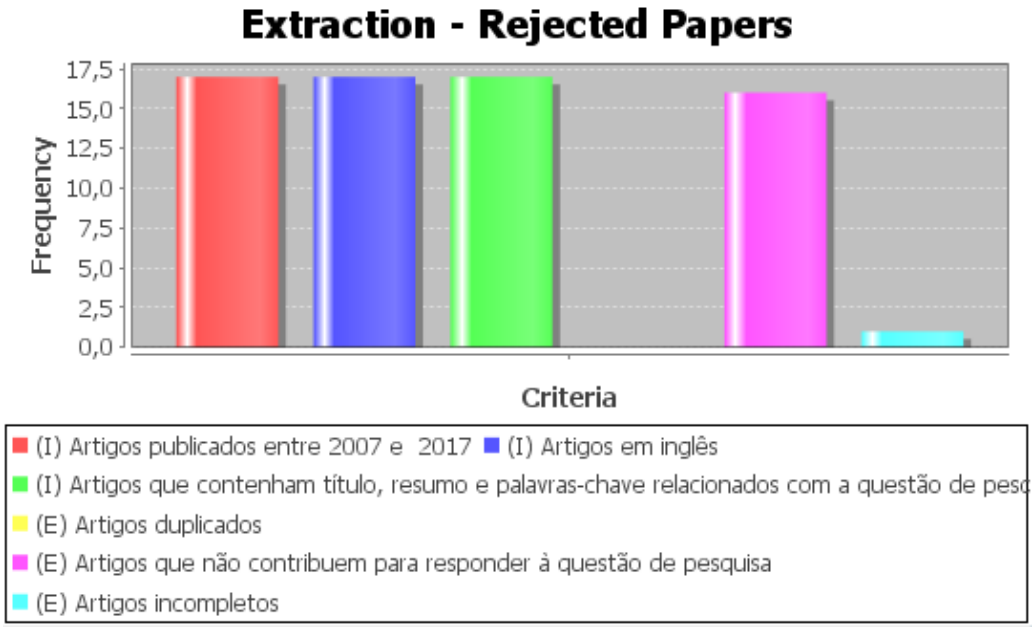

Figura 6 – Artigos rejeitados após a leitura completa

Fonte: Ferramenta StArt

Em adição aos 24 artigos selecionados pela busca automatizada, outros 4 artigos foram selecionados por meio da busca manual. Esses artigos não foram retornadas pela busca automatizada com a *string* de busca, porém foram encontrados por busca manual e foram considerados relevantes para serem adicionados a esse estudo.

Finalmente, após todo o procedimento, 28 artigos foram selecionados para a extração de dados. A Figura 7 apresenta a síntese de toda a coleta de dados após as 3 etapas, mais a busca manual.

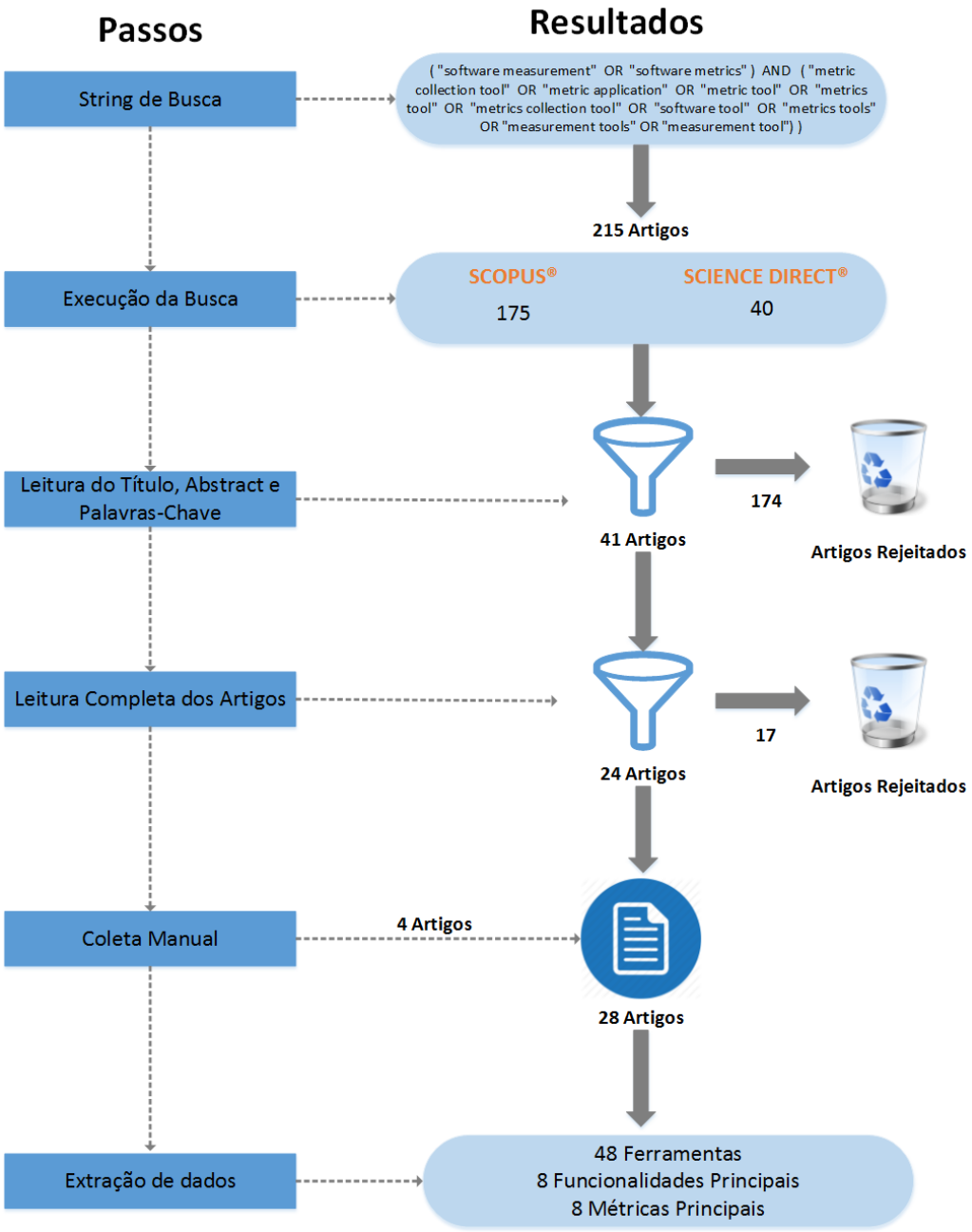

Figura 7 – Síntese da coleta de dados

A Tabela 1 apresenta a lista de artigos selecionados para extração de dados, com o identificador, título, autor, ano de publicação e a base que foi encontrado.

Tabela 1 – Artigos Selecionados para Extração de Dados

ID	Título do Artigo	Autor	Ano	Base
A1	A metrics tool for multi-language software	LINOS; LUCAS; MYERS; MAIER	2007	ACM Digital Library

A2	Design optimization metrics for UML based object-oriented systems	RAMARAJ; DURAISAMY	2007	World Scientific
A3	Managing software process measurement: A metamodel-based approach	GARCIA et al.	2007	Elsevier
A4	A metamodel for the measurement of object-oriented systems: An analysis using alloy	MCQUILLAN; POWER	2008	IEEE Xplore
A5	How to measure agile software development	KUNZ; DUMKE; SCHMIETENDORF	2008	Springer Link
A6	Software metrics for agile software development	KUNZ; DUMKE; ZENKER	2008	IEEE Xplore
A7	"Unit metrics- A tool to support refactoring in agile software development	KUNZ; ZENKER; MENCKE; DUMKE	2008	dblp
A8	Using a combination of measurement tools to extract metrics from open source projects	BAKAR; BOUGHTON et al.	2008	SEA 2008 - Conference Proceedings
A9	A coupling and cohesion metrics suite for object-oriented software	HUSEIN; OXLEY	2009	IEEE Xplore
A10	Analysis and implementation of software metric for object-oriented	DA-WEI	2009	IEEE Xplore
A11	A framework for source code metrics	MANEVA; GROZEV; LILOV	2010	ACM Digital Library
A12	Towards a 'Universal' software metrics tool: Motivation, process and a prototype	RAKIć; BUDIMAC; BOTHE	2010	ICSOFT 2010 - Conference Proceedings
A13	A pluggable tool for measuring software metrics from source code	HIGO et al.	2011	IEEE Xplore

A14	SMIILE prototype	RAKIĆ; BUDIMAC	2011	AIP Conference Proceedings
A15	XML-based integration of the SMIILE tool prototype and software metrics repository	RAKIĆ; GERLEC; NOVAK; BUDIMAC	2011	AIP Conference Proceedings
A16	Customizing GQM models for software project monitoring	MONDEN et al.	2012	J-STAGE
A17	Towards the better software metrics tool	BUDIMAC; RAKIC; HERICKO; GERLEC	2012	IEEE Xplore
A18	Validation of measurement tools to extract metrics from open source projects	BAKAR; BOUGHTON	2012	IEEE Xplore
A19	A Methodology for Obtaining Universal Software Code Metrics	NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO	2013	Elsevier
A20	ASSIST: An integrated measurement tool	KESER; IYIDOGAN; OZKAN	2013	IEEE Xplore
A21	An object based software tool for software measurement	SANTHOSHINI; ANBAZHAGAN	2015	IEEE Xplore
A22	Evaluating metrics at class and method level for java programs using knowledge based systems	UMAMAHESWARI; NATARAJAN; GHOSH	2015	ARPN Journals
A23	Integration of software measurement supporting tools: A mapping study	FONSECA; BARCELLOS; FALBO	2015	Conference: XV Brazilian Symposium on Software Quality
A24	A systematic literature review on software measurement programs	TAHIR; RASOOL; GENCEL	2016	Elsevier

A25	Building a user oriented application for generic source code metrics extraction from a metrics framework	NÚÑEZ-VARELA; PÉREZ-GONZÁLEZ; MARTÍNEZ-PÉREZ; CUEVAS-TELLO	2016	IEEE Xplore
A26	An ontology-based approach for integrating tools supporting the software measurement process	FONSECA; BARCELLOS; FALBO	2017	Elsevier
A27	Investigating differences and commonalities of software metric tools	AVERSANO; GRASSO; GRASSO; TORTORELLA	2017	SciTePress
A28	Source code metrics: A systematic mapping study	NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO	2017	Elsevier

3.1.6 Análise Qualitativa dos Estudos

Depois da realizada a seleção dos estudos para extração dos dados, foi feita a análise qualitativa desses estudos. Essa etapa visa avaliar a qualidade dos artigos, como parte dos critérios de inclusão e exclusão, ajudando na interpretação dos resultados, na investigação de possíveis diferenças e o que está gerando essas diferenças (KITCHENHAM; CHARTERS, 2007).

Neste estudo, a análise da qualidade dos estudos foi utilizada para auxiliar na interpretação dos resultados, de forma há verificar se há algum tipo de enviesamento no estudo.

Para realizar a avaliação qualitativa dos estudos, foram utilizadas 5 critérios baseados no estudo de (DYBÅ; DINGSØYR, 2008). Dessa forma, para cada um dos critérios, foi verificado se o artigo atendia ou não o critério, e dado uma pontuação de 1 se sim, e 0 se não. Assim, a maior pontuação que o artigo pode atingir na análise qualitativa é 5, e a menor é 0. Os critérios definidos foram (DYBÅ; DINGSØYR, 2008):

C1. O artigo é uma pesquisa (1) ou opinião de especialista (0)?

C2. Os objetivos da pesquisa estão claros?

C3. Existe uma descrição adequada do contexto em que a pesquisa foi realizada?

C4. A análise dos dados foi suficientemente rigorosa?

C5. Existe uma declaração clara dos resultados?

A Tabela 2 apresenta o resultado da análise qualitativa para cada um dos artigos selecionados.

Tabela 2 – Análise Qualitativa dos Artigos

Identificador do artigo	**C1**	**C2**	**C3**	**C4**	**C5**	**Total**
A1	1	1	1	1	1	5
A2	1	1	1	1	1	5
A3	1	1	1	1	1	5
A4	1	0	1	1	0	3
A5	1	0	0	0	1	2
A6	1	0	0	0	1	2
A7	1	0	0	0	1	2
A8	1	0	1	1	1	4
A9	1	1	0	0	1	3
A10	1	0	1	1	1	4
A11	1	0	1	0	1	3
A12	1	0	1	0	1	3
A13	1	0	1	1	1	4
A14	1	0	1	0	1	3
A15	1	0	1	0	1	3
A16	1	1	1	1	1	5
A17	1	0	1	0	1	3
A18	1	1	1	1	1	5
A19	1	0	0	0	1	2
A20	1	1	1	0	1	4
A21	1	1	0	0	0	2
A22	1	1	1	0	1	4
A23	1	1	1	1	1	5
A24	1	1	1	1	1	5
A25	1	1	0	1	1	4
A26	1	1	1	1	1	5
A27	1	1	1	1	1	5
A28	1	1	1	1	1	5

De acordo com a tabela, temos 15% dos artigos com avaliação inferior ao valor médio (3), 25% dos artigos com avaliação igual ao valor médio, e 57% com avaliação superior ao valor médio. Baseado nesses dados, nota-se que mais de 82% dos artigos está na média, ou acima dela. Baseado na análise qualitativa, existem indícios de que o enviesamento do estudo é baixo. Não é possível afirmar com plenamente sobre o enviesamento, uma vez que poucos critérios foram utilizados para avaliar os artigos e pela pontuação ser dada apenas em critérios de sim ou não.

4 Resultados

A revisão sistemática resultou em uma lista de 28 artigos que contribuíram para responder às questões de pesquisa. As Seções 4.1, 4.2, 4.3 e 4.4 apresentam os resultados da análise dos artigos em relação a cada uma das questões de pesquisa definidas.

4.1 Análise da Questão de Pesquisa 1

QP1. Quais são as ferramentas de medições encontradas na literatura?

Durante a revisão sistemática, foram encontradas 48 ferramentas de medições. Algumas ferramentas mostradas pelos artigos foram descontinuadas, ou seja, não são mais disponíveis para download ou uso. Por isso, para cada ferramenta encontrada por meio da revisão sistemática, foi verificado se essa ferramenta estava disponível. Para isso, foi verificado se o artigo apresentava o link para acesso e download, e também foram feitas buscas no Google. A Tabela 3 mostra as ferramentas encontradas, se estão disponível para download ou uso, e quais também o(s) artigo(s) que a referencia.

