INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Adoção de métricas ágeis integradas com a plataforma GITHUB

Bruna Amorim dos Santos

Mestrado em Gestão de Sistemas de Informação

Orientadora:
Professora Doutora Adriana Lopes Fernandes, Professora Auxiliar, ISCTE-IUL

Setembro, 2022

TECNOLOGIAS
E ARQUITETURA

Departamento de Ciência e Tecnologias da Informação

Adoção de métricas ágeis integradas com a plataforma GITHUB

Bruna Amorim dos Santos

Mestrado em Gestão de Sistemas de Informação

Orientadora:
Professora Doutora Adriana Lopes Fernandes, Professora Auxiliar, ISCTE-IUL

Setembro, 2022

Agradecimentos

Primeiramente quero agradecer à Deus que sempre me abençoou.

Aos meus pais Genuir e Olímpia por todo amor e dedicação que sempre tiveram, e por todo esforço que fizeram para nos dar a melhor educação e ensinamentos. À minha irmã Haline por todo carinho e amizade ao longo de nossas vidas.

Aos meus amigos do coração Leandro Mendes e Tatiana Viana pela ajuda nas minhas dificuldades e apoio no que eu precisei neste trabalho.

À minha orientadora Adriana Lopes Fernandes por toda orientação e disponibilidade para esclarecer minhas dúvidas.

À todas as pessoas que participaram desse trabalho, aos meus colegas e professores que compartilharam seus conhecimentos ao longo do mestrado.

Resumo

Devido à crescente demanda e complexidade dos *softwares*, as empresas adotam, cada vez mais, a abordagem ágil no seu processo de desenvolvimento para aumentar a produtividade, qualidade e responder rapidamente às mudanças e imprevistos do mercado. Diante dessas necessidades, faz-se necessária a utilização de ferramentas que auxiliem os gestores e as equipas de projetos a monitorizar o processo de construção de *software* por meio da visualização de métricas. No entanto, muitas dessas ferramentas não exibem informações confiáveis, significativas e atualizadas, o que prejudica a análise e tomada de decisões. Face a este cenário, o objetivo deste trabalho é desenvolver uma ferramenta *web* para a recolha e visualização de métricas ágeis para equipas de desenvolvimento de *software* que utilizam o GitHub para armazenamento de código e gestão de tarefas. Para que o objetivo fosse alcançado, iniciou-se o estudo com o levantamento dos requisitos necessários para a modelação da ferramenta, e posteriormente a definição da arquitetura geral. Após o desenvolvimento da ferramenta, a mesma foi testada e avaliada por utilizadores considerados relevantes no contexto ágil. Os resultados mostram que a ferramenta desenvolvida foi considerada simples, segura, com gráficos de fácil análise, apresenta informações confiáveis, com controle de acesso aos dados dos projetos dos utilizadores, útil para as equipas ágeis, capaz de fornecer métricas fundamentais e auxiliar as equipas a melhorar o seu produto e, principalmente, os seus processos de trabalho, proporcionando a melhoria contínua.

Palavras-Chave: Métricas Ágeis, Metodologia Ágil, GitHub, Melhoria Contínua

Abstract

Due to the growing demand and complexity of software, companies are increasingly adopting an agile approach to their development process in order to increase productivity and quality and to quickly respond to market changes and unforeseen events. Therefore, it is necessary the use of tools that help project managers and teams to monitor the software construction process through the visualization of metrics. However, many of these tools do not display reliable, meaningful and up-to-date information, which hinders analysis and decision making. Given this scenario, the present study aims to develop a web tool for the collection and visualization of agile metrics for software development teams that use GitHub for code storage and task management. The study began with the identification of the necessary requirements for modeling the tool, and subsequent definition of the general architecture. After being developed, the tool was tested and evaluated by users considered relevant in the agile context. The results showed that the tool was considered simple, secure, with easy analysis charts, with reliable information, with controlled access to user project data, useful for agile teams, capable of providing fundamental metrics and helping teams to improve their product and their work processes, providing continuous improvement.

Keywords: Agile Metrics, Agile Methodology, GitHub, Continuous Improvement

Índice

Índice de Tabelas

Índice de Figuras

Glossário de Abreviaturas e Siglas

API – Application Programming Interface

BI - *Business Intelligence*

CFD – Cumulative Flow Diagram

CPU – Central Processing Unit

GQM – *Goal-Question-Metric Paradigm*

IDE – *Integrated Device Eletronics*

JSON – Notação de Objetos *JavaScript*

LT – *Lead Time*

NPS – *Net Promoter Score*

NPV – *Net Present Value*

OKR – *Objectives and Key Results*

OSS – *Open Source Software*

RAM – *Random Access Memory*

REST – *Representational State Transfer*

RF – Requisitos Funcionais

RFN – Requisitos Não Funcionais

ROA – *Retorn on Assets*

ROI – *Return on Investiment*

SI – Sistema de Informação

UML - *Unified Modeling Language*

URL – *Uniform Resource Locator*

VSM – *Value Stream Maps*

XP – eXtreme Programming

WIP – *Work in Progress*

Capítulo 1 – Introdução

1.1. Enquadramento do tema

Os sistemas de informação (SI) têm uma importância vital para a sociedade e fazem parte de quase todas as atividades do dia-a-dia, estando presentes nas casas, escolas, hospitais, supermercados, entre outros (Singh, 2007).

Os SI são um modelo manual ou automatizado de processos responsáveis por coletar, processar, armazenar e transmitir informações que atendam as necessidades e ajudam a solucionar problemas das empresas, organizações e utilizadores (Singh, 2007).

Para responder à crescente necessidade e complexidade dos sistemas de informação, exigências dos utilizadores e à rápida mudança do mercado, a área da engenharia de *software* precisou de evoluir significativamente, recorrendo a novas teorias, abordagens e métodos (Fernandes & Machado, 2017).

As metodologias de desenvolvimento de *software,* como a abordagem ágil, possuem um conjunto de atividades que auxiliam a produção do mesmo. Com o surgimento dessa abordagem, alterou-se a metodologia de desenvolvimento, dando origem a uma nova forma de pensar e a uma mudança cultural nas empresas e nas pessoas (Galup et al., 2020). As empresas adotaram esta abordagem com o objetivo de aumentar a velocidade de lançamento de novos produtos ou negócios, aumentar a qualidade, a produtividade, responder mais rapidamente às mudanças e imprevistos, melhorar a integração entre pessoas, a gestão e o acompanhamento de projetos e tarefas (Venkatesh et al., 2020).

Para analisar e monitorizar esses objetivos e o processo de construção de *software* como um todo, é vantajoso utilizar métricas ágeis, pois essas fornecem dados reais sobre o estado de desenvolvimento das tarefas, qualidade do produto, produtividade da equipa, trazendo visibilidade sobre a saúde do processo de desenvolvimento da empresa. As métricas ágeis permitem projetar cenários de prazos de entrega e tomar decisões mais assertivas, bem como promovem ações de melhoria contínua para equipas ágeis de desenvolvimento de *software* (Albino, 2017).

Existem vários tipos de métricas ágeis que podem ser utilizadas, mas antes de as escolher, é necessário compreender os objetivos pretendidos, de forma a gerar os benefícios esperados e não afetar o desempenho do processo e/ou do produto (Hazzan & Dubinsky, 2007).

Atualmente, existem uma série de ferramentas que auxiliam os gestores e as equipas de projetos a monitorizar o trabalho por meio de métricas. No entanto, muitas dessas informações estão dispersas em diferentes fontes, causando dificuldades na consolidação da informação e, principalmente, na tomada de decisão. Assim sendo, torna-se relevante compreender a possibilidade de elaborar uma ferramenta que forneça as métricas necessárias para a gestão de equipas ágeis, utilizando uma das plataformas mais usadas por desenvolvedores de *software*, o GitHub.

1.2. Motivação e relevância do tema

Com a adoção de métodos ágeis pelas empresas para aumentar a sua produtividade, melhorar a qualidade, corresponder melhor às necessidades dos utilizadores e diminuir os custos dos *softwares*, as métricas passaram a ser necessárias para planear e acompanhar os projetos (Budacu & Pocatilu, 2018). As empresas precisam de definir objetivos de medição e adequar as métricas aos mesmos, permitindo refiná-las e adaptá-las sempre que necessário. A medição deve ser o mais simples possível para ser em tempo real e de fácil interpretação por todos os envolvidos no projeto (Hazzan & Dubinsky, 2007).

As métricas ajudam os profissionais, como por exemplo, os desenvolvedores, a analisar o processo de construção de *software*, tarefas, performance e segurança. Ajudam também o gerente de projetos a projetar cenários de entrega, compreender problemas na equipa, tomar determinadas ações estratégicas, avaliar a motivação das pessoas, entre outros (Albino, 2017).

Atualmente, existem no mercado ferramentas para a gestão de tarefas e de projetos que auxiliam equipas e gestores. Contudo, muitos ainda possuem dificuldades em acompanhar a situação e o progresso, realizar estimativas, acompanhar o esforço, qualidade e tempo de entrega. Essas dificuldades surgem, frequentemente, pelo facto de as ferramentas não possuírem as informações que são relevantes e/ou estarem dispersas em diferentes ferramentas, serem necessárias recolhas manuais e pela falta de definição e de entendimento das métricas. Assim, a motivação para a realização desta dissertação é auxiliar equipas e gestores a compreender a importância de definir as métricas necessárias para avaliar os seus problemas e objetivos, otimizar a recolha e organização dessas informações, centralizar os dados recolhidos e facilitar a análise e a visualização dos resultados para proporcionar decisões assertivas e a melhoria contínua.

1.3. Objetivos de investigação

O presente trabalho tem como objetivo geral elaborar uma ferramenta para visualização de métricas, relevantes para equipas de desenvolvimento de *software* que trabalham segundo a abordagem ágil, a partir de dados extraídos da plataforma GitHub. O que deriva a seguinte questão que motiva a investigação desse estudo:

Que métricas são consideradas relevantes para medir e avaliar a eficiência do modelo de trabalho em equipas ágeis de desenvolvimento de software?

Para alcançar o objetivo geral e responder à questão de investigação definiram-se os seguintes objetivos específicos:

1. Identificar e compreender quais as métricas ágeis existentes para acompanhar o desempenho do projeto e das equipas de desenvolvimento de *software*;
2. Identificar quais as métricas ágeis que podem ser concebidas a partir da plataforma GitHub para acompanhar o desempenho do projeto e das equipas;
3. Desenvolver uma ferramenta para a visualização dos resultados das métricas recolhidas da plataforma GitHub;
4. Aplicação e avaliação da ferramenta desenvolvida por utilizadores da plataforma GitHub, com conhecimento em métricas ágeis de desenvolvimento de *software*.