Tabela 3 – Ferramentas de medição

Nome da Ferramenta	**Disponível**	**Artigos que referenciam**
Analizo	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
AOPMetrics	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
ASSIST		(KESER; IYIDOGAN; OZKAN, 2013)
CCCC	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017) (AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
CCMETRICS	✓	(HUSEIN; OXLEY, 2009)

Chidamber and Kemerer Java Metrics	✓	(BAKAR; BOUGHTON et al., 2008) (BAKAR; BOUGHTON, 2012) (NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
CIDE	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
Code Analyzer	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
CodePro Analytix		(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
CMT++	✓	(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
CMTJAVA	✓	(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
DePress	✓	(FONSECA; BARCELLOS; FALBO, 2017) (FONSECA; BARCELLOS; FALBO, 2015)
DIONE		(FONSECA; BARCELLOS; FALBO, 2017)(FONSECA; BARCELLOS; FALBO, 2015)
DPT		(RAMARAJ; DURAISAMY, 2007)
DSS		(FONSECA; BARCELLOS; FALBO, 2017)
DXCore		(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
Eclipse Metrics Plugin 1.3.6	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017) (NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
EPM	✓	(MONDEN et al., 2012)

Essential metrics	✓	(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
Featureous	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
Framework for source code metrics		(MANEVA; GROZEV; LILOV, 2010)
GenMETRIC		(GARCIA et al., 2007)
GQM Tool		(FONSECA; BARCELLOS; FALBO, 2017) (TAHIR; RASOOL; GENCEL, 2016) (FONSECA; BARCELLOS; FALBO, 2015)
JArchitect	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
JAM	✓	(UMAMAHESWARI; NATARAJAN; GHOSH, 2015)
JHawk	✓	(BAKAR; BOUGHTON, 2012) (NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
JMetric	✓	(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
JStyle		(BAKAR; BOUGHTON et al., 2008) (BAKAR; BOUGHTON, 2012)
LocMetrics	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
MASU	✓	(HIGO et al., 2011)
McCabeIQ		(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013)
Metamodel for the Measurement of Object-Oriented Systems		(MCQUILLAN; POWER, 2008)

Metriflame		(FONSECA; BARCELLOS; FALBO, 2017) (TAHIR; RASOOL; GENCEL, 2016) (FONSECA; BARCELLOS; FALBO, 2015)
MMT		(LINOS; LUCAS; MYERS; MAIER, 2007)
Moose software analysis platform	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
Object Based Software Tool for Software Measurement		(SANTHOSHINI; ANBAZHAGAN, 2014)
OOMT		(DA-WEI, 2009)
Pri2mer		(TAHIR; RASOOL; GENCEL, 2016)
Rational Software Analyzer		(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
RSM	✓	(NÚÑEZ-VARELA; PEREZ-GONZALEZ; CUEVAS-TELLO; SOUBERVIELLE-MONTALVO, 2013) (BAKAR; BOUGHTON et al., 2008) (BAKAR; BOUGHTON, 2012)
SAS	✓	(TAHIR; RASOOL; GENCEL, 2016)
SDMetrics (tool for UML)	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
SMIILE		(BUDIMAC; RAKIC; HERICKO; GERLEC, 2012) (RAKIĆ; BUDIMAC, 2011) (RAKIĆ; GERLEC; NOVAK; BUDIMAC, 2011) (RAKIć; BUDIMAC; BOTHE, 2010)
Source Monitor	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
src2srcml	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017)
Stan4j	✓	(AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)

Understand	✓	(NUÑEZ-VARELA; PÉREZ-GONZALEZ; MARTÍNEZ-PEREZ; SOUBERVIELLE-MONTALVO, 2017) (AVERSANO; GRASSO; GRASSO; TORTORELLA, 2017)
UnitMetrics	✓	(KUNZ; DUMKE; ZENKER, 2008) (KUNZ; ZENKER; MENCKE; DUMKE, 2008) (KUNZ; DUMKE; SCHMIETENDORF, 2008)
User Oriented Metric Tool		(NÚÑEZ-VARELA; PÉREZ-GONZÁLEZ; MARTÍNEZ-PÉREZ; CUEVAS-TELLO, 2016)

Dentre as ferramentas que estão disponíveis, foi feita análise em relação ao tipo de métrica que elas coletam. Algumas ferramentas trabalham com métricas de produto, outras permitem a coleta de ambos os tipos de métricas. A Figura 8 apresenta a distribuição das ferramentas encontradas na literatura em relação ao tipo de métrica que coletam.

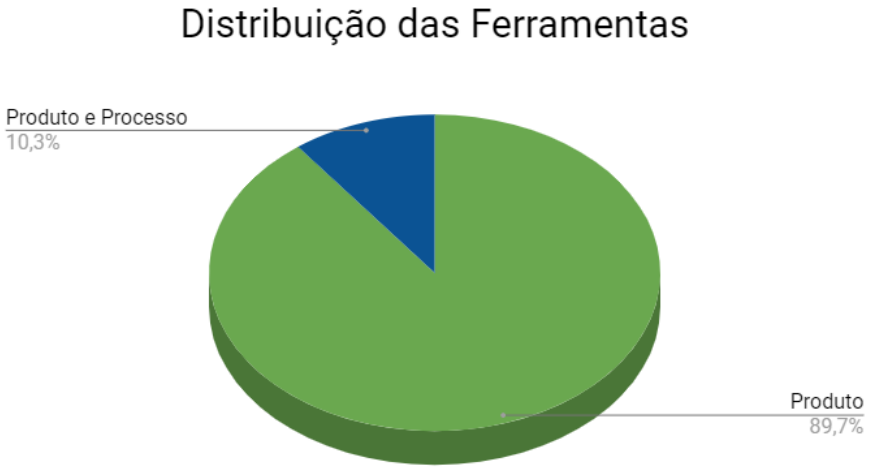

Figura 8 – Distribuição das ferramentas em relação à métricas fornecidas

Fonte: Autora

Das 29 ferramentas que estão disponíveis para uso, encontradas durante a revisão sistemática, 26 delas, cerca 90%, são ferramentas que coletam apenas medições de produto, ou seja, métricas de código. Nenhuma ferramenta fornece exclusivamente a coleta de métricas de processo, e outras 3 ferramentas realizam a coleta de ambos os tipos de métricas. A Tabela 4 apresenta a distribuição das ferramentas em relação ao tipo de métricas que elas coletam.

Tabela 4 – Distribuição das Ferramentas

Tipo de métrica	**Ferramentas**
Produto	Analizo, AOP Metrics, CCCC, CCMETRICS, Chidamber and Kemerer Java Metrics, CIDE, Code Analyzer, CMT++, CMTJAVA, Eclipse Metrics Plugin 1.3.6, Essential metrics, Featureous, JArchitect, JAM, JHawk, JMetric, LocMetrics, MASU, Moose software analysis platform, RSM, SDMetrics (tool for UML), Source Monitor, src2srcml, Stan4j, Understand, UnitMetrics
Produto e Processo	DePress, EPM, SAS

4.2 Análise da Questão de Pesquisa 2

QP2. Quais as funcionalidades as ferramentas de medições apresentam?

As ferramentas de medição exibidas pelos artigos, apresentam diferentes funcionalidades e características. Diversas ferramentas apresentam, por exemplo, suporte à uma única linguagem de programação, enquanto outras, apresentam suporte à várias linguagens. A Tabela 5 lista as principais funcionalidades das ferramentas. G = Gráficos, I = Integração, M = Múltiplas Linguagens de Programação, D = Definição de Métricas, R = Relatórios, E = Exportação de Dados, Im = Importação de Dados, H = Histórico.

A funcionalidade **Gráficos** significa que a ferramenta realiza a criação de algum tipo de gráfico para auxiliar na análise dos dados. A funcionalidade **Integração** significa que a ferramenta realiza integração com outras ferramentas. A funcionalidade **Múltiplas Linguagens de Programação** significa que a ferramenta atende mais do que uma linguagem de programação. A funcionalidade **Definição de Métricas** significa que a ferramenta permite ao usuário gerar métricas personalizadas. A funcionalidade **Relatórios** significa que a ferramenta gera algum tipo de relatório para análise das métricas. A funcionalidade **Exportação de Dados** significa que a ferramenta permite fazer download dos seus dados para algum formato utilizável em ferramentas como Excel ou outro leitor de textos. A funcionalidade **Importação de Dados** significa que a ferramenta permite que dados de fora dela sejam utilizados dentro dela, por meio de um procedimento de importação de dados. A funcionalidade **Histórico** significa que a ferramenta mantém os registros das medições realizadas anteriormente.

As funcionalidades **Gráficos** e **Relatórios** são as funcionalidades mais atendidas pelas ferramentas, aparecendo 14 e 13 vezes, respectivamente, dentro do conjunto de 29 ferramentas. As funcionalidades **Integração**, **Definição de Métricas**, **Importação de Dados** e **Histórico** são as menos contempladas pelas ferramentas, aparecendo menos de 5 vezes no conjunto de ferramentas. Dentro da funcionalidade **Definição de Métricas**, há algumas peculiaridades. Algumas das ferramentas permitem apenas a definição de métricas derivadas, ou seja, baseado no conjunto de métricas que a ferramenta fornece,

Tabela 5 – Principais Funcionalidades

Ferramenta	**G**	**I**	**M**	**D**	**R**	**E**	**Im**	**H**
Analizo	x		x		x			x
AOPMetrics	x							
CCCC			x			x		
CCMetrics	x		x					
Chidamber and Kemerer Java Metrics					x	x		
CIDE	x		x					
Code Analyzer			x		x			
CMT++			x		x			
CMTJAVA					x			
DePress		x			x	x	x	
Eclipse Metrics Plugin 1.3.6						x		
EPM	x			x			x	
Essential metrics			x		x			
Featureous	x	x						x
JArchitect		x			x			x
JAM	x							
JHawk	x			x	x			
JMetric					x			
LocMetrics	x		x					
MASU			x					
Moose software analysis platform	x				x			
RSM			x		x			
SAS					x			
SDMetrics (tool for UML)	x	x		x			x	
Source Monitor	x		x			x		x
src2srcml		x		x				
Stan4j	x				x			
Understand	x		x	x				
UnitMetrics	x					x		

o usuário pode combiná-las a fim de gerar novas métricas. Além disso, nenhuma das ferramentas permite que as métricas já inseridas sejam alteradas futuramente, ou até mesmo que a definição da métrica seja alterada.

4.3 Análise da Questão de Pesquisa 3

QP3. Quais as vantagens e desvantagens das ferramentas de medições?

Fenton e Pfleeger (1997) listam uma série de características importantes que uma ferramenta de métricas deve apresentar. Baseado nisso, foi derivada uma lista de vantagens

e desvantagens para as ferramentas.

As principais vantagens que uma ferramenta pode apresentar são (FENTON; PFLEEGER, 1997):

- Utilizar técnicas que aderem a padrões de qualidade do processo de medição;
- Preocupar-se com experiência do usuário.

Atender à padrões de qualidade significa que a ferramenta está preocupada em seguir padrões definidos por ISO ou outros padrões internacionais. Preocupação com a experiência de usuário significa, além de ter uma interface gráfica, apresentar meios que facilitam a usabilidade.

Como desvantagem, as seguintes características foram consideradas (FENTON; PFLEEGER, 1997):

- Ser difícil de utilizar do ponto de vista do usuário.
- Não fornecer meios de automatizar a coleta;

Ser difícil de utilizar significa que a ferramenta não possui nenhum tipo de interface gráfica, ou seja, utiliza linhas de comando. Não fornecer meios de automatização significa que a ferramenta apenas permite a coleta de métricas por meio de formulários que o usuário preenche.

Foi observado, em cada uma das ferramentas, se elas se enquadram em alguma das vantagens ou desvantagens apresentadas. Pela análise, foram obtidos os seguintes dados:

- Utilizar técnicas que aderam a padrões de qualidade do processo de medição:

- Essential metrics;
- RSM;

- Preocupação com experiência do usuário:

- Code Analyzer;
- JHawk;
- SDMetrics;
- Understand;

- Ser difícil de utilizar do ponto de vista do usuário:

- Analizo
- CCCC;
- Chidamber and Kemerer Java Metrics;
- CMT++;
- CMT JAVA;
- Essential Metrics;
- src2srcml;

  - *Não fornecer meios de automatizar a coleta:*

- Todas as ferramentas analisadas apresentam métodos de automatiza a coleta.

4.4 Análise da Questão de Pesquisa 4

QP4. Quais as métricas identificadas e utilizadas pelas ferramentas de medições?

A maioria das ferramentas de métricas identificadas durante a revisão sistemática apresentam conjuntos base de métricas. Estes conjuntos representam as métricas que o usuário pode utilizar, quando a ferramenta não permite a criação de métricas.

A Tabela 6 apresenta as principais métricas que as ferramentas coletam, a quantidade de vezes na qual as métricas aparecem, e a porcentagem total para cada uma das métricas.

Tabela 6 – Métricas apresentadas pelas ferramentas

Métrica	Ocorrência	Porcentagem	Ferramentas
Linhas de Código (LOC)	15	51,7%	Analizo, AOPMetrics, CCCC, CIDE, CMT++, CMTJAVA, Code Analyzer, Eclipse Metrics Plugin, JAM, LOCMetric, RSM, Source Monitor, Stan4J, Understand, UnitMetrics
Complexidade Ciclomática (CC)	13	44,8%	Analizo, CCCC, CMT++, CMTJAVA, Eclipse Metrics Plugin, Essential Metrics, JAM, JArchtect, JMetric, MASU, SDMetrics (tool for UML), Stan4J, Understand
Acoplamento entre objetos (CBO)	12	41,4%	Analizo, CCCC, Chidamber and Kemerer Java Metrics, Essential Metrics, JAM, JArchtect, JHawk, MASU, RSM, Stan4J, Understand, UnitMetrics
Profundidade de Herança (DIT)	11	37,9%	Analizo, CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU, RSM, SDMetrics (tool for UML), Stan4J, Understand
Número de Filhos (NOC)	11	37,9%	Analizo, CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU, RSM, SDMetrics (tool for UML), Stan4J, UnitMetrics
Lack de coesão dos métodos (LCOM)	8	27,6%	Analizo, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU, Stan4J, Understand
Métodos ponderados por classe (WMC)	8	27,6%	Analizo, CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JHawk, MASU, Stan4J
Resposta por classe (RFC)	5	17,2%	Analizo, Chidamber and Kemerer Java Metrics, JAM, MASU, Stan4J

A métrica linhas de código é a métrica mais relevante dentro do conjunto de ferramentas apresentadas neste estudo, aparecendo em mais de 50% das ferramentas analisadas. É seguida pelas métricas complexidade ciclomática e acoplamento entre objetos.

Várias outras métricas são propostas pelas ferramentas, porém todas elas tem taxa de ocorrência menor do 3%. Desta forma, não é relevante o estudo destas métricas.

As Seções 4.4.1, 4.4.2, 4.4.3, 4.4.4, 4.4.5, 4.4.6, 4.4.7 e 4.4.8 contém a descrição das métricas apresentadas e também um exemplo de como calculá-las.

4.4.1 Linhas de código

A métrica linhas de código (*Lines of Code* - LOC) faz a contagem do tamanho do programa medindo a quantidade de linhas de código que ele contém (UMAMAHESWARI; NATARAJAN; GHOSH, 2015)

Uma forma de como calcular esta métrica é apresentada por (FENTON; PFLEEGER, 1997). Ele apresenta duas métricas para fazer o cálculo, a métrica de linhas de código não comentadas (*Non-commented Line of Code* - NCLOC) e a métrica linhas de comentários do programa (*Comment lines of Program Text* - CLOC). Assim, para calcular a métrica linhas de código:

$$LOC = NCLOC + CLOC \tag{4.1}$$

Esta medida é em escala absoluta.

4.4.2 Complexidade Ciclomática

A métrica complexidade ciclomática mede o número de caminhos linearmente independentes (FENTON; PFLEEGER, 1997).

Para calcular a complexidade ciclomática em um software com fluxograma F:

$$v(F) = e - n + 2 \tag{4.2}$$

Onde e representam os arcos e n os nós.

Quanto maior for a complexidade ciclomática de um sistema, mais difícil ele será de manter (FENTON; PFLEEGER, 1997).

4.4.3 Acoplamento entre objetos

A métrica acoplamento entre objetos (*Coupling Between Objects* - CBO) calcula o grau de dependência entre módulos (FENTON; PFLEEGER, 1997)

Um exemplo de como calcular esta métrica é apresentado por (FENTON; MELTON, 1990). No caso apresentado por ele, é mostrado como calcular o acoplamento entre dois módulos. Pra calcular o acoplamento entre esses módulos, primeiramente é necessário classificar os tipo de relacionamento existente. Os relacionamentos podem ser (FENTON; MELTON, 1990):

- R5: Se x se ramifica de y, altera dados ou declarações de y;
- R4: Se x e y fazem referencia ao mesmo dado global;
- R3: Se x passa parâmetros para y visando controlar seu comportamento;

- R2: Se x e y aceitam o mesmo tipo de registro como parâmetro;
- R1: Se x e y se comunicam por parâmetros;
- R0: Se x e y não fazem comunicações.

A fórmula apresentada é:

$$M(x, y) = i + \frac{n}{n+1} \tag{4.3}$$

Onde, i representa o maior tipo de relacionamento, e n representa o número de interconexões entre x e y (FENTON; MELTON, 1990). Esta medida é em escala ordinal.

4.4.4 Profundidade de herança

A métrica profundidade de herança (*Depth of Inheritance Tree* - DIT) calcula a profundidade da classe na hierarquia de classes do projeto. Quanto mais profundo uma classe estiver, mais difícil é de prever seu comportamento (UMAMAHESWARI; NATARAJAN; GHOSH, 2015) (HIGO et al., 2011).

4.4.5 Número de filhos

A métrica número de filhos (*Number of Children* - NOC) calcula a quantidade de classes derivadas da classe que está sendo medida (HIGO et al., 2011).

O cálculo desta métrica é feito pelo somatório de filhos imediatos da classe (FENTON; PFLEEGER, 1997).

4.4.6 Lack de coesão dos métodos

A métrica lack de coesão dos métodos (*Lack of Cohesion of Methods* - LCOM) mede a discrepância de métodos em uma classe mediante as variáveis instanciadas (UMAMAHESWARI; NATARAJAN; GHOSH, 2015).

Uma forma de cálculo para essa métrica é apresentada por (HIGO et al., 2011), e é feita pegando cada par de métodos dentro da mesma classe, verificando se eles possuem acesso disjunto à um conjunto de variáveis de instância. Se sim, aumentar P em um. Porém, se eles tiverem pelo menos uma variável em comum, aumentar P em um. Dessa forma:

$$LCOM = \begin{cases} P - Q & \text{se P} > \text{Q} \\ 0 & \text{se não} \end{cases} \tag{4.4}$$

4.4.7 Métodos ponderados por classe

A métrica métodos ponderados por classe (*Weighted Methods per Class* - WMC) realiza a soma das complexidades dos métodos na classe (HIGO et al., 2011).

O cálculo desta métrica é feito pelo somatório de cada métodos ponderados com sua complexidade (FENTON; PFLEEGER, 1997). Desta forma, temos:

$$WMC = \sum_{i=1}^{n} c_i \tag{4.5}$$

4.4.8 Resposta por classe

A métrica resposta por classe (*Responde for Class* - RFC) representa o tamanho de um conjunto de respostas para a classe. Este conjunto de resposta consiste de todos os métodos chamados por métodos locais (FENTON; PFLEEGER, 1997).

O cálculo desta métrica é feito pela soma de número de métodos locais mais o número de métodos chamados por métodos locais (FENTON; PFLEEGER, 1997).

5 Desenvolvimento da Ferramenta

Este Capítulo apresenta todo os procedimentos para o desenvolvimento da ferramenta. A Seção 5.1, apresenta a metodologia que será utilizada para desenvolver a ferramenta. A Seção 5.2 apresenta o planejamento da implementação da ferramenta. A Seção 5.3 apresenta como foi feito a configuração do ambiente de programação, assim como a linguagem de programação escolhida e o como foi o versionamento do sistema. A Seção 5.4 apresenta a arquitetura escolhida para desenvolver o sistema e o modelo de dados da aplicação. A Seção 5.5 apresenta as funcionalidades do sistema, descritas como histórias dos usuários, os critérios de aceitação e as telas da aplicação. A Seção 5.6 apresenta como foi feita a validação da ferramenta desenvolvida. A Seção 5.7 apresenta como foi o processo de integrar as histórias desenvolvidas e como foi feito o *deploy* da ferramenta. A Seção 5.8 apresenta a comparação da ferramenta desenvolvida com as demais ferramentas do estudo. Finalmente, a Seção 5.9 apresenta sugestões de funcionalidades para serem desenvolvidas no futuro.

5.1 Metodologia de Desenvolvimento

A metologia de desenvolvimento utilizada para desenvolver a ferramenta de base histórica de medição flexível será uma versão simplificada da metodologia ágil Scrum. A Figura 9 apresenta o processo de desenvolvimento que será utilizado para a implementação da ferramenta de medição.

A primeira atividade do processo corresponde à "Levantar histórias de usuários". As histórias de usuário serão escritas o formato "Eu, como usuário..., desejo fazer..., para obter...".

A segunda atividade do processo é "Definir critérios de aceitação". Os critério de aceitação define como é esperado que o sistema se comporte (LAPOLLI; CRUZ; MOTTA; TOLLA, 2010). Cada história de usuário, tem uma série de critérios de aceitação.

A terceira atividade é "Implementar história". Durante esta atividade, a implementação da história de usuário será feita, seguindo os critérios de aceitação definidos.

A quarta atividade corresponde à "Realizar validação da história". Esta atividade irá checar se a história implementada atende aos critérios de aceitação que foram definidos anteriormente. Se não atender ao critérios, deverá ser executada a quinta atividade "Fazer correção da história", que corresponde à corrigir os erros reportados durante a atividade de validação.

A sexta atividade corresponde à "Fazer integração da história". Após uma história

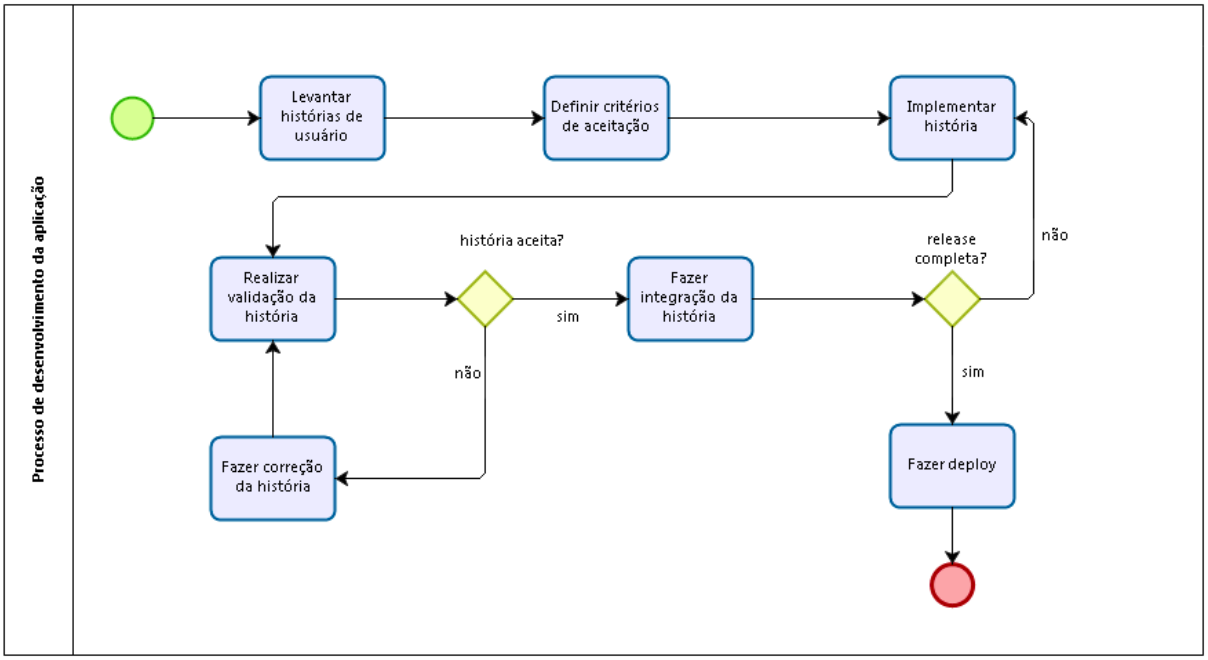

Figura 9 – Processo de desenvolvimento da aplicação

Fonte: Autora

estar implementada e validada, deve ser feito a integração dessa história com o conjunto de outras histórias implementadas, a fim de construir uma *release* para entrega.

Caso a *release* não esteja completa, novas histórias de usuário deverão ser implementadas. Se a *release* estiver completa, a sétima atividade "Fazer *deploy*"deverá ser realizada. Esta atividade corresponde à implantação da *release* no ambiente de produção, ou seja, o ambiente no qual o usuário irá utilizar a ferramenta. Após a atividade de *deploy*, o processo é encerrado.

5.2 Planejamento da Implementação

Foi desenvolvido um cronograma preliminar para o desenvolvimento da ferramenta de base histórica flexível. A Tabela 7 apresenta o planejamento da implementação da ferramenta.

Tabela 7 – Planejamento da implementação da ferramenta

Atividade	**Março**	**Abril**	**Maio**	**Junho**	**Julho**
Configuração do ambiente	x				
Levantamento das histórias de usuário		x			
Definição dos critérios de aceitação		x			
Implementação das histórias			x	x	
Validação das histórias			x	x	
Integração das histórias			x	x	
Deploy					x

5.3 Configuração do Ambiente

O primeiro passo para iniciar a implementação da ferramenta consistiu na configuração do ambiente para programação. Foi necessário definir qual seria a linguagem de programação a ser utilizada, explicado na Seção 5.3.1. Além disso, definir como seria feito o versionamento do código em implementação, conforme o explicado na Seção 5.3.2.

5.3.1 Linguagem de Programação

A linguagem de programação escolhida foi Ruby[1]. É uma linguagem de programação orientada a objetos, interpretada e com uma sintaxe simples. Essa linguagem foi escolhida por familiaridade da autora com a linguagem de programação.

Além disso, outro motivo da escolha dessa linguagem, é porque a linguagem Ruby contém o *framework* Ruby on Rails[2]. Ruby on Rails é um *framework* de código aberto, orientado a convenção e voltado para aplicações web. Foi definido que a ferramenta desenvolvida nesse trabalho seria uma aplicação web, para facilitar o acesso dos usuários a ela e aumentar o alcance dessa ferramenta, uma vez que não seria necessário nenhuma instalação adicional por parte dos usuários.

5.3.2 Controle de Versão

O controle de versão para desenvolver a ferramenta foi feito no GitHub[3]. O repositório que mantém a ferramenta é o <https://github.com/KarineValenca/hbfm>.

O controle de versão permite que versões estáveis do código sejam mantidas, enquanto novas funcionalidades, que ainda estão instáveis, sejam desenvolvidas. Assim, no *branch master* do projeto, existe apenas código estável, enquanto nos demais *branches*, existem novas funcionalidades sendo desenvolvidas. A Figura 10 mostra a organização de *branches* no projeto.