Com a ferramenta de visualização de resultados, será possível visualizar métricas ágeis de acompanhamento de projetos e tarefas em equipas de desenvolvimento de *software* que utilizam o GitHub para armazenamento de código e gestão de tarefas, possibilitando melhorias do produto e do processo, decisões mais assertivas e a melhoria contínua da equipa ou empresa.

1.4. Abordagem metodológica

Visando alcançar os objetivos propostos no trabalho, foi adotada uma abordagem metodológica dividida em quatro fases:

1. A primeira consiste na análise da literatura sobre os conceitos relacionados com o tema do trabalho: análise sobre abordagem ágil, processo ágil de desenvolvimento de *software*, tipos de métricas ágeis existentes e os conceitos utilizados na plataforma GitHub.
2. A segunda fase está relacionada com a definição das métricas ágeis retiradas do GitHub e utilizadas para medir e visualizar a eficiência das equipas ágeis de desenvolvimento de *software*.

3. A terceira é a fase de modelagem e desenvolvimento da ferramenta para extração dos dados do GitHub e para a visualização dos resultados das métricas, levando em consideração o estudo realizado e as métricas ágeis identificadas. Essa fase contempla as etapas de análise de requisitos, modelagem e desenvolvimento da ferramenta proposta.
4. A quarta e última fase refere-se à aplicação e avaliação da ferramenta desenvolvida por utilizadores da plataforma GitHub e com conhecimento em métricas ágeis de desenvolvimento de *software*. Nesta fase serão definidos e planeados os objetivos da avaliação, executada a avaliação e por fim, realizada a análise dos resultados.

1.5. Estrutura e organização da dissertação

O presente estudo está organizado em cinco capítulos que pretendem refletir as diferentes fases do trabalho até à sua conclusão.

O primeiro capítulo introduz o tema da investigação e os objetivos da mesma, bem como uma breve descrição da estrutura do trabalho e da abordagem metodológica adotada. O segundo capítulo expõe o enquadramento teórico, designado por revisão da literatura, apresenta o tema estudado, conceitos e informações que fundamentam este trabalho com ênfase em métricas ágeis de *software*. No terceiro capítulo descreve-se a metodologia utilizada para a elaboração desta pesquisa, desde o levantamento dos requisitos à conceção da modelação da ferramenta. O quarto capítulo apresenta todas as informações recolhidas e a análise das mesmas com o objetivo de desenvolver a ferramenta de extração dos dados do GitHub e a visualização das métricas. Este capítulo apresenta, também, a modelação da ferramenta idealizada, desde os diagramas UML, passando pela arquitetura, protótipos da ferramenta e a avaliação da mesma. E, por fim, no quinto e último capítulo apresentam-se as conclusões deste estudo, bem como as limitações e as recomendações para trabalhos futuros.

Capítulo 2 – Revisão da Literatura

2.1. Abordagem ágil

2.1.1. Definição

A abordagem ágil refere-se a um conjunto de práticas e técnicas que surgiram para solucionar problemas comuns na gestão de projetos e para tornar os processos mais simples, flexíveis e iterativos. Esta abordagem teve uma procura crescente entre 1998 e 2002, tornando-se o período mais produtivo para o seu uso e desenvolvimento (Abrahamsson et al., 2008). Em 2001, foi escrito o manifesto ágil para o desenvolvimento de *software*, levando ao aparecimento de muitos métodos ágeis, que continuam a crescer exponencialmente (El Sheikh & Alnoukari, 2012).

Os autores Islam (2013) e El Sheikh e Alnoukari (2012), definem ágil não só como um conjunto de ferramentas ou uma simples metodologia, mas sim como uma filosofia que implica ser eficaz e flexível. Os métodos ágeis pretendem ser mais eficientes do que os métodos tradicionais, com os quais se faz mais com menos, procurando sempre a melhoria contínua. Para isso, sugerem equipas pequenas, auto-organizadas, autónomas, flexíveis e adaptáveis às mudanças, que realizem entregas rápidas e com qualidade para maior satisfação do cliente (Webb, 2015).

O manifesto ágil para desenvolvimento de *software*, publicado em fevereiro de 2001 com a colaboração de 17 profissionais, aborda os valores e princípios necessários para o desenvolvimento de *software* (Beck et al., 2001). O manifesto descreve 12 princípios (descritos no Apêndice A) e quatro valores principais para o conceito ágil: indivíduos e interações mais do que processos e ferramentas; *software* funcional mais do que documentação abrangente; colaboração com o cliente mais do que negociação contratual; e resposta à mudança mais do que seguir um plano.

Com a adoção destes princípios e valores nos projetos, há uma significativa melhoria dos resultados, já que o desenvolvimento do *software* se torna colaborativo, iterativo e incremental (Galup et al., 2020). De acordo com Conboy (2009) e Ozkan et al. (2020), após o Manifesto existiu um aumento da procura da mentalidade e desenvolvimento de *softwares* ágeis, o que resultou no surgimento de uma variedade de Metodologias de Desenvolvimento Ágeis tais como o Scrum, Kanban, XP (*eXtreme Programming*), *Lean Programming*, *Crystal*, entre outros. Estas metodologias são bem

aceitas pela comunidade de desenvolvimento de sistemas de informação (Conboy, 2009) e, embora divirjam nas práticas, todas procuram atender aos mesmos valores presentes no manifesto, possuindo uma abordagem iterativa, incremental e centrada nas pessoas para o desenvolvimento de *software*. As Metodologias de Desenvolvimento Ágeis permitem, assim, a adaptação rápida a mudanças de requisitos empresariais, de mercado e de tecnologias; promovem equipas auto-organizadas e flexíveis às mudanças, interação frequente com os clientes; e aumentam a qualidade do *software* e o desempenho do projeto (Venkatesh et al., 2020; Conboy, 2009).

2.1.2. Processo ágil de desenvolvimento de software

De acordo com os autores Vijayasarathy e Turk (2012) e Hazzan e Dubinsky (2007), o desenvolvimento de *software* tornou-se um processo mais ágil ao adotar os princípios criados no manifesto ágil, tendo características como: (1) Produção precoce e rápida de código de trabalho; (2) Alterações frequentes, pequenas e incrementais; (3) Colaboração e *feedback* contínuos dos clientes, tornando-os parte do processo de desenvolvimento; (4) Utilização de métricas que permitem a melhoria da qualidade do *software* e do processo de desenvolvimento; (5) O desenvolvimento de *software* orientado para testes (*Test-Driven*), testes unitários e de integração são incorporados no processo, e a cultura de qualidade é realçada; (6) Programação em pares num processo interativo de trabalho, onde a comunicação e a partilha de conhecimento estão presentes; (6) Aplicação da prática de *refactoring* para reduzir a complexidade e melhorar a legibilidade do *software*;

A abordagem ágil tem-se revelado substancialmente diferente dos métodos tradicionais de desenvolvimento, os quais enfatizam a análise extensiva antes da codificação, criação e manutenção dos modelos, períodos de tempo relativamente longos entre os produtos e a entrega dos marcos, estrutura organizacional burocrática, e geralmente poucas interações com o cliente (Vijayasarathy & Turk, 2012). O processo ágil traz mais benefícios ao promover o aumento da produtividade, melhoria da visibilidade, transparência do projeto, aumento da motivação da equipa, melhor previsibilidade de entrega, redução dos riscos, melhoria da qualidade, da manutenção do *software* e da comunicação e um melhor conhecimento do plano de projeto e dos objetivos (Baumeister et al., 2017).

2.2. Métricas de Desenvolvimento de *Software*

As empresas procuram, cada vez mais, aumentar a sua produtividade e qualidade. No entanto, segundo Lehner (2013), possuem uma certa dificuldade em avaliar como está a evoluir o processo de desenvolvimento e o produto final. As métricas de desenvolvimento de *software* surgiram precisamente para fornecer dados reais para analisar e avaliar esse processo de construção (Fenton & Bieman, 2014), conduzindo a uma melhoria contínua (Ram et al., 2019). Para Eisty et al. (2018), o objetivo das métricas é facultar uma visão contínua sobre os produtos e processos através de um cálculo que avalia a eficácia do *software* ou processo.

As métricas começaram a ser estudadas há muitos anos e com o crescimento das metodologias ágeis, o rápido avanço das necessidades dos utilizadores e da tecnologia, a flexibilidade nas alterações e a complexidade do ecossistema de *software*, torna-se cada vez mais relevante usá-las para compreender, controlar e melhorar o que fazemos e como o fazemos (Kupiainen et al., 2015). Segundo os autores Suresh et al. (2012), Ram et al. (2019) e Kupiainen et al. (2015), as métricas permitem acompanhar o progresso do projeto, realizar estimativas e planeamentos, avaliar a complexidade de um algoritmo e a eficácia e segurança do *software*, ajudar na compreensão, melhorar a qualidade de *software* e a resolução de problemas de processo, compreender objetivos comerciais e melhorar a comunicação e a motivação das pessoas.

Conforme Matthies et al. (2016), de forma a obter uma maior eficiência nos resultados, as métricas utilizadas devem seguir os princípios da abordagem ágil, ter resultados confiáveis, ser adaptadas e relevantes ao contexto de uma equipa e estar sujeitas a aperfeiçoamento iterativo.

2.2.1. Uso das métricas de software

Os programas de métrica permitem a avaliação quantitativa dos processos de *software*, são úteis na identificação e resolução dos problemas do processo e na motivação das pessoas (Ram et al., 2019). Contudo, para se obterem resultados significativos para os objetivos do projeto, as métricas devem ser mapeadas. Segundo os autores Hazzan e Dubinsky (2007), o mapeamento deve ser avaliado regularmente para que não haja redundâncias e para que sejam adotados os ajustes necessários.

A recolha das métricas não deve afetar o desempenho do processo ou do produto. Para facilitar a escolha das métricas que melhor se adequam aos seus objetivos, as empresas devem procurar responder às seguintes questões: Porque é que as métricas são necessárias e como serão utilizadas? Quem decide o que é medido? O que deve ser medido? Quando e como é que as métricas devem ser recolhidas? Quem reúne as métricas? (Hazzan & Dubinsky, 2007; Kupiainen et al., 2015). As métricas escolhidas precisam de ser relevantes e importantes para a equipa, não devem ser utilizadas para as comparar, nem para selecionar métodos de desenvolvimento (Kupiainen et al., 2015).