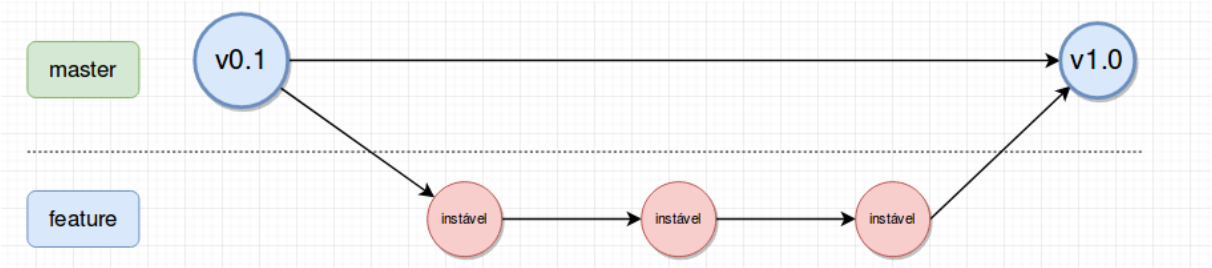

Figura 10 – Organização de *Branches*

Fonte: Autora

[1] Disponível em: https://www.ruby-lang.org/pt/
[2] Disponível em: https://rubyonrails.org/
[3] Disponível em https://github.com/

Inicialmente, o primeiro *branch* criado foi o master, na sua versão 0.1. Para cada funcionalidade a ser desenvolvida, um novo *branch* foi criado, e várias versões instáveis do código foram sendo desenvolvidas até que a funcionalidade estivesse pronta e estável. Assim, um *pull request* para o *branch* master era criado, e uma nova versão estável do código estava pronta, na versão 1.0. Esse procedimento foi realizado desenvolver cada uma das funcionalidades da ferramenta.

5.4 Arquitetura do Sistema

O sistema desenvolvido segue alguns padrões e boas práticas adotadas por engenheiros de software e programadores *Ruby on Rails*. Um dos padrões mais adotados para programação web, e apoiado pelo *Framework Ruby on Rails* é o *Model-View-Controller* (MVC), que foi utilizado no desenvolvimento dessa ferramenta. Detalhes do MVC estão melhor explicados na Seção 5.4.1. Outro padrão utilizado em desenvolvimento de software, é o banco de dados relacional. O banco relacional utilizado nesse projeto, está explicado na Seção 5.4.2

5.4.1 Model-View-Controller

A arquitetura MVC foi criada visando aumentar a reutilização e capacidade de conexão dos softwares. Ela aplica o chamado "fatoração em três vias", onde há separação entre as partes que representam o modelo de domínio da aplicação (*Model*), as partes que são apresentadas ao usuário (*View*) e a forma com a qual o usuário interage com o sistema (*Controller*) (KRASNER; POPE et al., 1988).

O *Ruby on Rails* utiliza dessa arquitetura, separando os conceitos, e deixando cada parte com sua responsabilidade bem definida. A Figura 11 apresenta graficamente uma arquitetura MVC. A *model*, cria as validações e associações entre outros modelos, e só deve interagir diretamente com a *controller*. A *view* apresenta os dados para os usuários e só deve interagir diretamente com a *controller*. A *controller* cuida das decisões e fluxos que devem acontecer no sistema, e é a camada que intermedeia a comunicação entre a *model* e a *view* (TK, 2017).

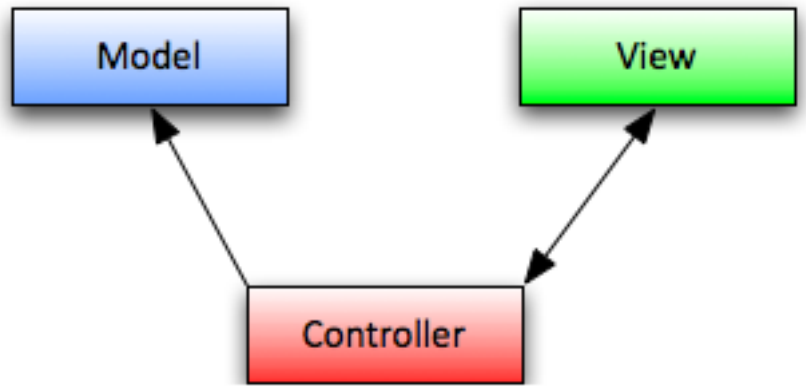

Figura 11 – Modelo MVC

Adaptado de <https://medium.com/the-renaissance-developer/ruby-on-rails-http-mvc-and-routes-f02215a46a84>

Essa arquitetura foi utilizada no desenvolvimento do sistema. Durante toda a implementação, houve cuidado para não ferir a arquitetura, que pode ser facilmente quebrada ao fazer referências da *model* diretamente na *view*. É esperado, então, que a ferramenta seja mais fácil de evoluir, de realizar manutenções.

5.4.2 Modelo de Dados

A ferramenta desenvolvida nesse projeto, utiliza um banco de dados relacional. A definição de um banco de dados relacional corresponde "à um conjunto não vazio finito D (domínio), onde existe um número finito de relações R em D"(CHANDRA; MERLIN, 1977). Assim, em um banco de dados relacional, as relações fazem referência ao domínio que elas pertencem.

Um banco de dados relacional é formado por tabelas, que contém categorias de interesse. Cada uma dessas tabelas, possui colunas que são os dados a serem coletados das categorias. E também possuem linhas, que são uma instância única dos dados (LEAVITT, 2010). Foi escolhida esse tipo de modelo de dados, pois os modelos relacionais fornecem uma representação uniforme para os dados. Mesmo não tendo desempenho de acesso alto, como abordagens não relacionais, esse modelo atende bem as necessidades do projeto (LEAVITT, 2010) (GARDARIN; VALDURIEZ, 1989).

A Figura 12 apresenta o modelo de dados da ferramenta. A tabela *users* mantém os usuários do sistema. A tabela *unit_of_measurements* mantém as unidades de medida do usuário e precisa de um id de usuário para ser criada. A tabela *metrics* mantém as métricas do usuário, e precisa de um id de usuário e um id de unidade de medida. A tabela *final_measures* mantém as medidas finais que são geradas após o usuário combinar as medidas, e necessita de um id de métrica para ser criada. A tabela *measures* mantém as medidas de uma medida final, e necessita de um id de medida final para ser criada.

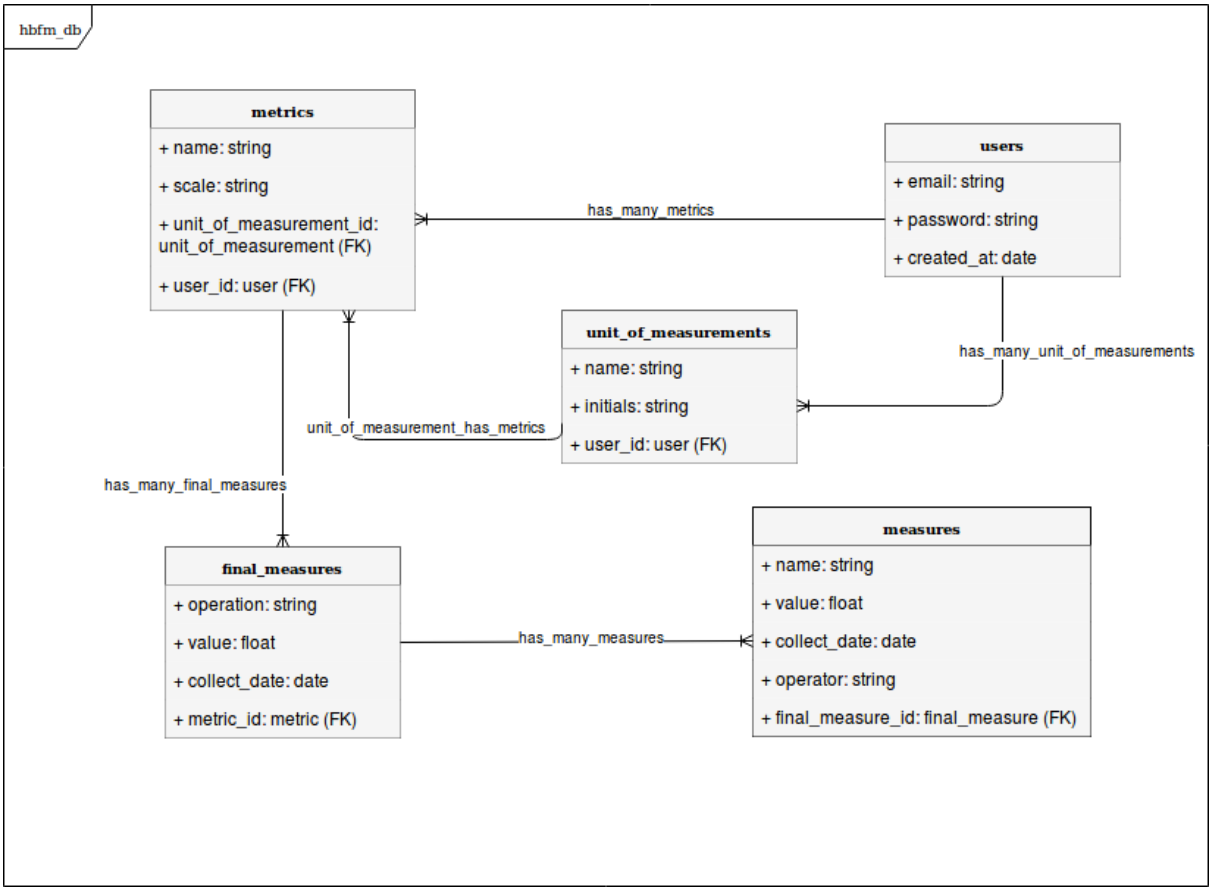

Figura 12 – Modelo de dados da ferramenta

Fonte: Autora

5.5 Histórias de usuário

Segundo (REES, 2002), uma história de usuário corresponde à "um pequeno pedaço de funcionalidade do software final, percebido do ponto de vista do usuário final". Desta forma, as histórias de usuário devem ser entregáveis que contenham valor de uso para o usuário do sistema.

As histórias de usuário desenvolvidas neste projeto, foram criadas com base nessa definição. A seguir, são apresentadas as histórias de usuário definidas para a ferramenta de base histórica de medição.

5.5.1 US01: Criar/Editar Unidade de Medidas

Definição da História:

Eu, como usuário, desejo cadastrar e editar unidades de medidas, a fim de determinar a grandeza das medidas de software.

Critérios de Aceitação:

- Uma unidade de medida deve conter os campos: nome e sigla;
- Nome deve conter até 20 caracteres;
- Sigla deve conter até 7 caracteres;

Telas da aplicação:

Acessar página de unidades de medida:

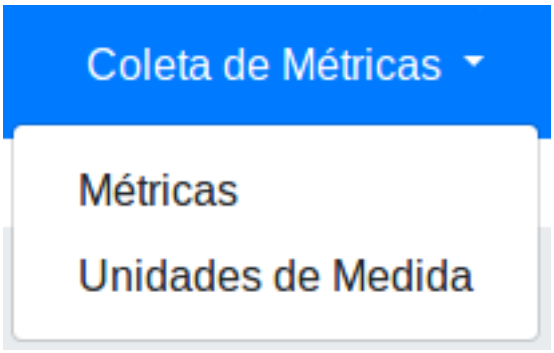

Figura 13 – Acessar página de unidades de medida

Fonte: Autora

Cadastro de unidades de medida:

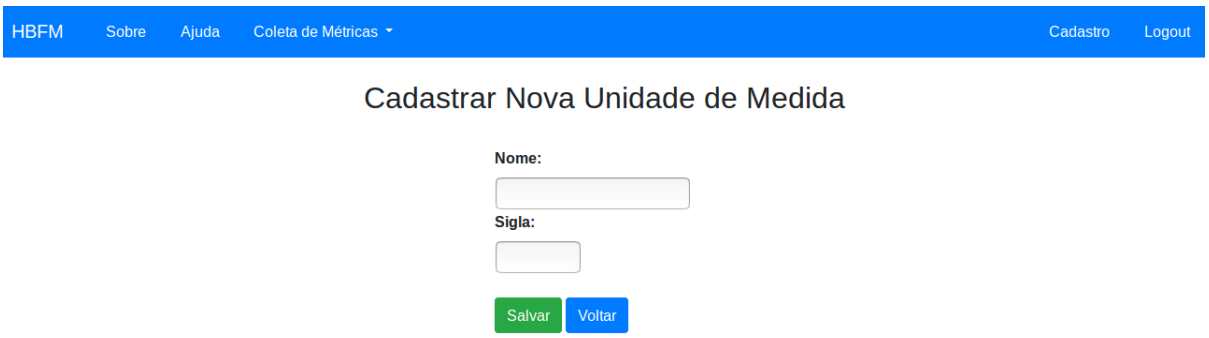

Figura 14 – Cadastro de unidades de medida

Fonte: Autora

Edição de unidades de medida:

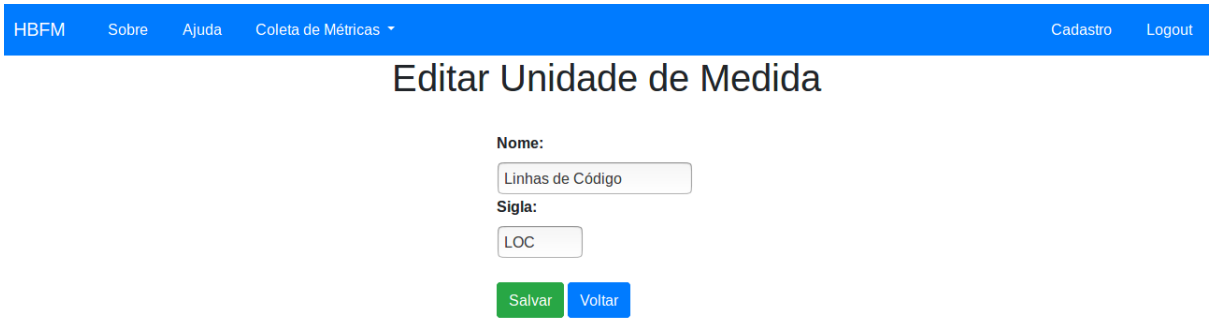

Figura 15 – Edição de unidades de medida

Fonte: Autora

5.5.2 US02: Criar/Editar Métricas

Descrição da História:

Eu, como usuário, desejo cadastrar e editar métricas no sistema, a fim de manter o histórico das medidas no sistema.

Critérios de Aceitação:

- Uma métrica deve conter os campos: Nome, Escala de medição, Unidade de medida;
- Nome deve conter no máximo 200 caracteres;
- Escala de medição pode ser: Nominal, Ordinal, Intervalar, Ratio ou Absoluta;
- Unidade de medida: Deve listar alguma unidade de medida criada anteriormente pelo usuário;
- Caso não exista unidade de medida cadastrada, o usuário deve ser redirecionado para a página de criação de unidades de medida.

Telas da aplicação:

Acessar página de métricas:

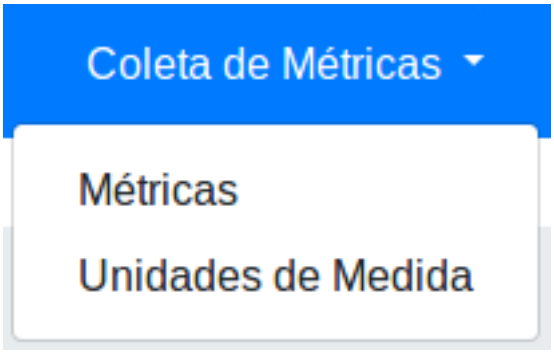

Figura 16 – Acessar página de métricas

Fonte: Autora

Cadastro de métricas:

Figura 17 – Cadastro de métricas

Fonte: Autora

Edição de métricas:

Figura 18 – Edição de métricas

Fonte: Autora

5.5.3 US03: Criar/Editar Medidas

Descrição da História:

Eu, como usuário, desejo cadastrar e editar medidas no sistema, a fim de adicionar dados às métricas e manter o histórico das medidas.

Critérios de Aceitação:

- Uma medida deve conter os campos: Nome da Medida, Valor da Medida, Data de Coleta;
- Nome da medida deve ter até 200 caracteres;
- Valor da Medida pode ser inteiro ou decimal;
- Data de coleta deve seguir o padrão DD/MM/YYYY;
- Deve haver um campo que permita ao usuário combinar várias medidas, utilizando os operadores +, -, *, ou / ;
- Deve haver uma tela que mostra o resultado final da medida.

Telas da Aplicação:

Acessar cadastro de medidas:

Figura 19 – Acessar cadastro de medidas

Fonte: Autora

Cadastro de Medidas:

Figura 20 – Cadastro de medidas

Fonte: Autora

Edição de Medidas:

Figura 21 – Edição de medidas

Fonte: Autora

5.5.4 US04: Visualizar Unidades de Medida

Descrição da História:

Eu, como usuário, desejo visualizar dados referentes às minhas unidades de medidas, a fim de saber quais unidades de medidas já estão cadastradas

Critérios de Aceitação:

- Deve haver uma lista com todas as unidades de medidas do usuário logado;
- Deve ser mostrado o nome da unidade de medida, a sigla;

- Deve haver um botão para editar a unidade de medida.

Telas da Aplicação:

Lista de Unidades de Medida:

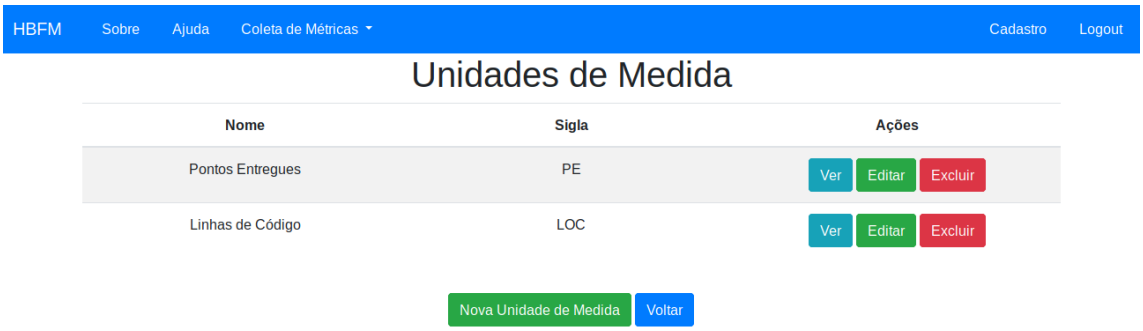

Figura 22 – Lista de Unidades de Medida

Fonte: Autora

Visualização de uma Unidade de Medida

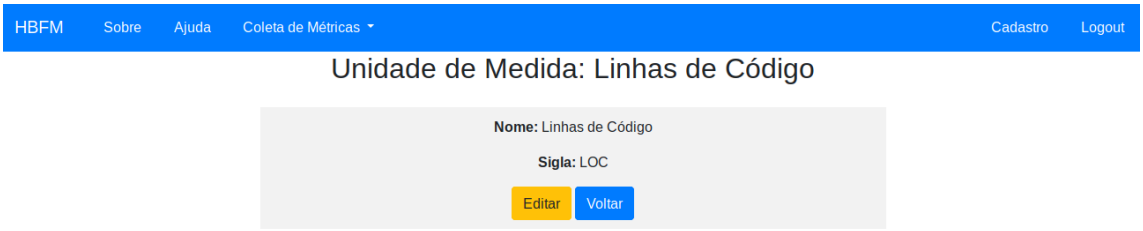

Figura 23 – Visualização de uma Unidade de Medida

Fonte: Autora

5.5.5 US05: Visualizar Métricas

Descrição da História:

Eu, como usuário, desejo visualizar dados referente às minhas métricas a fim de entender melhor os processos da organização.

Critérios de Aceitação:

- Deve haver uma lista com todas as métricas do usuário logado;
- Deve ser mostrado o nome da métrica, unidade de medida, escala de medição;
- Deve haver um gráfico em pizza caso a escala de medição seja nominal ou ordinal;

- Deve haver um gráfico de área, caso a escala de medição seja intervalar, ratio ou absoluta;
- Deve haver uma lista com todas as medidas da métrica, com um botão que permite a edição.

Telas da Aplicação:

Lista de Métricas:

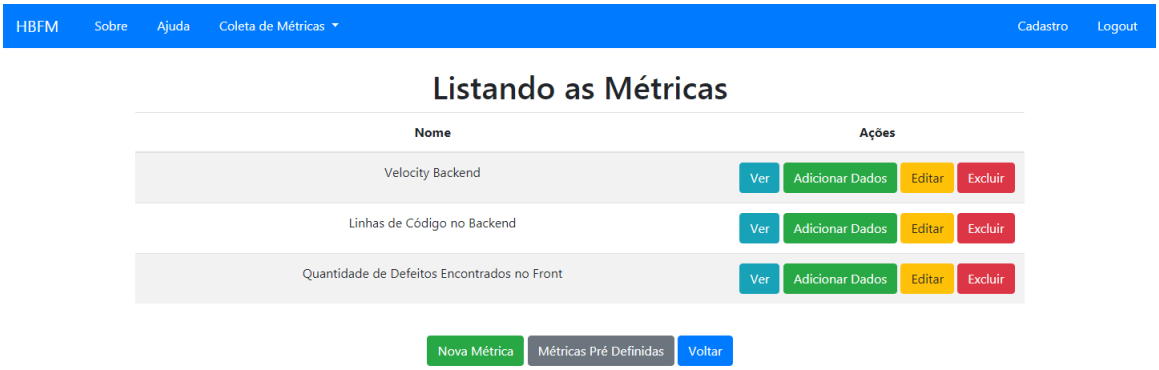

Figura 24 – Lista de métricas

Fonte: Autora

Visualização de uma Métrica Absoluta

Figura 25 – Linhas de Código - Métrica Absoluta

Fonte: Autora

Visualização de uma Métrica Ordinal

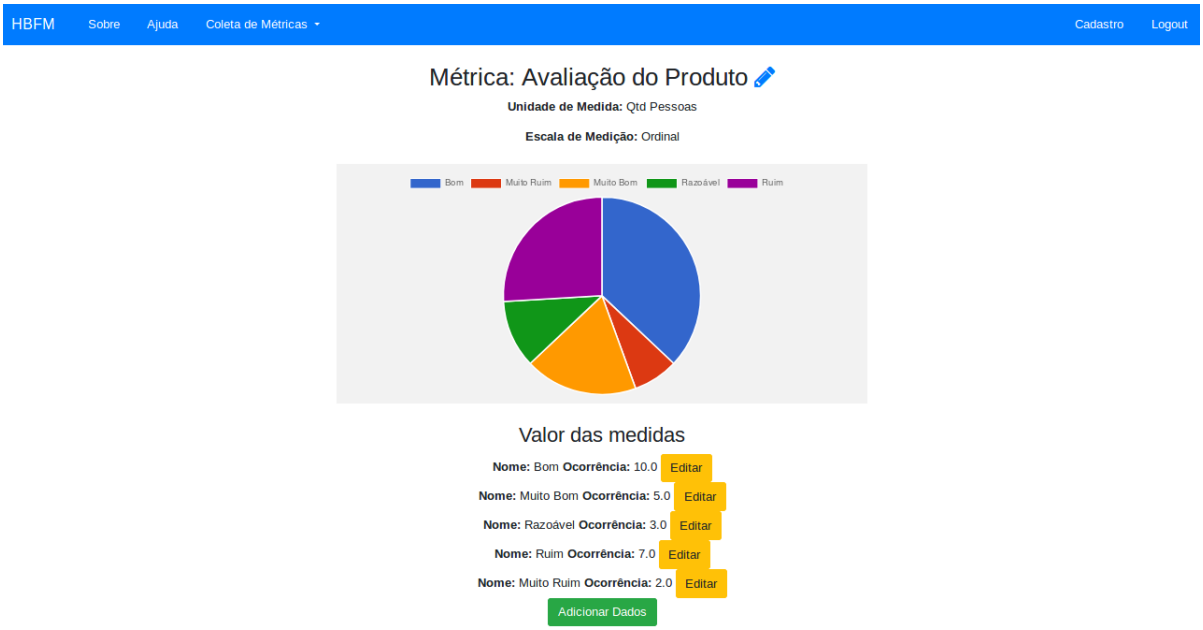

Figura 26 – Avaliação do Produto - Métrica Ordinal

Fonte: Autora

5.5.6 US06: Criar Usuários

Descrição da História:

Eu, como gerente de projeto, desejo me cadastrar e logar no site, para poder acompanhar as métricas do meu projeto.

Critérios de Aceitação:

- Cadastro: Deve pedir e-mail, senha e confirmação de senha;
- Login: Deve pedir e-mail e senha;
- Logout: Deve ser possível deslogar do sistema;
- Senha deve ter pelo menos 6 caracteres.

Telas da Aplicação:

Acessar página de cadastro ou de login

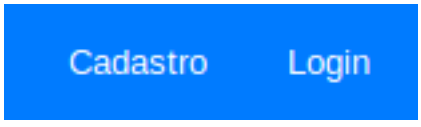

Figura 27 – Acessar página de cadastro ou de login

Fonte: Autora

Cadastro de Usuários

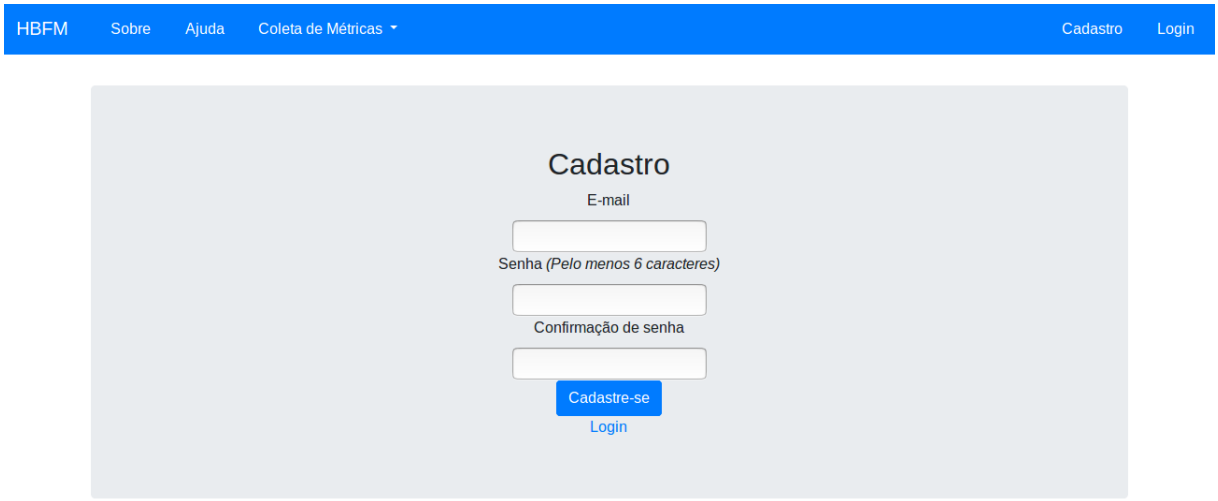

Figura 28 – Cadastro de Usuários

Fonte: Autora

Login no Sistema

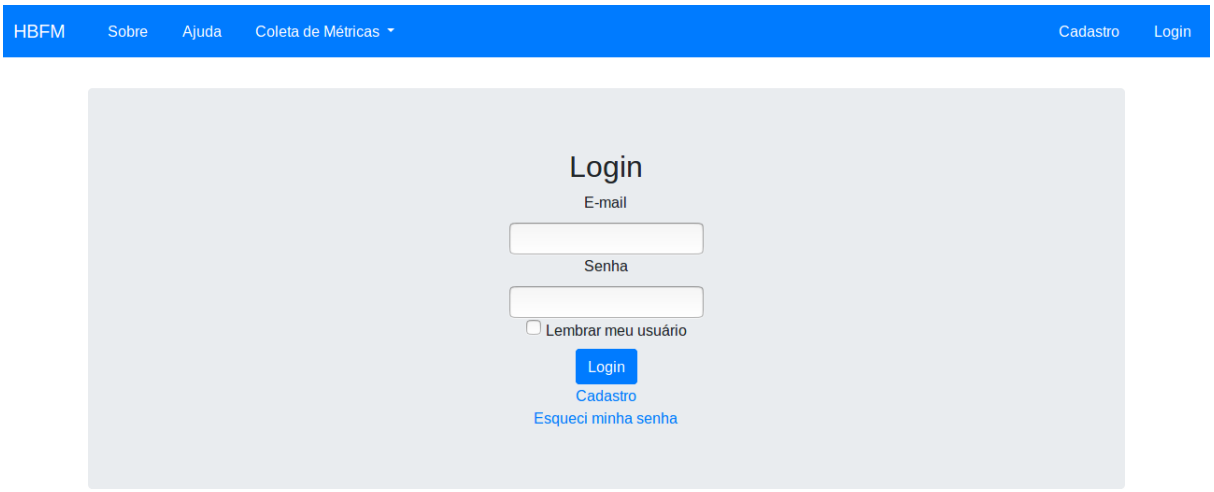

Figura 29 – Login no Sistema

Fonte: Autora

5.5.7 US07: Excluir Unidades de Medida

Descrição da História:

Eu, como usuário, desejo pode excluir as unidades de medida, a fim de manter apenas as unidades de medida que ainda são utilizadas.