Uma abordagem que pode ser utilizada para auxiliar a escolha de métricas é o GQM *Paradigm (Goal-Question-Metric Paradigm*), que têm como objetivo definir a medição a partir do zero (Mendonc & Basili, n.d.). O primeiro passo do paradigma GQM consiste em definir os objetivos das métricas, adaptados às necessidades específicas de uma organização. Os objetivos são aprimorados de uma forma operacional e rastreável através de um conjunto de perguntas quantificáveis. As perguntas, por sua vez, implicam um conjunto de métricas e dados para recolha.

Para atingir os objetivos esperados com as métricas, alguns autores sugerem várias etapas de aplicação. O autor Matthies et al. (2016) propõe um modelo iterativo de métricas com quatro passos: (1) Definição: primeiro deve haver um entendimento comum da prática a ser executada e medida. A métrica deve ser suficientemente detalhada, assim como os seus resultados. (2) Execução da consulta: após a recolha dos dados de desenvolvimento, executa-se o cálculo das métricas para listar os resultados. (3) Análise do Contexto: analisar a qualidade dos resultados da etapa anterior. (4) Melhoria: uma parte importante do ciclo de vida da métrica é a sua melhoria contínua, isto significa assegurar que a métrica se enquadra no contexto do projeto e da equipa. O conhecimento que foi recolhido na etapa de melhoria pode depois ser novamente explicitado na primeira etapa do ciclo de vida.

Adicionalmente, os autores Mendonc e Basili (n.d.) sugerem quatro passos na aplicação de métricas, sendo: (1) Caracterização do quadro de métricas: esta primeira fase tem como objetivo a identificação dos componentes chave: métricas, atributos, dados, grupos de utilizadores e utilização dos dados. (2) Análise *Top-Down*: esta fase é utilizada para capturar os objetivos dos utilizadores de dados e para mapear os dados que estão a ser recolhidos. Isto ajuda a compreender a real necessidade dos utilizadores, empregando o método Paradigma GQM. (3) Análise *Bottom-Up*: o objetivo é descobrir novas

informações que podem ser úteis nos dados existentes, melhorando o conhecimento e a utilização dos dados. (4) Validação do trabalho: nesta fase é realizado a validação dos resultados: validação da abordagem, dos objetivos, melhorias, eficácia e custos.

2.2.2. Tipos de Métricas Ágeis de Software

Como mencionado em cima, depois de se entender o problema e de se definirem os objetivos é necessário escolher as métricas que serão utilizadas. Existem vários tipos de métricas para aplicar conforme as diferentes necessidades das empresas.

O autor Eisty et al. (2018), classifica as métricas de desenvolvimento de *software* em métricas de código, que medem a complexidade (e.g., LCOM4 – Métodos de Falta de Coesão) e outras características do código (e.g., Densidade de defeitos); em métricas de processo, que são recolhidas durante períodos de tempo mais longos e fornecem informações sobre o processo de desenvolvimento de *software* (e.g., *Cycle Time*); em métricas de teste, que medem e monitorizam as atividades de teste (e.g., Cobertura de testes); em métricas gerais de qualidade, relacionadas com propriedades desejáveis do *software* (e.g., Interoperabilidade); em métricas de desempenho de execução do *software* em plataformas informáticas de alto desempenho, que abordam o tempo de execução, armazenamento (e.g., RAM ou espaço em disco) ou escalabilidade (e.g., Tempo vs. CPUs/-núcleos); e por fim, em métricas de reconhecimento que medem como um projeto ou os seus promotores quantificam o interesse externo no seu trabalho (e.g., Citações ou *Downloads).*

Por outro lado, o autor Kupiainen et al.(2015), classifica as métricas de desenvolvimento de *software* em métricas para *sprint* e planeamento do projeto, que compreendem as métricas para priorização de atividades e estimativas de esforço (e.g., Pontos de Caso de Uso), de definição de âmbito do projeto utilizadas para estimar o tamanho e números de *features* que podem ser desenvolvidas (e.g., Velocidade) e de recursos e flexibilidade de desenvolvimento (e.g., Eficácia da Equipa); em métricas que ajudam a compreender e a corrigir problemas em processos de engenharia de *software* (e.g., VSM - *Value Stream Maps*); em métricas utilizadas para motivar os colaboradores a reagir rapidamente aos problemas (e.g., *Build Status)*; e por último, em métricas para o acompanhamento do progresso do projeto, que compreendem as métricas aplicadas para monitorizar a evolução do projeto (e.g., *Burndown*), para simplificar aspetos do

desenvolvimento de *software* e aumentar a visibilidade para todas as partes interessadas (e.g., *Cost Type)*, compreender se os objetivos do projeto podem ser alcançados, para reduzir o âmbito de uma iteração ou para acrescentar mais recursos, caso seja necessário (e.g., *Lead Time)*, para adequar o fluxo de trabalho e para prevenir a sobrecarga de pessoas (e.g., *Work in Progress).*

2.2.3. Métricas ágeis aplicadas no presente trabalho

Com o objetivo de monitorizar a eficiência do processo de construção de *software*, ter uma previsibilidade mais assertiva na entrega de tarefas, verificar a capacidade de produção das equipas e consequentemente, contribuir para a qualidade final do produto (Albino, 2017), no presente trabalho foram aplicadas cinco métricas que são utilizadas no mercado e fundamentais para as equipas ágeis de desenvolvimento de *software*.

Uma delas foi a métrica *Lead Time* (LT) que, de acordo com Niemi et al. (2021), compreende ao tempo entre o início de um trabalho, processo ou projeto e o aparecimento dos seus resultados. É o tempo médio que um pedido demora a percorrer todo o processo, desde a data da sua criação até à entrega ao cliente (Kupiainen et al., 2015).

Outra métrica aplicada foi a *Cycle Time*, que refere-se à quantidade de tempo desde que o trabalho realmente começou (início do desenvolvimento) até à conclusão do pedido (entrega ao cliente). O *Cycle Time* é um dos fatores mais críticos que afeta diretamente a taxa de produção e, deste modo, é um indicador-chave da eficiência do processo (Budacu & Pocatilu, 2018). A Figura 1 mostra um exemplo da *Cycle Time* e *Lead Time*.

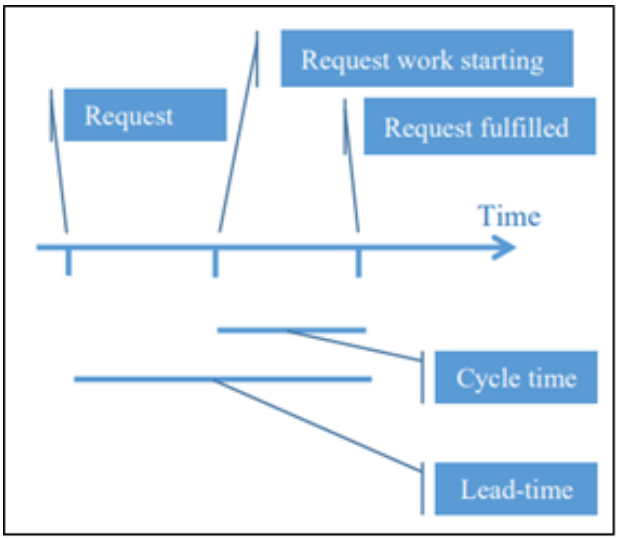


Figura 1 - Lead Time e Cycle Time (Budacu & Pocatilu, 2018).

Foi também utilizada a *Cumulative Flow Diagram* (CFD) que, para Caroli (2020), representa graficamente a evolução do trabalho, exibe obstáculos e alerta sobre possíveis instabilidades. É uma ferramenta fundamental para rastrear, prever a realização das tarefas e indicar a necessidade de agir sobre o fluxo de trabalho (Figura 2).

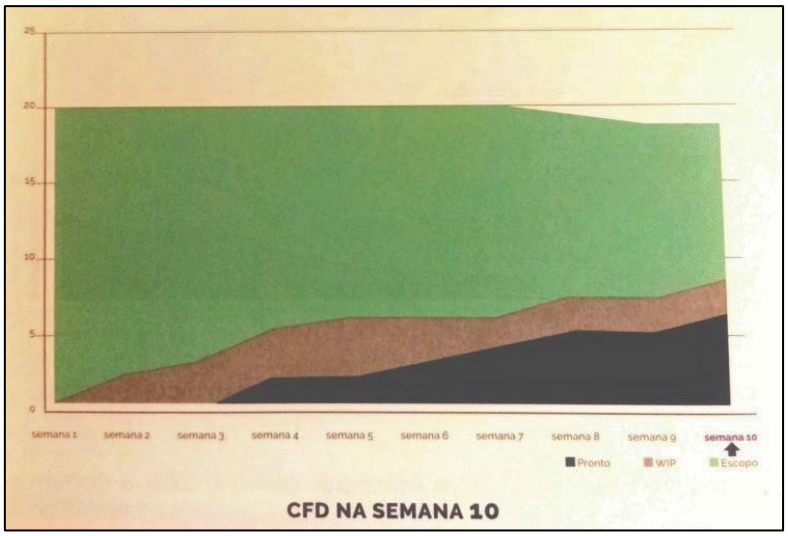


Figura 2 - Exemplo Cumulative Flow Diagram na semana 10 de um projeto(Caroli, 2020)

O *Work in Progress* (WIP), também utilizado, refere-se ao trabalho em progresso, ou seja, ao número de itens de trabalho atualmente em curso. Para a realização de um trabalho mais eficiente e melhorar a gestão da capacidade é importante limitar o trabalho em progresso (Hemalatha et al., 2021). A redução do WIP resulta num nível mais elevado de liquidez, melhor fluxo de caixa, melhor serviço ao cliente e reduz os riscos para o

negócio. A Figura 3 apresenta um exemplo onde o WIP foi limitado a 3 itens nas etapas de trabalho em progresso.