Critérios de Aceitação

- Na listagem de unidades de medida, deve haver um botão para permitir a exclusão das mesmas;

- Após o usuário clicar em excluir, deve haver uma mensagem perguntando se o usuário tem certeza da ação.

Telas da Aplicação:

Botão de Excluir nas Unidades de Medida Existentes

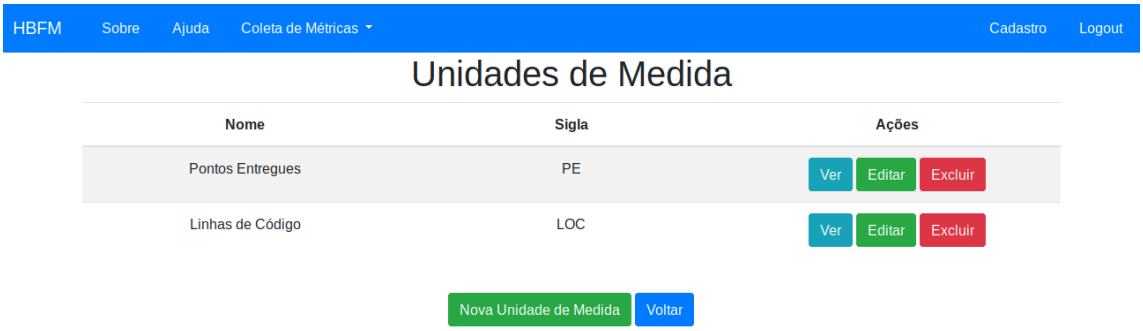

Figura 30 – Botão de Excluir nas Unidades de Medida Existentes

Fonte: Autora

5.5.8 US08: Excluir Métricas

Descrição da História:

Eu, como usuário, desejo pode excluir as métricas, a fim de manter apenas as métricas que ainda são utilizadas.

Critérios de Aceitação

- Na listagem de métricas, deve haver um botão para permitir a exclusão das mesmas;
- Após o usuário clicar em excluir, deve haver uma mensagem perguntando se o usuário tem certeza da ação.

Telas da Aplicação:

Botão de Excluir nas Métricas Existentes

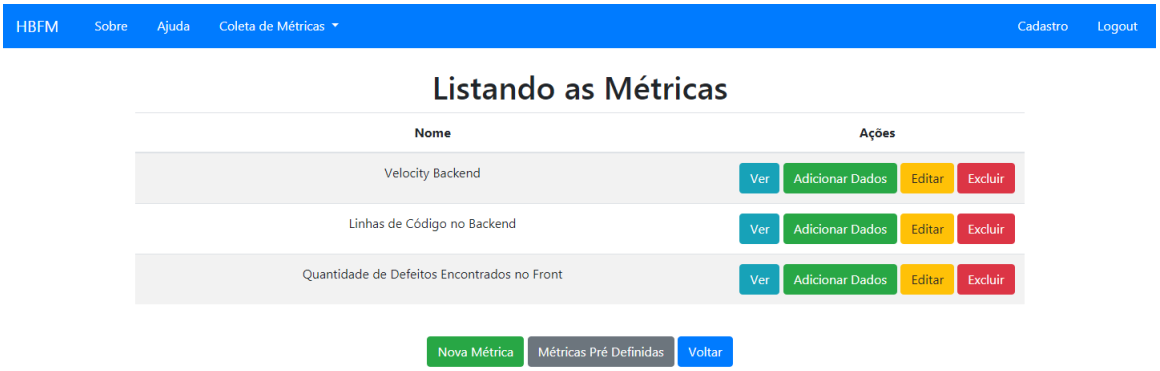

Figura 31 – Botão de Excluir nas Métricas Existentes

Fonte: Autora

5.5.9 US09: Importar Medidas

Definição da História:

Eu, como usuário, desejo poder importar medidas, a fim de poder migrar facilmente meus dados e automatizar o procedimento de adição de dados nas métricas.

Critérios de Aceitação:

- A importação deve ser feita por arquivos CSV;
- Deve haver um formulário em cada métrica que permite a importação do arquivo CSV;
- As medidas criadas por importação devem ter o mesmo comportamento das medidas criadas manualmente, assim deve ser possível visualizá-las nas métricas e editá-las;
- Deve haver um menu de ajuda descrevendo como deve ser feita a importação;

Telas da Aplicação

Formulário importação de medidas

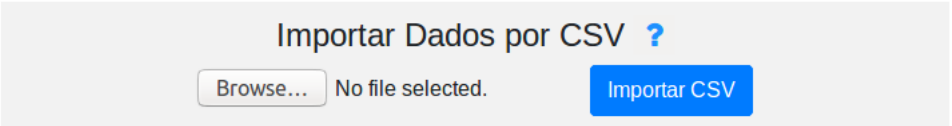

Figura 32 – Formulário de importação de medidas

Fonte: Autora

Mensagem de confirmação da importação

Figura 33 – Mensagem de confirmação da importação

Fonte: Autora

Informação de Ajuda

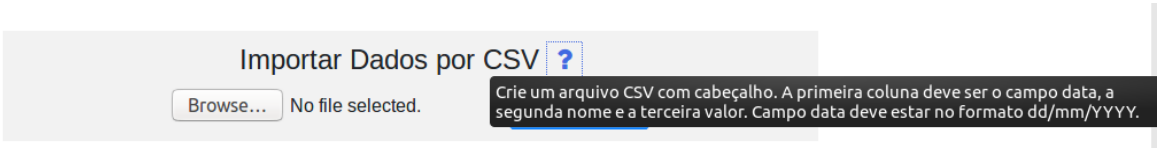

Figura 34 – Informação de Ajuda

Fonte: Autora

5.5.10 US10: Ajuda

Definição da História:

Eu, como usuário, desejo visualizar informações de ajuda, a fim de entender melhor o funcionamento do sistema.

Critérios de Aceitação:

- Deve haver um link de ajuda na barra superior;
- Deve haver ajuda sobre as principais; funcionalidades: cadastrar unidade de medida, cadastrar métrica, cadastrar medida.

Telas da Aplicação:

Link de Ajuda na Barra Superior

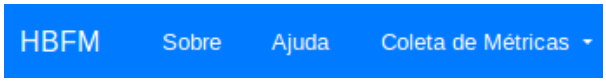

Figura 35 – Link de Ajuda na Barra Superior

Fonte: Autora

Tela de Ajuda

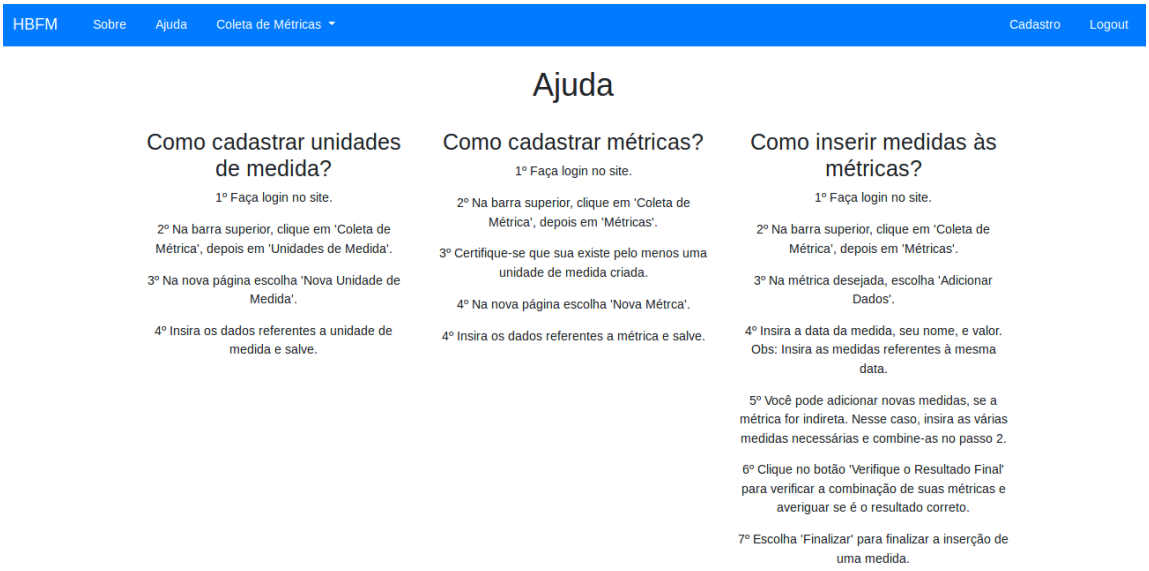

Figura 36 – Tela de Ajuda

Fonte: Autora

5.5.11 US11: Métricas Predefinidas

Definição da História:

Eu, como usuário, desejo poder selecionar métricas predefinidas, a fim de agilizar o processo de criação de métricas

Critérios de Aceitação:

- Deve haver uma opção na tela de métricas que leva às métricas predefinidas;
- Deve haver uma lista de métricas predefinidas que permita o usuário escolher alguma delas;
- Na lista de métricas, deve estar explicito o nome da métrica, sua escala de medição e sua unidade de medida;
- Após escolher a métrica, o usuário deve ser redirecionado para a página da métrica.

Telas da Aplicação:

Lista de Métricas Predefinidas

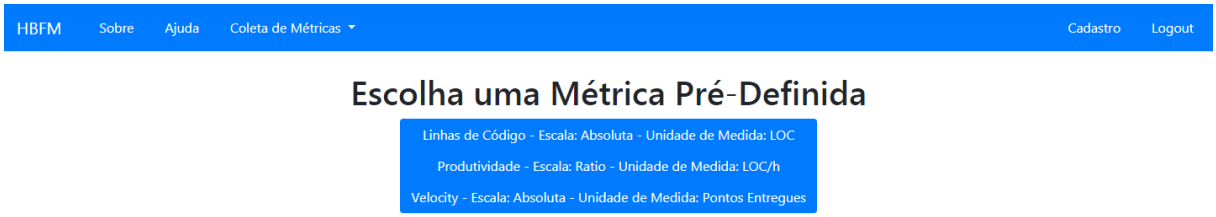

Figura 37 – Tela de Ajuda

Fonte: Autora

5.6 Validação do Sistema

Quando o desenvolvimento do sistema chegou a um ponto que ele era minimamente viável para o usuário, foi iniciada sua validação. A validação do sistema aconteceu de duas formas: por meio das técnicas de Interação Humano Computador (IHC) - Observação de Usuários e Perguntando aos Usuários, explicado na Seção 5.6.1; e por meio da aplicação da ferramenta em um ambiente real de desenvolvimento de software, explicado na Seção 5.6.2.

5.6.1 Iterações de Design

A primeira forma de validação para a ferramenta, foi feita aplicando técnicas de IHC. Essa validação tinha por objetivo validar o design da ferramenta e sua usabilidade para os usuários. As duas técnicas usadas foram Observação de Usuários e Perguntando aos Usuários. A técnica de Observação de Usuários consiste em observar e escutar aos usuários enquanto eles utilizam o software (PREECE; ROGERS; SHARP, 2001). A técnica Perguntando aos Usuários consiste em perguntar aos usuários por meio de entrevistas e questionários o que eles gostaram no sistema e o que eles não gostaram (PREECE; ROGERS; SHARP, 2001). Assim, após aplicar essas técnicas de IHC, as melhorias observadas e sugeridas seriam aplicadas ao sistema, gerando as Iterações de Design.

As técnicas foram aplicadas em ambientes controlados, fora do contexto real de aplicação da ferramenta. Foi pedido aos usuários realizarem determinada ação, com determinados dados. Também foi solicitado que o usuário tentasse falar durante a experimentação. Durante o experimento, anotações sobre as principais dificuldades do usuário foram feitas por parte do avaliador. Após o final do experimento, foi solicitado para o usuário dizer o que ele gostou na ferramenta, o que ele não gostou, e onde ele sentiu mais dificuldade. Cada uma das iterações foram feitas com pessoas diferentes, para não haver enviesamento nos resultados.

Primeira iteração de design:

Foi sugerido usuário cadastrar a métrica **Número de defeitos encontrados**. Foi dado nome da métrica, sua escala de medição, sua unidade de medida, assim como os valores coletados das medidas, com suas respectivas datas de coleta.

Dificuldades apresentadas:

- Após criar a métrica, clicou em editar para tentar adicionar os dados das medidas;
- Não soube identificar que deveria clicar em finalizar após inserir cada uma das medidas;
- Não clicou em 'Verificar Resultados'.

Bugs encontrados:

- Valor e nome das medidas não estão sendo mantidos, ao tentar salvar uma medida sem clicar em 'Verificar Resultados';
- Erro na visualização de uma métrica quando existia uma medida sem nome e valor.

Melhorias aplicadas:

- Adicionado botão de 'Adicionar Dados' na tela de visualização de métricas.

Segunda iteração de design:

Foi sugerido usuário cadastrar a métrica ***Velocity Backend***. Foi dado nome da métrica, sua escala de medição, sua unidade de medida, assim como os valores coletas das medidas, com suas respectivas datas de coleta.

Dificuldades apresentadas:

- Não soube identificar que deveria clicar em finalizar após inserir cada uma das medidas;
- Não notou o botão 'Verificar Resultados';
- Dificuldade em encontrar os botões de cadastro.

Bugs encontrados:

- Erro ao cadastrar medidas sem alterar seu valor inicial de operador;

Melhorias aplicadas:

- Data é a primeira informação a ser preenchida;
- Inserido uma dica informando que devem ser inseridas apenas medidas para a mesma data;
- Adicionado informação sobre os passos que devem ser tomados para cadastrar uma medida;
- Modificada a cor do botão de 'Verificar Resultados';
- Modificada a cor dos textos na *topbar*;

5.6.2 Validação em Ambiente Real de Desenvolvimento

A segunda forma de validação foi feita dentro de um ambiente real de desenvolvimento de software. Essa validação tinha por objetivo validar se a ferramenta atendia a necessidade da organização por manter sua base histórica de métricas. Assim, dados reais de desenvolvimento seriam coletados e inseridos na ferramenta de base histórica.