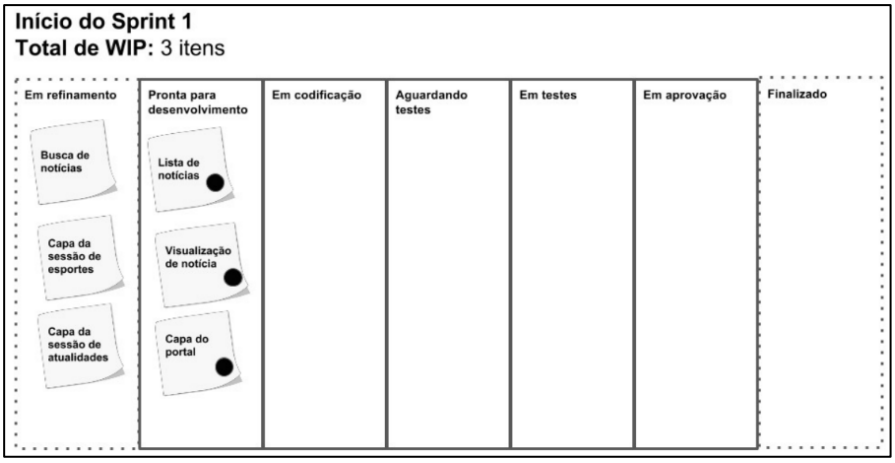


Figura 3 - WIP limitado para 3 itens nas etapas de trabalho em progresso (Albino, 2017)

Por fim, a métrica *Throughput* (taxa de transferência), mostra a vazão ou a quantidade de itens entregues num determinado período de tempo. Com ela é possível medir a quantidade de itens que podem ser entregues por uma equipa (Caroli, 2020). A Figura 4 mostra um gráfico *Throughput* de itens entregues por semana.

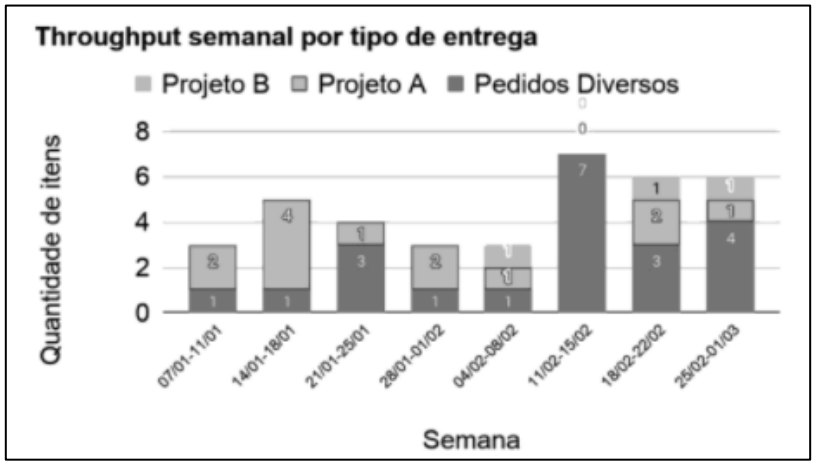


Figura 4 - Gráfico Throughput por tipo de itens de entrega (Muniz et al., 2021)

2.3. Plataforma GitHub

Várias metodologias da engenharia de *software* baseiam-se em sistemas de controlo de versões que permitem armazenar artefactos de código fonte e realizam a gestão de alterações. Um destes sistemas é o Git, utilizado principalmente por empresas

de desenvolvimento de *software* de código fonte aberto. Existem algumas plataformas que hospedam gratuitamente os repositórios criados em GIT, entre elas, a plataforma GitHub (Ortu et al., 2020).

O GitHub é uma companhia que viu imenso potencial no Git e construiu uma camada de *web services* em cima das funcionalidades encontradas (Dawson & Straub, 2016). Os repositórios do Git são o centro do desenvolvimento de *software* e do GitHub, e este possui outros benefícios além de armazenar repositórios, como poder colaborar em fluxos independentes, rever o trabalho em progresso, visualizar o progresso da equipa, documentar erros, novas funcionalidades ou outras necessidades do projeto (Beer, 2018).

Graças ao aumento das ferramentas de desenvolvimento de *software* de código aberto (OSS), o número de utilizadores e de projetos tem vindo a crescer todos os anos, os quais não poderiam existir sem a continuação e contribuição de outros desenvolvedores. Existem algumas plataformas para essa colaboração, o GitHub é a maior delas e aloja um milhão de repositórios de projetos OSS[1] (Subramanian et al., 2022). Nela são gerenciados os projetos, os problemas e as alterações, além de fornecer diversas informações que podem ser usadas sob a forma de métricas para resolver vários desafios da engenharia de *software* (Şeker et al., 2021).

A plataformas sociais de desenvolvedores, como GitHub, são parte integrante do desenvolvimento de *software* colaborativo, proporcionam um espaço onde as pessoas podem mostrar ativamente os seus projetos, partilhar o seu código fonte, procurar colaborações, fazer perguntas e resolver questões. A partilha do código fonte sobre estas plataformas está associada à ideia de que deve haver conteúdos disponíveis *online* para que a comunidade possa utilizar a informação como quiser. O conteúdo que está a ser gerado é fortemente dependente da participação das pessoas (Bhasin et al., 2021).

O presente trabalho visa a integração da ferramenta de visualização de métricas a ser desenvolvida com a plataforma GitHub, extraindo dados de métricas relevantes para equipas ágeis de desenvolvimento de *software*. Nos próximos tópicos, serão explicados os conceitos da plataforma utilizados no trabalho.

[1] OSS (*Open Source Software*) são projetos de *software* de código-fonte aberto.

2.3.1. Conceitos do GitHub utilizados no presente trabalho

De forma a compreender os objetivos do presente trabalho, é importante conhecer alguns conceitos relacionados com a plataforma GitHub.

O primeiro conceito é o de repositório. Este armazena todos os arquivos e o histórico de revisão do projeto, é possível compartilhá-lo com outros utilizadores da mesma organização, discutir e gerenciar o trabalho do projeto (*GitHub Docs*, 2022). Qualquer repositório público pode ser clonado para o ambiente local do computador, e nele receber alterações e executar o código, mas somente utilizadores com permissão podem publicar no repositório remoto (Guthals & Haack, 2019).

Cada repositório tem o que é denominado *branch.* Uma *branch* é uma série de *commits* e pode ser utilizada para experimentar ou criar uma nova funcionalidade. Ao criar um novo projeto no Git, cria-se uma *branch* padrão chamada de *master* e a partir dela, podem ser criadas várias *branches* para trabalhar localmente sem afetar a padrão (Beer, 2018). Assim, é possível corrigir erros, construir diferentes funcionalidades e experimentar novas ideias, com segurança, numa área contida no repositório local (Guthals & Haack, 2019). A *branch master* deve conter os trabalhos finalizados das *branches* locais para ser publicado em produção (Şeker et al., 2021).

Já um *commit* é a menor unidade de trabalho do Git e representa um pequeno grupo de alterações na *branch* (Guthals & Haack, 2019). Sempre que se alterarem um ou mais arquivos, é possível criar um novo *commit*, o GitHub irá registar o momento das alterações e o autor que as realizou (Beer, 2018).

Assim que as alterações feitas na *branch* local estiverem concluídas para ir para produção, o *push* deve ocorrer (Guthals & Haack, 2019). Ao executar um *push*, as alterações confirmadas (*commits* realizados) no repositório local são enviados para o repositório remoto do GitHub, permitindo o acesso a outros utilizadores (GitHub Docs, 2022; Şeker et al., 2021). Se as alterações foram efetuadas diretamente no *website* do GitHub não é necessário realizar o *push*, caso contrário, esta etapa deve ocorrer (Guthals & Haack, 2019).

Os pull requests são o principal meio para contribuir para um projeto. São parte integrante das atividades de sincronização de código e, como o seu nome indica, não correspondem a uma mudança feita diretamente no repositório central, mas sim à solicitação da mesma (Ortu et al., 2020). Para o solicitar, o utilizador deve primeiro criar

uma *branch* do projeto e realizar alterações, posteriormente solicita o *pull request* que será analisado e revisto por outros utilizadores antes do *commit* na *branch* padrão, ou seja, todas as alterações serão analisadas antes da sua inclusão na *branch* padrão do projeto (Cosentino et al., 2014). Os *pull requests* são um aspeto importante na colaboração do desenvolvimento de *software* e são muito populares no GitHub (Ortu et al., 2020).

Com as alterações finalizadas, revistas e aprovadas é possível realizar o *merge*. Este corresponde ao processo de recolha do trabalho completo de uma *branch* e à sua incorporação noutra *branch*. O mais comum é fazer o *merge* de uma funcionalidade da *branch* local para a *branch master* (Beer, 2018). Quando é realizada a ação *merge* assumimos que a *issue* deve ser fundida na *branch* principal (Ortu et al., 2020).

As *issues* (problemas) são criadas para sugerir melhorias, reportar erros, discutir ideias, inserir tarefas, questões relacionadas com o repositório, comentários, organizar e monitorizar tarefas. Permitem acompanhar o trabalho no GitHub e podem ser criadas de várias formas: a partir de um repositório, de um item numa lista de tarefas, de uma observação num projeto, de uma linha de código ou de um comentário sobre um problema (Heller, 2021). Os sistemas de rastreio de *issues*, como o GitHub, permitem aos utilizadores e desenvolvedores discutirem problemas relacionados com o *software* ou novas funcionalidades (Paing et al., 2022). Existem dois estados para uma *issue*: aberto e fechado. O estado aberto indica que a *issue* ainda está ativa e à espera de ser abordada; o fechado indica que foi abordada, ou seja, resolvida ou encerrada (Zhou et al., 2021). Para cada *issue* é possível atribuir *label* (etiqueta), *assignatures* (atribuição), projetos e *milestones* (marcos). As *issue* criadas podem ter um ou mais responsáveis por realizá-las e é o GitHub quem realiza essas atribuições (Heller, 2021).

Outro conceito importante é o de etiqueta ou *label*. As etiquetas auxiliam a gestão dos trabalhos no GitHub ao permitirem classificar *issues*, *pull requests* e discussões (Heller, 2021). No sistema de administração de problemas, o GitHub fornece uma variedade de etiquetas que podem ser usadas para representar informações e decisões sobre questões relatadas. Os colaboradores do projeto podem adicionar livremente qualquer etiqueta para classificar o tipo da *issue*, aumentar a visibilidade e facilitar a resolução dos problemas. O GitHub possui algumas *labels* já criadas, mas é possível criar outras (Kim & Lee, 2021).

Os *milestones*, ou marcos, permitem acompanhar o progresso do projeto e, para um melhor controlo, é possível associá-los a *issues* e *pull requests* (*GitHub Docs*, 2022).

O GitHub possui várias funcionalidades que facilitam as tarefas dos desenvolvedores como, por exemplo, organizar e controlar repositórios e o trabalho colaborativo de um grupo de pessoas no mesmo projeto (Heller, 2021). É importante conhecer os conceitos básicos do GitHub para perceber o correto funcionamento e o fluxo de trabalho da plataforma, que será apresentado a seguir.

2.3.2. Fluxo do GitHub

A plataforma GitHub segue um sistema *fork-and-pull*, onde os utilizadores criam uma cópia própria de um projeto, fazem alterações, *commits* e criam um *pull request* para propor contribuições no repositório do projeto. Os moderadores do projeto e o público em geral podem rever as alterações efetuadas e solicitar modificações, após as quais os moderadores podem decidir incorporá-las ou não (Subramanian et al., 2022).

O fluxo do GitHub é leve e útil não só para os desenvolvedores, mas também para outro tipo de utilizadores. Para seguir o fluxo, são necessários uma conta e um repositório na plataforma (*GitHub Docs*, 2022). Qualquer pessoa que tenha uma conta pode realizar uma cópia completa de um repositório e começar a trabalhar com a mesma base de código do original (Perez-Riverol et al., 2016), seguindo determinados passos.

Primeiramente, o utilizador cria uma *branch* a partir da principal para realizar o seu trabalho, nela podem ser desenvolvidas todas as alterações desejadas sem a preocupação de alterar a principal (Miyashita et al., n.d.). Podem desenvolver-se e testar-se novas funcionalidades, corrigir erros e contribuir com melhorias. Vários utilizadores (colaboradores) podem trabalhar simultaneamente no mesmo repositório sem conflitos (Perez-Riverol et al., 2016). Ao terminar as alterações deve-se realizar o *commit* e o *push* e, depois de tudo finalizado, solicitar a revisão (*GitHub Docs*, 2022). Criar uma *branch* e solicitar o *pull request* constitui um método simples de colaboração entre equipas e permite o controlo sobre as contribuições externas que são aceites pelo proprietário do repositório (Perez-Riverol et al., 2016). O pedido de revisão (*pull request*) é a última etapa antes da entrega da alteração, proporcionando uma oportunidade de revisão de código ao funcionar como um mecanismo de rastreio quando os projetos de código aberto recebem contribuições de outros participantes. A revisão ajuda a identificar erros e a indicar as modificações necessárias antes do *merge* das alterações na *branch* principal (Kalliamvakou et al., 2015). Se identificada a necessidade de realizar modificações, o

desenvolvedor precisa de as realizar e criar um novo pedido de revisão (Miyashita et al., n.d.). Após a aprovação das revisões é necessário realizar *merge* , o que fará com que as alterações e mudanças desenvolvidas na *branch* apareçam no repositório central (Kalliamvakou et al., 2015). E por fim, após conclusão do *merge*, é importante excluir a *branch* para que outras pessoas não a usem acidentalmente (*GitHub Docs*, 2022). A Figura 5 ilustra o fluxo do GitHub.

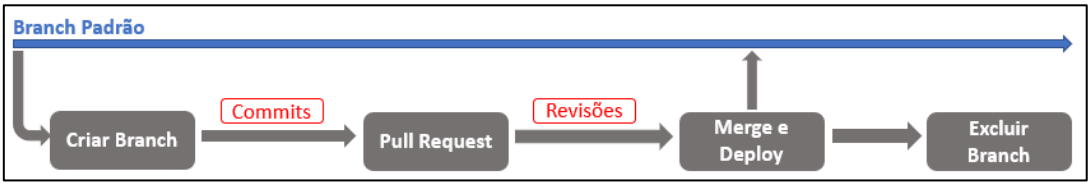


Figura 5 - Fluxo GitHub

2.3.3. Métricas do GitHub

As atividades dos desenvolvedores podem ser vistas nalgumas métricas do GitHub. Essas métricas são criadas a partir das atividades de criação da cópia de um projeto, realização do *commit*, solicitação do *pull request* e aprovação ou não do *merge* no repositório principal (Şeker et al., 2021). A plataforma disponibiliza alguns gráficos e métricas de desenvolvedores e de repositórios, como a dependência de projetos no repositório, contribuidores, *commits* e cópias de um repositório (*GitHub Docs*, 2022).

A visão geral do atual estado do projeto é apresentada no gráfico *Pulse* (Davis, 2015), que mostra o número de *pull requests* com *merge* aberto e completo, *issues* fechadas e abertas, discussões não resolvidas e utilizadores comprometidos ativamente no projeto (Beer, 2018). Na Figura 6 é possível observar o gráfico *Pulse.*

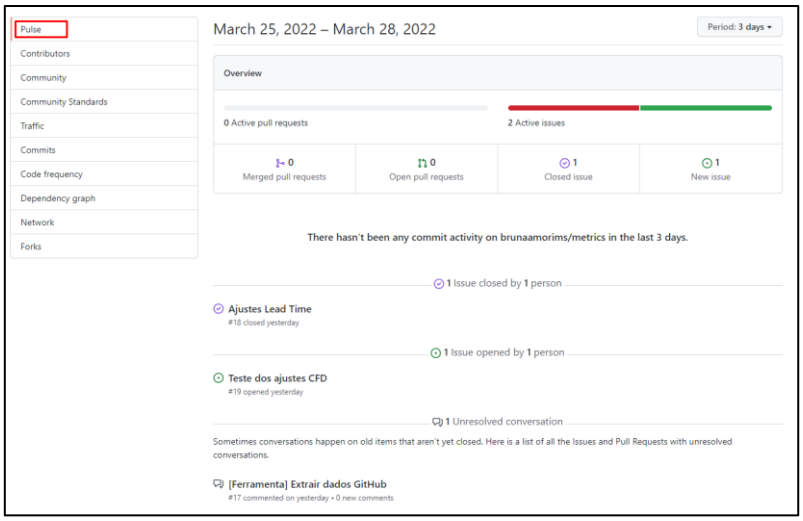


Figura 6 - Gráfico Pulse (GitHub Docs, 2022)

O gráfico *Contributors* apresenta os utilizadores que mais contribuíram para o projeto durante um período de tempo específico (Pipinellis, 2018), mas só aqueles com *commits* adicionados (*merge* realizado) no repositório principal (Beer, 2018), cujo exemplo pode ser visto na Figura 7.

Figura 7 - Gráfico Contributors (Pipinellis, 2018)

Outra métrica que se encontra no GitHub é a *Community*, que ajuda a entender a atividade de contribuição (discussões, *issues* e *commits*), as exibições da página e o crescimento de discussões, todos ao nível do repositório (Pipinellis, 2018). A funcionalidade de discussão no GitHub foi concebida para apoiar conversas relacionadas ou não com o código entre os membros (Hata et al., 2022). Como pode ser visto na Figura 8, este gráfico possui informações das atividades de contribuição, mostrando o total de contribuições para as discussões, *issues* e *pull requests*; o total de visualizações de páginas de discussões segmentadas por usuários conectados e usuários anônimos; o total de

contribuidores diários de discussões que reagiram, votaram, marcaram uma resposta, comentaram ou postaram no período de tempo selecionado; e o total de novos contribuidores de discussões que reagiram, votaram, marcaram uma resposta, comentaram ou postaram no período de tempo selecionado (*GitHub Docs*, 2022).

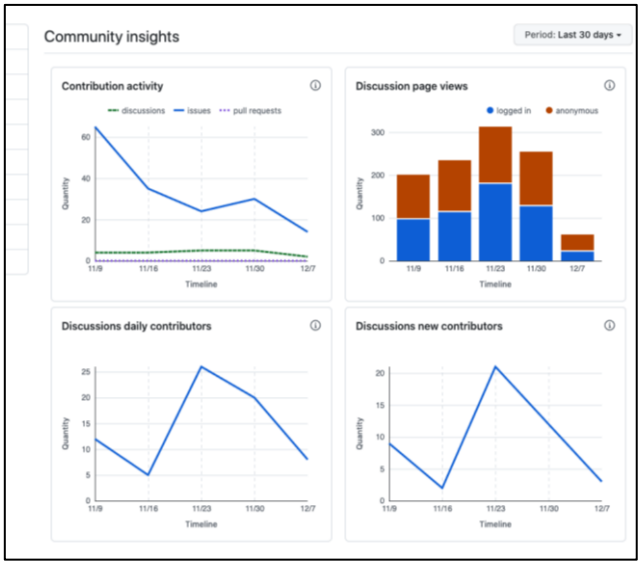


Figura 8 - Gráfico Community (GitHub Docs, 2022)

Os dados referentes ao tráfego de um repositório podem ser vistos no gráfico *Traffic*, que apresenta a quantidade total de clones do repositório, a quantidade total de utilizadores que o clonaram, o total de visualizações do repositório e o total de utilizadores que o visualizaram (Beer, 2018). Este gráfico só pode ser visualizado pelo proprietário ou pelos membros do repositório (Pipinellis, 2018), como pode ser observado na Figura 9.

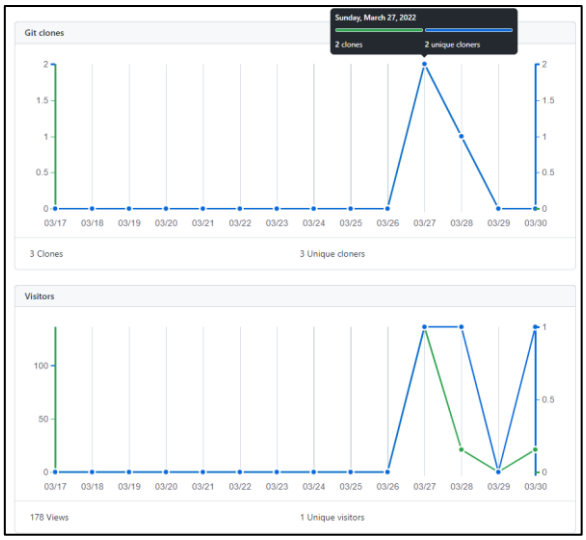


Figura 9 - Gráfico Traffic

Os desenvolvedores clonam repositórios por diferentes razões, como adicionar novas funcionalidades, corrigir erros, alterar configurações, entre outras. Com as alterações finalizadas é preciso fazer o *commit* para enviá-las para o repositório principal (Ren et al., 2018). As alterações realizadas ao conteúdo do repositório podem ser visualizadas nos gráficos de *Commit* e *Code Frequency*. O primeiro gráfico (Figura 10) apresenta a quantidade e a média de *commits* feitos num determinado período no repositório (*GitHub Docs*, 2022).