A coleta foi feita em uma startup de Brasília, que desenvolve ponto de venda para micro e pequenos empreendedores. A empresa possui 8 desenvolvedores e outra pequena equipe voltada para vendas e marketing. A equipe de desenvolvimento coleta poucas métricas, mas são principalmente métricas de produto. Exemplos das métricas já coletadas na empresa são: Cobertura de Teste, Estilo de Código, Maus cheiros de código: duplicação de código, complexidade ciclomática, etc. As métricas de produto, não são mantidas historicamente. A única métrica de processo existente era o *velocity* da equipe, que era mantida em um cartaz no escritório.

Foi perguntando ao chefe de tecnologia da empresa, se havia o interesse de manter o histórico de métricas e até mesmo de coletar novas métricas na empresa. A resposta foi positiva, e então, foi solicitada a permissão para fazer a validação da ferramenta dentro da empresa, também obtendo resposta positiva.

Assim, foram decididas que seriam coletadas 3 métricas. A primeira métrica coletada foi o ***Velocity Backend***, que corresponde a quantidade de pontos entregues pela equipe de desenvolvimento do *backend* do software em uma *sprint*. Essa métrica é à uma métrica de processo, e é utilizada para auxiliar nas decisões de quantos pontos serão planejados para as demais *sprints*. A coleta dessa métrica utilizou dados do mês de março, e era coletada toda sexta-feira, quando a *sprint* é finalizada, e então, é possível saber quantos pontos foram entregues. A Figura 38 apresenta o resultado final da coleta da métrica **Velocity Backend**.

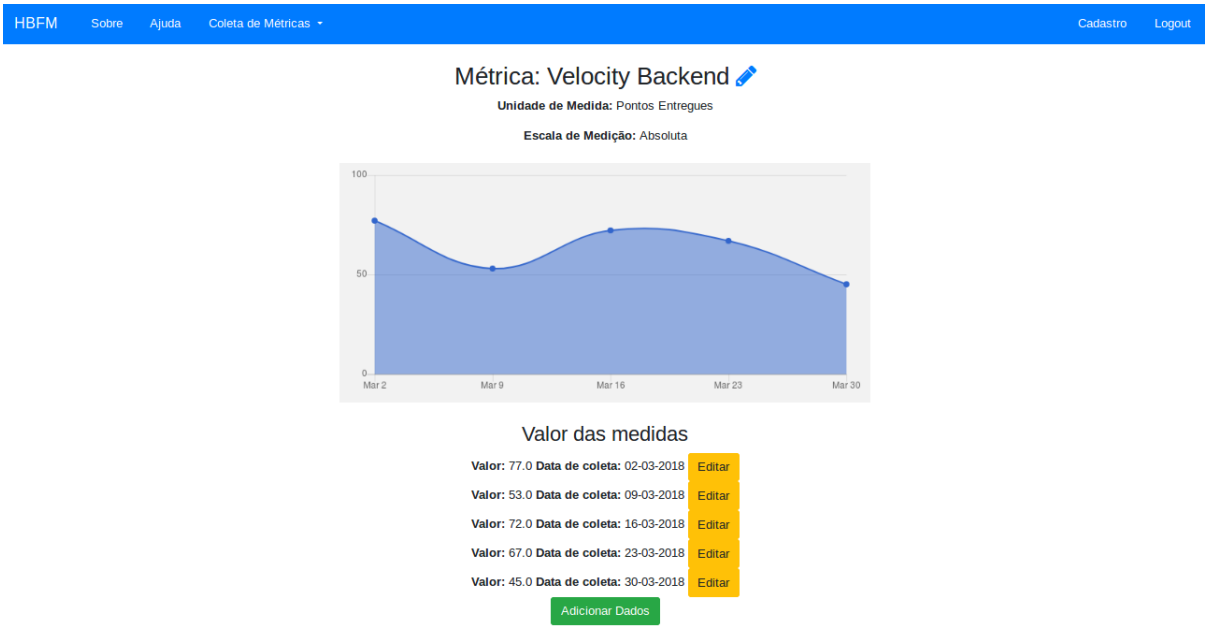

Figura 38 – Métrica *Velocity Backend*

Fonte: Autora

A segunda métrica coletada foi **Linhas de Código do *Backend***, que corresponde a quantidade de linhas de código total no projeto do *backend*, incluindo linhas de código comentadas. Essa métrica é uma métrica de produto e pode ser utilizada para acompanhar o aumento no tamanho do software. A coleta dessa métrica utilizou dados do mês de final do abril e dados das 3 primeiras semanas de maio, e era coletada toda sexta-feira, ao final de uma *sprint*. A Figura 39 apresenta o resultado final da coleta da métrica **Linhas de Código do *Backend***.

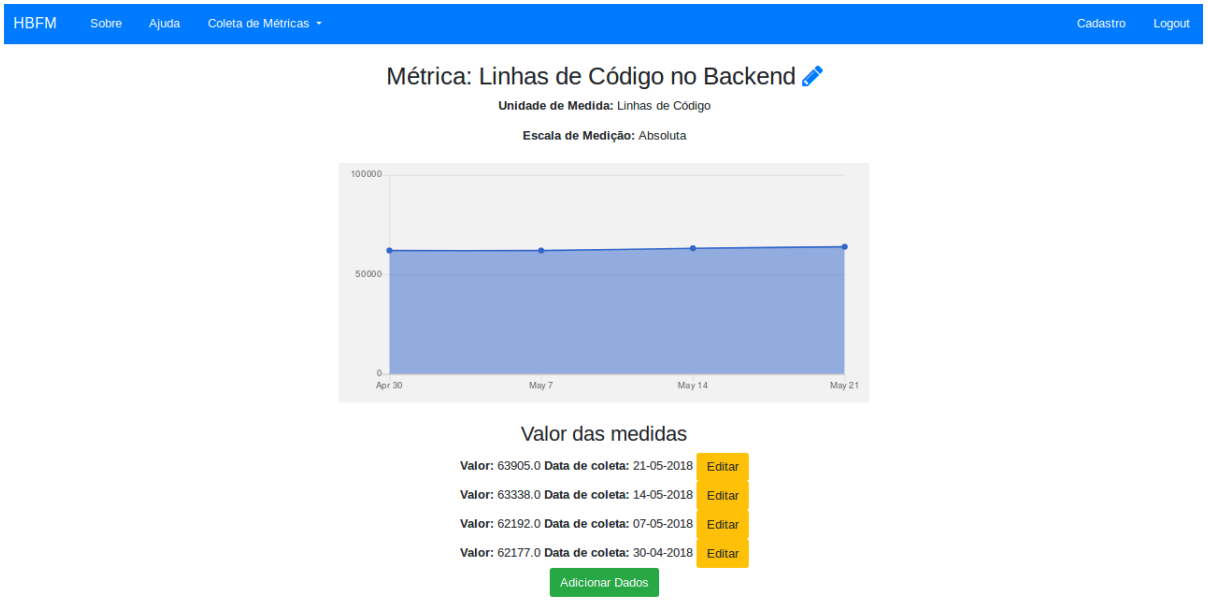

Figura 39 – Linhas de Código do *Backend*

Fonte: Autora

A terceira métrica coletada foi **Quantidade de Defeitos Encontrados no *Frontend***, que corresponde a todos os erros encontrados tanto em fase de teste, quanto em fase de produção, no *frontend* do projeto. Essa métrica é uma métrica de processo e pode ser utilizada para acompanhar a quantidade de defeitos encontrados ao longo do projeto. A coleta dessa métrica utilizou dados do mês de maio, e era coletada toda sexta-feira, ao final de uma *sprint*. A Figura 40 apresenta o resultado final da coleta da métrica **Quantidade de Defeitos Encontrados no *Frontend***

Figura 40 – Quantidade de Defeitos Encontrados no *Frontend*

Fonte: Autora

Após a coleta, os resultados foram apresentados para a equipe de desenvolvimento. A equipe mostrou-se satisfeita com a coleta de dados e com a ferramenta. Foi sugerido inserir um filtro por data, para melhorar a visualização das métricas quando houverem muitos dados históricos. Acredita-se que essa ferramenta, juntamente com um plano de métricas bem estruturado, vai ajudar a empresa a conhecer melhor os problemas e desenvolver boas soluções para eles.

Além disso, a equipe aceitou manter o uso a ferramenta para continuar coletando os dados históricos e novas métricas serão coletadas na empresa. A ferramenta será evoluída dentro da própria empresa e também será aberta para contribuições por meio do GitHub.

5.7 Integração e *Deploy*

Conforme mencionado na Seção 5.3.2, quando cada uma das funcionalidades estava pronta, seu *branch* era juntado ao *branch* master do projeto. Assim, versões estáveis do código eram mantidas.

No momento em que a ferramenta chegou em um estado que estava minimamente viável para o usuário final, foi iniciado o procedimento de colocar a ferramenta em produção, processo chamado *deployment*. O *deploy* foi feito utilizando a plataforma Heroku[4]. A plataforma Heroku é um sistema de contêineres em nuvem, que permite realizar *deploys* de manteira fácil e rápida.

Para realizar o *deploy* da ferramenta, foi necessário ajustar o banco de dados de produção para o *Postgress SQL*, adicionar um repositório remoto do Heroku por meio da ferramenta Git e então realizar enviar os dados para esse repositório. Após esse procedimento, o próprio Heroku gerou uma url aleatória para o sistema, então foi preciso apenas renomeá-la para o nome desejado. Atualmente, o sistema pode ser acessado pela url: <https://hbfm.herokuapp.com>.

Uma vez que a configuração do *deploy* estava pronta, sempre que uma nova versão estável do código estava pronta, essa versão era imediatamente enviada para o servidor de produção. Assim, as melhorias e correções de erros seriam rapidamente enviadas aos usuários. Isso só foi possível devido à facilidade de realizar *deploys* com o Heroku.

5.8 Comparação com ferramentas do estudo

A primeira parte desse estudo fez um estudo das ferramentas de medições encontradas na literatura. Foram listada diversas funcionalidades que as ferramentas apresentam, além de vantagens e desvantagens.

Desta forma, é interessante comparar a ferramenta desenvolvida com as demais ferramentas estudadas, por meio da análise de quais funcionalidades a ferramenta desenvolvida entregue, e quais são as vantagens e desvantagens.

As principais funcionalidades apresentadas pelas ferramentas foram: **Gráficos (G)**, **Integração (I)**, **Múltiplas Linguagens de Programação (M)**, **Definição de Métricas (D)**, **Relatórios (R)**, **Exportação de Dados (E)**, **Importação de Dados (Im)** e **Histórico**. A Tabela 8 apresenta as funcionalidades que a ferramenta desenvolvida contempla.

Ferramenta	**G**	**I**	**M**	**D**	**R**	**E**	**IM**	**H**
hbfm	x		x	x			x	x

Tabela 8 – Funcionalidade contemplada pela ferramenta desenvolvida

A ferramenta desenvolvida gera 2 tipos de gráficos, de acordo com a escala da métrica. Atende múltiplas linguagens, uma vez que é possível fazer o registro de métricas de qualquer linguagem. É possível fazer a definição de métricas, com qualquer escala de

[4] Disponível em: http://heroku.com/

medição e qualquer unidade de medida desejada pelo usuário. A importação de dados é feita por meio de CSV dentro da ferramenta. E também existe a manutenção histórica dos dados, que é o foco principal da ferramenta.

As demais funcionalidades ainda não existem na ferramenta, mas pretendem ser desenvolvidas em versões futuras da ferramenta, conforme descrito na Seção 5.9

As vantagens que foram apresentadas foram:

V1: Utilizar técnicas que aderem a padrões de qualidade do processo de medição.

V2: Preocupar-se com a experiência do usuário.

As desvantagens apresentadas foram:

D1: Ser difícil de utilizar do ponto de vista do usuário.

D2: Não fornecer meios de automatizar a coleta.

Em relação às vantagens, a ferramenta desenvolvida contempla a V2: Preocupar-se com a experiência do usuário, visto que possui interface gráfica e tutoriais de como utilizar a ferramenta. A V1 ainda não está contemplada na versão atual da ferramenta, mas planeja ser inserida em uma versão futura, conforme explicado na Seção 5.9.

Em relação às desvantagens, a ferramenta não apresenta nenhuma delas. A ferramenta possui interface gráfica, não apresentando, assim, a D1. A automatização da coleta de dados, atualmente, é feita por importação de dados CSV, eliminando a D2.

O principal diferencial da ferramenta está nas funcionalidades que ela oferece. Como mostrado na Seção 4.2, as funcionalidades menos contempladas são Integração, Definição de Métricas, Importação de Dados e Histórico. Dessas quatro funcionalidades, a ferramenta desenvolvida apresenta três.

5.9 Limitações e Funcionalidades futuras

A ferramenta desenvolvida já apresenta diversas funcionalidades úteis para equipes de software coletarem métricas e analisarem dados historicamente. Porém, ainda existem algumas limitações na ferramenta que impede seu total potencial de base histórica. As principais limitações são:

- É possível alterar a unidade de medida de uma métrica. Não faz sentido, uma vez que os dados ficarão incorretos. Por exemplo, se uma métrica medida em linhas de código/hora for alterada para pontos de função/hora, o valor antigo da métrica estará na unidade de medida errada.
- Não existe o conceito de perfis e projeto na ferramenta. Assim, apenas um usuário tem acesso aos dados históricos.

- Não existe filtro por data nas métricas. É uma funcionalidade importante em uma base histórica, para permitir visualizar os dados de um período específico.

Além disso, já existem várias funcionalidades que estão planejadas para serem adicionadas à ferramenta. Após a conclusão deste trabalho, como já mencionado anteriormente, a ferramenta estará aberta para contribuições e também será evoluída dentro da empresa onde a validação foi feita.

As principais funcionalidades planejadas para serem desenvolvidas são:

- Integração com alguma ferramenta de análise estática de código;
- Comparação entre duas métricas com a mesma unidade de medida;
- Exportar os dados das métricas;
- Permitir ao usuário escolher a abordagem de medição de software (GQM, CMMI ou PSM).

Ao desenvolver essas funcionalidades, todas as principais funcionalidades apresentadas pelas ferramentas serão contempladas. Além disso, também será abordado a vantagem da ferramenta de utilizar técnicas que aderem a padrões de qualidade do processo de medição. Todas essas novas funcionalidades já estão cadastradas no repositório do GitHub. Porém, precisam ser melhor definidas para iniciar o desenvolvimento delas.