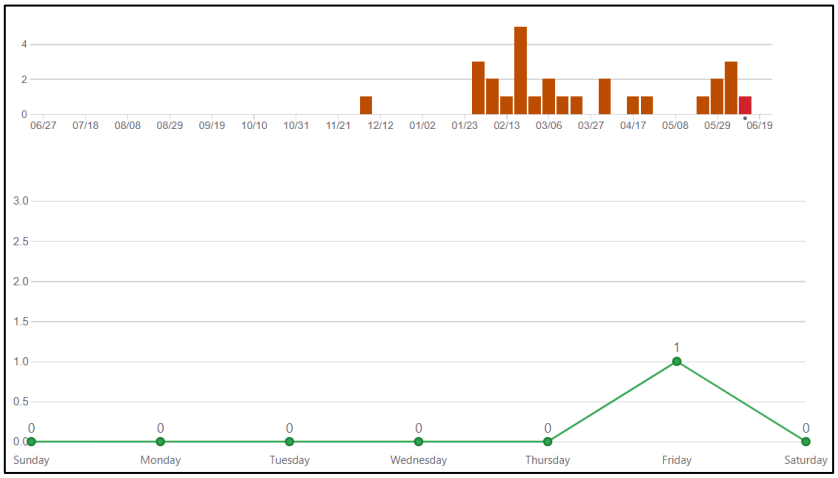


Figura 10 - Gráfico Commits

Já o gráfico *Code Frequency* () exibe as adições e exclusões de conteúdo no repositório por semana (Pipinellis, 2018).

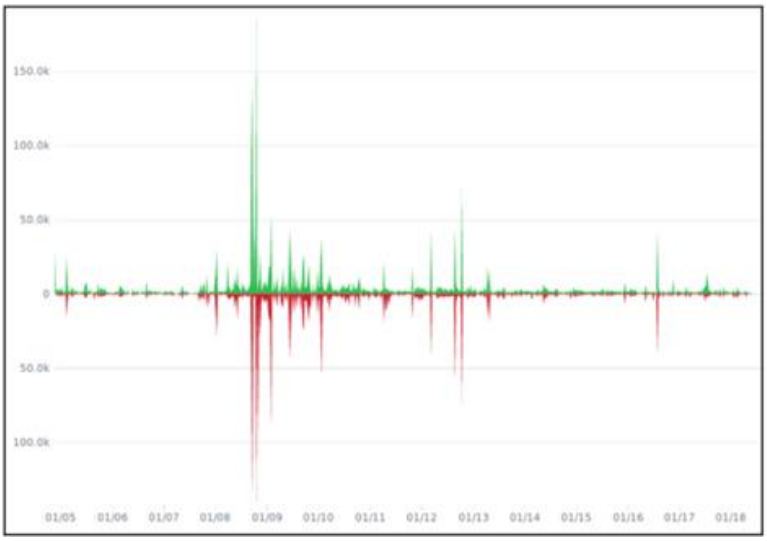


Figura 11 - Gráfico Code Frequency (Pipinellis, 2018)

Para entender melhor as ligações existentes entre o repositório original e os seus clones, existe o gráfico *Network*, que indica o histórico de *branches* e *commits* de todas as redes do repositório (Loeliger & McCullough, 2012). Este gráfico é útil para verificar o histórico de *branches* de toda a rede do repositório e o tipo de trabalho que os utilizadores estejam a realizar (Beer, 2018). Um exemplo do gráfico está presente na Figura 12.

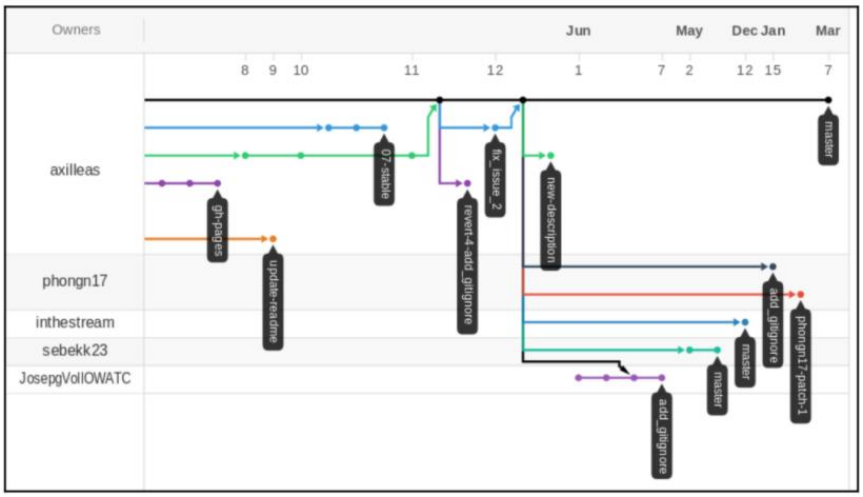


Figura 12 - Gráfico Network (Pipinellis, 2018)

As métricas apresentadas acima são métricas de atividades de desenvolvedores a nível de repositório e nem sempre são adequadas e suficientes. A equipa precisa de conseguir observar os vários parâmetros do fluxo de trabalho para obter uma visualização única e abrangente de todos os dados do projetos, repositórios e atividades, a partir dos

quais é possível melhorar a eficiência do processo e proporcionar a melhoria contínua (Caroli, 2020).

Para aceder a outras informações da plataforma, o GitHub fornece a *API Rest* (Shimada et al., 2022) que foi utilizada para recolher os dados necessários para as métricas deste trabalho.

2.3.4. API Rest

A quantidade de serviços disponíveis ao público na internet, APIs (*Application Programming Interfaces*), está a crescer rapidamente, assim como a implementação de *Web Services Rest* (*Representation State Transfer*) que permitem aos consumidores o uso e o acesso mais facilitado a serviços e dados (Neumann et al., 2021).

Por ser uma plataforma popular para a colaboração de projetos de código aberto e pelo facto de grande parte da sua atividade ser pública, o GitHub armazena uma grande quantidade de dados. Para interagir com esses dados e visualizá-los, foi desenvolvida uma funcionalidade que cria integrações, recupera dados e automatiza fluxos de trabalho (Rusk & Coady, 2014). O GitHub dispõe, assim, de uma *API Rest* que permite obter informações imediatas sobre a quantidade de colaboradores do repositório, linguagens de programação utilizadas, *branches* existentes, informações do projeto, da *issue*, de *pull request*, de *commits*, e várias outras (Rusk & Coady, 2014). A *API Rest* fornece ao utilizador final dados brutos no formato *JSON* (Notação de Objetos *JavaScript*) e é uma grande fonte de informação que pode ser usada para promover melhorias na equipa, como por exemplo, no planeamento (Pachouly et al., 2022).

A Figura 13 apresenta um exemplo de utilização. Nesse esquema o cliente Java envia um pedido para a *API Rest* do GitHub que responde com um texto no formato *JSON*. Quando o cliente Java recebe a resposta, processa os dados e armazena-os na sua base de dados local (Pachouly et al., 2022).

Como já mencionado, o GitHub possui uma grande comunidade de desenvolvedores que utilizam os seus serviços. Para facilitar a construção de ferramentas por parte da comunidade, a *API Rest* oferece o acesso a várias funcionalidades, encontradas ou não nas *interfaces* da plataforma (Loeliger & McCullough, 2012).

Devido à vasta gama de dados e à facilidade em acedê-los, foi utilizada a *API* para integração com a ferramenta do presente trabalho. Os detalhes sobre a construção, desenho e requisitos serão explicados de seguida.

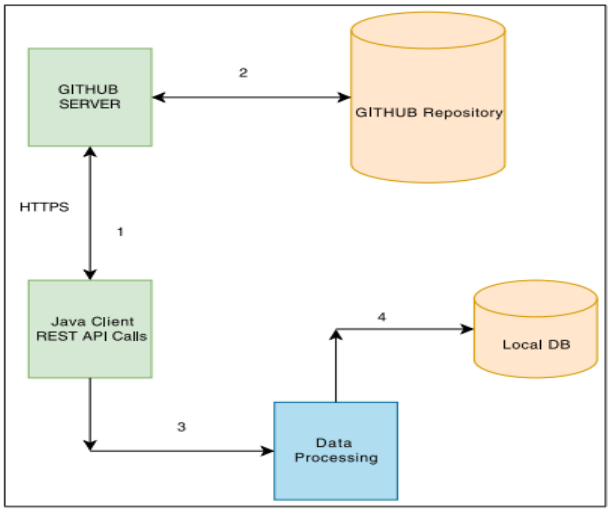


Figura 13 - Exemplo de uso API Rest GitHub (Pachouly et al., 2022)

Capítulo 3 – Metodologia

Para atingir os objetivos do presente trabalho, realizou-se um estudo constituído por quatro etapas. Na primeira, foi elaborado um questionário *online* por meio da ferramenta de Formulário do Google, descrito nos Apêndices B e C, para a recolha de dados com pessoas de diferentes cargos, que trabalhassem em projetos ágeis de desenvolvimento de *software*. Na segunda etapa, foi feita a análise das respostas obtidas para identificar uma lista inicial de requisitos. Na terceira, realizou-se uma sessão de *brainstorming* (resultado da sessão no Apêndice F) com cinco dos doze inquiridos que responderam o questionário, para colaboração de ideias, validação, definição de requisitos finais e modelação da ferramenta, e por último, foram realizados testes e avaliação da ferramenta por dois utilizadores que participaram da sessão de *brainstorming*, descrito no Apêndice E.

3.1. Etapa 1: Questionário

Os questionários são utilizados, principalmente, no início da atividade de levantamento de requisitos e é uma boa alternativa para o analista contactar um grande número de pessoas e obter opiniões sobre vários aspetos de um sistema (Mishra & Mohanty, 2011). Se existe um grande número de participantes a ser entrevistados e em diferentes lugares, um questionário *online* é uma opção viável para obter uma vasta quantidade de informações, num curto espaço de tempo e com baixo custo (Pohl & Rupp, 2016).

Para ser efetivo, os termos, os conceitos e o âmbito precisam de ser bem estabelecidos e entendidos pelos participantes e por quem irá desenvolver o questionário, as questões devem ser formuladas adequadamente para evitar a recolha de informações redundantes e irrelevantes (Mate & Silva, 2005). Criar um questionário apropriado é, muitas vezes, complicado e lento e requer um conhecimento profundo do domínio em questão, precisa ser adequado ao perfil dos utilizadores que irão responder e não deve ser extenso (Pohl & Rupp, 2016). As respostas de um questionário podem ser estruturadas (predeterminadas) e não estruturadas (abertas). As questões com respostas não estruturadas são do tipo espaço em branco e o participante precisa escrever sua opinião. Já nas questões com respostas estruturadas, as opções estão predeterminadas e o participante apenas assinala a que melhor corresponde à sua opinião, podem ser do tipo

múltipla escolha ou lista de verificação e permite fazer a avaliação mesmo o participante não sendo capaz de explicar o seu conhecimento (Mishra & Mohanty, 2011).

Neste estudo foi realizado um questionário *online* composto de 15 questões, com respostas predeterminadas e abertas, para reunir informações específicas e uniformes. O questionário foi respondido entre 01/02/2022 e 25/02/2022, por 12 colaboradores de empresas privadas de desenvolvimento de *software* que trabalham em projetos ágeis de *software* e possuem diferentes papéis e responsabilidades. As empresas são das mais variadas áreas, como gestão pública, setor da saúde, gestão de transportes rodoviários, indústria de veículos, setor da banca, companhia aérea e setor de distribuição de energia.

3.2. Etapa 2: Análise das respostas do questionário

Após a realização do questionário, foi feita a análise e interpretação das respostas obtidas de forma a identificar os requisitos funcionais para a ferramenta de visualização de métricas ágeis. Como já mencionado, os inquiridos possuem diferentes cargos e funções em projetos ágeis de desenvolvimento de *software*, o que ocasionou distintas visões e opiniões. Esta análise, em conjunto com os resultados obtidos, possibilitou enumerar características, funcionalidades e métricas pertinentes para a ferramenta. Após análise e entendimento das necessidades, foi realizada uma sessão de *brainstorming* e a validação de requisitos com alguns inquiridos que responderam o questionário, explicada de seguida.

3.3. Etapa 3: Idealização

Após se conhecer o problema e de se entenderem as necessidades dos utilizadores, realizou-se uma sessão de *brainstorming* em duas etapas. Na primeira etapa foi fomentada a criatividade em grupo, através da qual ideias e pensamentos foram partilhados espontaneamente entre os membros, a fim de alcançar soluções práticas para problemas (Al-Samarraie & Hurmuzan, 2018). Na segunda etapa, as ideias foram filtradas para definir e validar os requisitos finais da ferramenta (Chemuturi, 2012).

Dos 12 inquiridos que responderam o questionário, foram selecionados cinco para participar nessa sessão de *brainstorming*: um desenvolvedor, um *agile master*, um líder de projetos, um *project manager* e um analista funcional. Optou-se pela seleção dessas

cinco pessoas dado o bom conhecimento em abordagem ágil e a contribuição para esse estudo, e com a criação de um grupo pequeno a sessão não seria muito demorada, cansativa e contaria com a participação e envolvimento de todos (Laplante & Kassab, 2022). Ao término da sessão e com as validações realizadas, foram descritos os requisitos funcionais, ou seja, as necessidades, as características e as funcionalidades desejadas na ferramenta e os requisitos não funcionais, especificando como a ferramenta deve ser feita.

Em seguida, realizou-se a modelação da ferramenta utilizando-se da Linguagem de Modelação Unificada (*Unified Modeling Language*, UML) que corresponde um modelo de linguagem visual para modelar requisitos de sistemas, descrever desenhos e representar detalhes de implementações, que pode ser aplicada na análise de requisitos, análise sistêmica, de projeto, implementação e testes (Keng & Terry, 2000). Um modelo de linguagem pode ser composto por pseudocódigo, código real, imagens, diagramas, ou longas passagens de descrição que ajudam a descrever o sistema. A UML é uma linguagem formal em que cada elemento tem um significado fortemente conhecido e é concisa, ou seja, toda a linguagem é constituída por uma notação simples, direta e abrangente, com a descrição de todos os aspetos importantes do sistema (Miles & Hamilton, 2006).

Foi criada nos anos noventa e sua última versão foi aprovada em 2003, a UML 2. A versão mais recente, UML 2.5 é composta por quatorze diagramas divididos em três categorias: estrutural, comportamental e de interação. Entre os diagramas existentes destacam-se o diagrama de casos de uso, de classes, de sequência, de objetos, de atividades, de componentes, de pacotes e de interação (Chonoles & Schardt, 2011). Neste trabalho foi utilizado o diagrama de casos de uso, onde foram descritas as operações existentes no sistema e as suas funcionalidades, especificado na secção seguinte.

3.4. Etapa 4: Testes e avaliação da ferramenta

A etapa de teste e de avaliação do *software* é essencial para garantir a entrega de bons produtos, ajudar a entender se o projeto atingiu os objetivos esperados pelo cliente e descobrir a necessidade de ajustes (Kan, 2003). Para o cliente ou utilizador final, um sistema com qualidade significa não possuir funcionalidades com erros, ser confiável, seguro e fácil de usar e ter níveis aceitáveis de tolerância a falhas durante o seu uso (Chemuturi, 2010).

A avaliação da ferramenta foi feita por dois utilizadores, um analista de teste e um *product manager* com experiência em métricas ágeis na sua atividade profissional, que participaram no processo de validação dos requisitos finais. O processo de avaliação foi dividido em duas partes: na primeira foi realizada uma reunião individual com cada selecionado, de aproximadamente 30 minutos, para apresentar a ferramenta, relembrar os requisitos e explicar o objetivo da avaliação; na segunda, foram realizados os testes e a avaliação pelos mesmos. Para a realização da avaliação e para a recolha de dados foi realizado um questionário, instrumento de investigação que visa reduzir e facilitar o tempo de recolha e simplificar a análise dos dados (Presser et al., 2004). O questionário foi utilizado para auxiliar e guiar a avaliação e construído de acordo com o modelo de qualidade de produto, definido na ISO/IEC 25010, que estabelece oito características de qualidade. Para o contexto da presente ferramenta foram avaliados cinco desses critérios: 1) Usabilidade - Grau em que um produto ou sistema pode ser usado por utilizadores para atingir objetivos específicos com eficácia, eficiência e satisfação; 2) Eficiência de desempenho - Característica que representa o desempenho em relação à quantidade de recursos utilizados nas condições estabelecidas; 3) Confiabilidade - Grau em que um sistema, produto ou componente executa funções sob condições especificadas por um período de tempo determinado; 4) Segurança - Grau em que um produto ou sistema protege informações e dados para que pessoas, outros produtos ou sistemas tenham acesso apropriado aos seus tipos e níveis de autorização; e 5) Adequação Funcional - Característica que representa o grau em que um produto ou sistema fornece funções que atendem às necessidades declaradas e implícitas quando usado sob condições especificadas (ISO/IEC 25010, 2022). O questionário possui 24 questões e foi desenvolvido tendo como base outros já existentes, como o Ergolist (LABIUTIL - Laboratório de Utilizabilidade da Universidade Federal de Santa Catarina, 2018), o Web Usability Checklist (Patkai, 2022) e o 25-point Website Usability Checklist (Meyers, 2009). Na próxima secção serão apresentados os resultados alcançados.

Capítulo 4 – Análise e discussão dos resultados

4.1. Perfil dos inquiridos do questionário

Com base nos dados recolhidos, foi possível identificar o perfil dos 12 inquiridos que responderam ao questionário (Tabela 1). Todos desempenham cargos diferentes em projetos ágeis de desenvolvimento de *software*, sendo: um Líder de Projetos, um *Product Manager*, um *Infrastructure Consultant*, dois Desenvolvedores, um *Data Analyst*, um *Application Support Engineer*, um Analista Funcional, dois Analistas de Teste, um Analista de Negócios e um *Agile Master*. Os inquiridos têm diferentes graus de escolaridade, seis possuem pós-graduação, um está a frequentar o mestrado, um é detentor de mestrado e quatro possuem licenciatura. Essa diversidade enriqueceu o estudo com distintas respostas que serão apresentadas neste tópico.

Tabela 1 - Perfil dos inquiridos que responderam o questionário

Cargo	Formação (grau de escolaridade)	Funções desempenhadas em projetos ágeis
Analista de negócios	Pós-Graduação	Análise e documentação das necessidades de negócio e requisitos/ Cuidar do *backlog* do produto, auxiliar *Product Owner* e equipa de desenvolvimento / Modelar processos
Application Support Engineer	Estudante de Mestrado	Investigar e solucionar problemas de clientes em ambientes de produção/ gestão e administração de múltiplas aplicações
Analista de Testes	Pós-Graduação	Desenhar cenários de testes / executar testes manuais e automatizados
Product Manager	Pós-Graduação	Colaborar *Realease Management, stakeholder* e equipa para definir e planear os lançamentos (*releases*) / cuidar do *Roadmap* / acompanhar a evolução do produto / definir e acompanhar indicadores do produto
Agile Master	Pós-Graduação	Formar e orientar as equipas de desenvolvimento / moderar as reuniões ágeis: refinamento de *backlog*, planeamento, revisão e retrospetiva / Ajudar as equipas nos problemas / Monitorar os OKRs do produto
Desenvolvedor	Licenciatura	Desenvolver soluções de *software*
Analista Funcional	Pós-Graduação	Apoiar o *Product Owner* na gestão e refinamento do *Backlog* / Gestão do fluxo de entregas / Especificação de requisitos com as áreas de negócio, elaboração de *User Stories* e validação das mesmas com os Clientes / Modelação de processos

Analista de Testes	Licenciatura	Elaboração e execução de testes / Reportar erros
Desenvolvedor	Licenciatura	Desenvolvimento de aplicações
Data analyst	Mestrado	Recolha, tratamento e análise de dados / Construção de relatórios e *dashboards* de apoio ao negócio
Infrastructure Consultant	Licenciatura	Acompanhamento de Auditorias na vertente de infraestruturas / Análise e acompanhamento do Risco de Tecnologias de Informação na vertente de infraestruturas / Criar e adaptar soluções de infraestruturas / Testes de integração e de utilização
Líder de Projetos	Pós-Graduação	Gestão de tarefas e *issues* / Monitorizar o progresso e desempenho e contribuir para a melhoria na equipa / Acompanhamento de processos de projeto e resultados/ acompanhar as métricas da equipa / Organizar e facilitar cerimônias do *Scrum.*