6 Considerações Finais

O uso de métricas auxiliam as organizações de software à fazerem previsões confiáveis sobre o produto de software que está sendo desenvolvido. Para facilitar o processo de medição, o uso de ferramentas é encorajado.

Com estudo das ferramentas apresentadas pela literatura, foi possível responder às questões de pesquisa levantadas. As ferramentas mostraram-se restritivas em relação aos tipos de métricas que elas coletam e armazenam. Muitas delas utilizam um conjunto base de métricas pequeno e não permitem a adição de novas métricas para aumentar esse conjunto. Notou-se também que muitas ferramentas utilizam as mesmas métricas, podendo notar um padrão nas métricas mais utilizadas.

A revisão sistemática auxiliou a entender o estado atual das ferramentas de métricas e medições, desta forma, foi possível identificar as lacunas que essas ferramentas apresentam e quais são seus pontos fortes. Assim, foi possível desenvolver uma ferramenta que preencheu essas lacunas e que utilizou os melhores recursos propostos por elas.

O objetivo geral do trabalho foi atingido, uma vez que a ferramenta de base histórica de medição flexível foi criada e validada. A criação da ferramenta seguiu um processo de desenvolvimento de software e foram utilizadas boas práticas de projeto e de desenvolvimento. A validação da ferramenta ocorreu em duas formas. Primeiramente fora do contexto real de desenvolvimento, para validar o design da ferramenta. Depois, dentro de um contexto real para validar se as funcionalidades da ferramenta atendiam a necessidade da organização de manter seus dados históricos.

Como sugestão de trabalhos futuros, há possibilidade de continuar o desenvolvimento de novas funcionalidades na ferramenta, por meio da contribuição no repositório do GitHub. Além disso, é interessante aplicar a ferramenta em novos contextos reais de desenvolvimento de software para compreender melhor quais fraquezas existem na ferramenta e propor novas melhorias.

Referências

AVERSANO, L.; GRASSO, C.; GRASSO, P.; TORTORELLA, M. Investigating differences and commonalities of software metric tools. p. 249–256, 01 2017. Citado 6 vezes nas páginas 38, 40, 41, 42, 43 e 44.

BAKAR, N. A. A.; BOUGHTON, C. V. et al. Using a combination of measurement tools to extract metrics from open source projects. In: **Proceeding (632) Software Engineering and Applications-2008**. [S.l.]: ACTA Press, 2008. Citado 4 vezes nas páginas 36, 41, 42 e 43.

BAKAR, N. S. A. A.; BOUGHTON, C. V. Validation of measurement tools to extract metrics from open source projects. In: IEEE. **Open Systems (ICOS), 2012 IEEE Conference on**. [S.l.], 2012. p. 1–6. Citado 4 vezes nas páginas 37, 41, 42 e 43.

BUDIMAC, Z.; RAKIC, G.; HERICKO, M.; GERLEC, C. Towards the better software metrics tool. In: IEEE. **Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on**. [S.l.], 2012. p. 491–494. Citado 2 vezes nas páginas 37 e 43.

CHANDRA, A. K.; MERLIN, P. M. Optimal implementation of conjunctive queries in relational data bases. In: **Proceedings of the Ninth Annual ACM Symposium on Theory of Computing**. New York, NY, USA: ACM, 1977. (STOC '77), p. 77–90. Disponível em: <http://doi.acm.org/10.1145/800105.803397>. Citado na página 57.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. **CMMI for Development: Guidelines for Process Integration and Product Improvement**. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2011. ISBN 0321711505, 9780321711502. Citado na página 25.

DA-WEI, E. Analysis and implementation of software metric for object-oriented. In: IEEE. **Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on**. [S.l.], 2009. p. 1–4. Citado 2 vezes nas páginas 36 e 43.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and software technology**, Elsevier, v. 50, n. 9-10, p. 833–859, 2008. Citado na página 38.

FENTON, N. Software measurement: A necessary scientific basis. **IEEE Transactions on software engineering**, IEEE, v. 20, n. 3, p. 199–206, 1994. Citado 2 vezes nas páginas 21 e 23.

FENTON, N.; MELTON, A. Deriving structurally based software measures. **Journal of Systems and Software**, Elsevier, v. 12, n. 3, p. 177–187, 1990. Citado 2 vezes nas páginas 50 e 51.

FENTON, N.; PFLEEGER, S. L. **Software Metrics (2Nd Ed.): A Rigorous and Practical Approach**. Boston, MA, USA: PWS Publishing Co., 1997. Citado 12 vezes nas páginas 16, 17, 20, 21, 22, 23, 24, 46, 47, 50, 51 e 52.

FINKELSTEIN, L.; LEANING, M. A review of the fundamental concepts of measurement. **Measurement**, Elsevier, v. 2, n. 1, p. 25–34, 1984. Citado na página 20.

FLORAC, W. A.; PARK, R. E.; CARLETON, A. D. **Practical Software Measurement: Measuring for Process Management and Improvement**. Pittsburgh, PA 15213: Carnegie Mellon University, 1997. Citado 3 vezes nas páginas 16, 17 e 26.

FONSECA, V. S.; BARCELLOS, M. P.; FALBO, R. de A. Integration of software measurement supporting tools: A mapping study. In: **SEKE**. [S.l.: s.n.], 2015. p. 516–521. Citado 4 vezes nas páginas 37, 41, 42 e 43.

_____. An ontology-based approach for integrating tools supporting the software measurement process. **Science of Computer Programming**, Elsevier, v. 135, p. 20–44, 2017. Citado 4 vezes nas páginas 38, 41, 42 e 43.

GARCIA, F. et al. Managing software process measurement: A metamodel-based approach. **Information Sciences**, Elsevier, v. 177, n. 12, p. 2570–2586, 2007. Citado 2 vezes nas páginas 36 e 42.

GARDARIN, G.; VALDURIEZ, P. **Relational databases and knowledge bases**. [S.l.], 1989. Citado na página 57.

HIGO, Y. et al. A pluggable tool for measuring software metrics from source code. In: IEEE. **Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)**. [S.l.], 2011. p. 3–12. Citado 4 vezes nas páginas 36, 42, 51 e 52.

HUSEIN, S.; OXLEY, A. A coupling and cohesion metrics suite for object-oriented software. In: IEEE. **Computer Technology and Development, 2009. ICCTD'09. International Conference on**. [S.l.], 2009. v. 1, p. 421–425. Citado 2 vezes nas páginas 36 e 40.

ISO/IEC/IEEE. **ISO/IEC/IEEE Systems and Software Engineering - System Life Cycle Processes**. [S.l.], 2008. c1-84 p. Citado 2 vezes nas páginas 16 e 20.

_____. **Systems and software engineering - Measurement process**. [S.l.], 2017. Citado 3 vezes nas páginas 22, 23 e 24.

KAN, S. H. **Metrics and Models in Software Quality Engineering**. 2nd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201729156. Citado 2 vezes nas páginas 16 e 17.

KESER, B.; IYIDOGAN, T.; OZKAN, B. Assist: an integrated measurement tool. In: IEEE. **Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on**. [S.l.], 2013. p. 237–242. Citado 2 vezes nas páginas 37 e 40.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. **ech. rep. EBSE 2007-001**, Keele University and Durham University, 2007. Citado 4 vezes nas páginas 19, 27, 29 e 38.

KRASNER, G. E.; POPE, S. T. et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. **Journal of object oriented programming**, v. 1, n. 3, p. 26–49, 1988. Citado na página 56.

KUNZ, M.; DUMKE, R. R.; SCHMIETENDORF, A. How to measure agile software development. In: **Software Process and Product Measurement**. [S.l.]: Springer, 2008. p. 95–101. Citado 2 vezes nas páginas 36 e 44.

KUNZ, M.; DUMKE, R. R.; ZENKER, N. Software metrics for agile software development. In: IEEE. **Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on**. [S.l.], 2008. p. 673–678. Citado 2 vezes nas páginas 36 e 44.

KUNZ, M.; ZENKER, N.; MENCKE, S.; DUMKE, R. R. Unit metrics-a tool to support refactoring in agile software development. In: **Software Engineering Research and Practice**. [S.l.: s.n.], 2008. p. 389–395. Citado 2 vezes nas páginas 36 e 44.

LAPOLLI, F. R.; CRUZ, C. M.; MOTTA, C. L. R.; TOLLA, C. E. Modelo de desenvolvimento de objetos de aprendiagem baseado em metodologias ágeis e scaffoldings. **Brazilian Journal of Computers in Education**, v. 18, n. 02, p. 17, 2010. Citado na página 53.

LEAVITT, N. Will nosql databases live up to their promise? **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 43, n. 2, p. 12–14, fev. 2010. ISSN 0018-9162. Disponível em: <https://doi.org/10.1109/MC.2010.58>. Citado na página 57.

LINOS, P.; LUCAS, W.; MYERS, S.; MAIER, E. A metrics tool for multi-language software. In: ACTA PRESS. **Proceedings of the 11th IASTED International Conference on Software Engineering and Applications**. [S.l.], 2007. p. 324–329. Citado 2 vezes nas páginas 35 e 43.

MANEVA, N.; GROZEV, N.; LILOV, D. A framework for source code metrics. In: ACM. **Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies**. [S.l.], 2010. p. 113–118. Citado 2 vezes nas páginas 36 e 42.

MCQUILLAN, J.; POWER, J. A metamodel for the measurement of object-oriented systems: An analysis using alloy. In: IEEE. **Software Testing, Verification, and Validation, 2008 1st International Conference on**. [S.l.], 2008. p. 288–297. Citado 2 vezes nas páginas 36 e 42.

MONDEN, A. et al. Customizing gqm models for software project monitoring. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 95, n. 9, p. 2169–2182, 2012. Citado 2 vezes nas páginas 37 e 41.

NÚÑEZ-VARELA, A.; PEREZ-GONZALEZ, H. G.; CUEVAS-TELLO, J. C.; SOUBERVIELLE-MONTALVO, C. A methodology for obtaining universal software code metrics. **Procedia Technology**, Elsevier, v. 7, p. 336–343, 2013. Citado 4 vezes nas páginas 37, 41, 42 e 43.

NÚÑEZ-VARELA, A. S.; PÉREZ-GONZÁLEZ, H. G.; MARTÍNEZ-PÉREZ, F. E.; CUEVAS-TELLO, J. Building a user oriented application for generic source code metrics extraction from a metrics framework. In: IEEE. **Software Engineering Research**

and Innovation (CONISOFT), 2016 4th International Conference in. [S.l.], 2016. p. 27–32. Citado 2 vezes nas páginas 38 e 44.

NUÑEZ-VARELA, A. S.; PÉREZ-GONZALEZ, H. G.; MARTÍNEZ-PEREZ, F. E.; SOUBERVIELLE-MONTALVO, C. Source code metrics: A systematic mapping study. **Journal of Systems and Software**, Elsevier, v. 128, p. 164–197, 2017. Citado 6 vezes nas páginas 38, 40, 41, 42, 43 e 44.

PREECE, J.; ROGERS, Y.; SHARP, H. **Beyond Interaction Design: Beyond Human-Computer Interaction**. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 0471402494. Citado na página 72.

RAKIĆ, G.; BUDIMAC, Z. Smiile prototype. In: AIP. **AIP Conference Proceedings**. [S.l.], 2011. v. 1389, n. 1, p. 853–856. Citado 2 vezes nas páginas 37 e 43.

RAKIĆ, G.; GERLEC, Č.; NOVAK, J.; BUDIMAC, Z. Xml-based integration of the smiile tool prototype and software metrics repository. In: AIP. **AIP Conference Proceedings**. [S.l.], 2011. v. 1389, n. 1, p. 869–872. Citado 2 vezes nas páginas 37 e 43.

RAKIć, G.; BUDIMAC, Z.; BOTHE, K. Towards a 'universal' software metrics tool: Motivation, process and a prototype. In: . [s.n.], 2010. v. 2, p. 263–266. Cited By 1. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-78751551616&partnerID=40&md5=19d63245fc6b44179f32a853e4f4995f>. Citado 2 vezes nas páginas 36 e 43.

RAMARAJ, E.; DURAISAMY, S. Design optimization metrics for uml based object-oriented systems. **International Journal of Software Engineering and Knowledge Engineering**, World Scientific, v. 17, n. 03, p. 423–448, 2007. Citado 2 vezes nas páginas 36 e 41.

REES, M. J. A feasible user story tool for agile software development? In: **Proceedings of the Ninth Asia-Pacific Software Engineering Conference**. Washington, DC, USA: IEEE Computer Society, 2002. (APSEC '02), p. 22–. ISBN 0-7695-1850-8. Disponível em: <http://dl.acm.org/citation.cfm?id=785409.785814>. Citado na página 58.

SANTHOSHINI, G.; ANBAZHAGAN, K. An object based software tool for software measurement. In: IEEE. **Information Communication and Embedded Systems (ICICES), 2014 International Conference on**. [S.l.], 2014. p. 1–5. Citado 2 vezes nas páginas 37 e 43.

SHIN, Y.; MENEELY, A.; WILLIAMS, L.; OSBORNE, J. A. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. **IEEE Transactions on Software Engineering**, IEEE, v. 37, n. 6, p. 772–787, 2011. Citado na página 22.

SOLINGEN, R. V.; BERGHOUT, E. **The Goal/Question/Metric Method: a practical guide for quality improvement of software development**. [S.l.]: McGraw-Hill, 1999. Citado 4 vezes nas páginas 20, 21, 24 e 25.

TAHIR, T.; RASOOL, G.; GENCEL, C. A systematic literature review on software measurement programs. **Information and Software Technology**, Elsevier, v. 73, p. 101–121, 2016. Citado 3 vezes nas páginas 37, 42 e 43.

TK. **Ruby on Rails: HTTP, MVC and Routes**. 2017. Disponível em: <https://medium.com/the-renaissance-developer/ruby-on-rails-http-mvc-and-routes-f02215a46a84>. Citado na página 56.

UMAMAHESWARI, E.; NATARAJAN, B.; GHOSH, D. Evaluating metrics at class and method level for java programs using knowledge based systems. v. 10, p. 2047–2052, 01 2015. Citado 4 vezes nas páginas 37, 42, 50 e 51.