4.2. Métricas ágeis existentes para acompanhar o desempenho do projeto e da equipa de desenvolvimento de *software*

Inicialmente os inquiridos foram questionados quanto às ferramentas utilizadas, quatro inquiridos utilizam somente a ferramenta Jira *Software*[2] para o acompanhamento de projetos e tarefas; um utiliza o Jira juntamente com a ferramenta Microsoft Sharepoint[3]; um inquirido utiliza uma ferramenta desenvolvida internamente chamada AgileWork[4]; um a ferramenta Broadcom *Software*[5]; outro utiliza o Microsoft Planner[6]; um a plataforma GitLab[7]; um utiliza o Microsoft Excel juntamente com o Power BI[8]; um utiliza o Kanbanize[9]; e, por fim, um inquirido utiliza a ferramenta Sharepoint. A maioria

[2] Jira *Software*: desenvolvido pela Atlassian para suportar a colaboração em equipas ágeis. https://www.atlassian.com/software/jira
[3] Microsoft Sharepoint: é uma plataforma desenvolvida pela Microsoft para auxiliar na gestão de informações, projetos, fluxos de trabalho e equipas. https://www.microsoft.com/pt-pt/microsoft-365/sharepoint/collaboration
[4] AgileWork: ferramenta desenvolvida internamente pela empresa do participante para gestão de projetos e tarefas.
[5] Broadcom *Software*: desenvolvido pela Broadcom CA para atender principalmente aos negócios de segurança cibernética e infraestrutura de *software*. https://software.broadcom.com/
[6] Microsoft Planner: é uma solução desenvolvida pela Microsoft para gerir e organizar projetos, criar planos, atribuir tarefas e partilhar ficheiros. https://www.microsoft.com/pt-pt/microsoft-365/business/task-management-software
[7] Plataforma GitLab: desenvolvida pela GitLab Inc. para hospedar repositórios e otimizar o fluxo de trabalho de desenvolvimento de *software*. https://about.gitlab.com/company/
[8] Power BI: desenvolvido pela Microsoft para visualização de dados. https://powerbi.microsoft.com/
[9] Kanbanize: plataforma para gestão ágil de projetos e fluxos de trabalho desenvolvida pela empresa Kanbanize. https://kanbanize.com/

dos inquiridos (*n*=11) considerou quase todas as ferramentas de fácil uso, mas apenas nove dos inquiridos as consideram úteis para a sua equipa no dia-a-dia, manifestando o restante a opinião oposta de que a ferramenta não é útil para a sua equipa no dia-a-dia.

Para conhecer o cenário das métricas ágeis adotadas nas empresas, os inquiridos foram questionados sobre quais as métricas que utilizam na equipa para acompanhar o desempenho do projeto e da equipa de desenvolvimento de *software*. Os resultados mostraram que sete dos inquiridos referiram utilizar a *Lead Time* e a *Cycle Time*, cinco referiram a WIP, a *BurnDown* e a *BurnUp*, e quatro a *Throughput* e a CFD.

A maioria dos inquiridos (*n*=9) referiu que as métricas que utilizam são visualizadas na própria ferramenta. Os restantes referem a visualização no Microsoft Excel, no Power BI ou, até mesmo, a não utilização de ferramenta para a visualização. Contudo, apesar de grande parte dos inquiridos (*n*=7) ter considerado que as métricas disponíveis não são suficientes para realizar o planeamento, estimativas e acompanhar o projeto e as tarefas da equipa, metade considerou que a ferramenta possui todos os dados necessários para a exibição das métricas mais relevantes. A Tabela 2 apresenta as informações detalhadas.

Tabela 2 - Resumo métricas ágeis existentes para acompanhar o desempenho do projeto e da equipa de desenvolvimento de *software*

Métricas Ágeis	**N=12**	**Ferramenta**	**N=12**
Métricas usadas na empresa		**Onde são visualizadas as métricas**	
Lead Time	58,3%	Na ferramenta	75%
Cycle Time	58,3%	Não usam ferramenta	8,3%
Work In Progress	41,7%	Excel	8,3%
Burdown e Burn-Up	41,7%	Power BI	8,3%
Throughput	33,3%		
CFD	33,3%	**Ferramenta utilizada fornece informações suficientes**	
OKR	16,7%	Sim	41,7%
Estimativa de Monte Carlo	16,7%	Não	58,3%
Work Item Age	8,3%		
Excel	8,3%		
Eficiência de fluxo	8,3%	**Ferramenta utilizada possui todas as informações**	
Nenhuma	8,3%	Sim	50%
		Não	50%

4.3. Métricas ágeis concebidas a partir da plataforma GitHub

Visto que a maioria dos inquiridos não consideram suficientes as métricas disponíveis na ferramenta para auxiliar no planeamento, estimativas e acompanhamento do projeto e das tarefas, procurámos saber quais as métricas ágeis que são importantes para utilizar da equipa. Os inquiridos puderam escolher mais do que uma opção e os resultados mostraram que a WIP foi a métrica considerada mais importante (*n*=9), seguida da *Lead Time* e da *Throughput* (*n*=8), da *Cycle Time (n=7)* e, por fim, da *Cumulative Flow Diagram* (CFD, *n*=6). A Tabela 3 mostra, em detalhe, as respostas.

Tabela 3 - Resumo métricas ágeis importantes

Métricas importantes para utilizar na equipa			
	n=12		**n=12**
Work In Progress	75%	**OKR**	41,7%
Lead Time	66,7%	**Estimativa de Monte Carlo**	25%
Throughput	66,7%	**Nenhuma**	16,7%
Cycle Time	58,3%	**Somente OKR**	8,3%
CFD	50%	**Eficiência de Fluxo**	8,3%
Burdown e Burn-Up	41,7%	**Work In Age**	8,3%

4.4. Dados recolhidos da plataforma GitHub e necessários para as métricas ágeis

A fim de obter informações para as métricas ágeis definidas, foi utilizada a *API Rest* GitHub, através de método GET de requisição. Para cada métrica foram utilizadas as URLs descritas na Tabela 4.

Tabela 4 - Dados necessários para as métricas

Métricas Ágeis	**URL**
***Work In Progress* (WIP)**	
Para recolha de todas as colunas do quadro da equipa	*https://api.github.com/projects/<id projeto>/columns*
Para recolha dos cards (***issues***) de cada coluna	*https://api.github.com/projects/columns/<id coluna>/cards*
Diagrama de Fluxo Cumulativo (CFD)	
Para recolha dos cards (***issues***) de cada coluna	*https://api.github.com/projects/columns/<id coluna>/cards*
Para recolha dos dados da ***timeline*** de cada ***issue***	*https://api.github.com/repos/<user GitHub>/<projeto>/issues/<id issue>/timeline*

Troughput	
Para recolha das ***issues*** fechadas	*https://api.github.com/repos/<user GitHub>/<projeto>/issues?state=closed*
Lead Time	
Para recolha dos cards (***issues***) de cada coluna	*https://api.github.com/projects/columns/<id coluna>/cards*
Para recolha dos dados da ***timeline*** de cada ***issue***	*https://api.github.com/repos/<user GitHub>/<projeto>/issues/<id issue>/timeline*
Cycle Time	
Para recolha dos cards (***issues***) de cada coluna	*https://api.github.com/projects/columns/<id coluna>/cards*
Para recolha dos dados da ***timeline*** de cada ***issue***	*https://api.github.com/repos/<user GitHub>/<projeto>/issues/<id issue>/timeline*

4.5. Ferramenta para visualização das métricas recolhidas do GitHub

Para identificar as funcionalidades da ferramenta de visualização de métricas ágeis, os inquiridos foram questionados sobre os pontos positivos e negativos da ferramenta atualmente usada, conforme Tabela 5.

Tabela 5 - Pontos positivos e negativos da ferramenta

Pontos positivos da ferramenta	Pontos negativos da ferramenta
Visualização fácil.	**Dificuldade** na programação.
Tarefa a realizar e ponto de situação.	Não ter **personal data**.
É boa apenas para o registro das horas trabalhadas.	Precisa ser muito bem customizada, pois traz muitas **informações irrelevantes**.
Acompanhamento das tarefas/projetos, exportação dos dados, geração de gráficos, automatização, customização de campos.	Por ser uma **ferramenta complexa**, a curva de aprendizado é grande; existem muitas configurações por vários caminhos, então é necessário conhecer bem o Jira.
Facilidade de uso.	Nenhum... talvez a **métrica Scrum** poderia ter.
A usabilidade da ferramenta é boa, possui várias informações que a equipe necessita, permite pesquisar por várias informações e a ferramenta armazena várias informações importantes.	Sinto um pouco de **dificuldade** em acompanhar os projetos e também podia ter mais gráficos, permitindo **fácil** visualização e **análise dos projetos**.
Tudo acessível.	**Muitas subtarefas**.
Objetiva, fácil de utilizar (friendly), rápida/leve e completa.	Não vejo atualmente.
Gestão de IN e CH.	Muitos **bugs**.
Métricas diversas.	Pouco **caro** para grandes empresas.

Posteriormente foi solicitado que identificassem as características importantes para uma ferramenta de acompanhamento de projetos e tarefas. O resumo dessas respostas encontra-se na Tabela 6.

Tabela 6 – Característica ferramenta para visualização de métricas

Características importantes de uma ferramenta
Facilidade no manuseio.
Deve ser dividida por estado de processo, quem está envolvido, duração, se é prioritária ou não.
Uma visão de estimativa de tempo, com possibilidade de acompanhamento em tempo real do andamento das tarefas.
A ferramenta deve ser adaptável a qualquer processo de trabalho, e permitir extrair de forma clara as informações inseridas nela. Também deve permitir gerenciar os projetos até ao nível de tarefa. Deve gerar relatórios com as métricas mais utilizadas do mercado (citadas aqui no questionário) sendo capaz de fornecer excelentes relatórios.
Deve ser de fácil uso, design atraente e com recursos de métricas facilitados.
Informações corretas e atualizadas, conter as informações necessárias para o acompanhamento, gráficos que facilitem a análise e visualização e ser uma ferramenta fácil de usar com a possibilidade de introduzir filtros e pesquisas.
Melhor organizado e com menos subtarefas.
Friendly, ter apenas o que é realmente necessário, ser desenvolvida a pensar em quem vai realmente utilizar e a ter como base o mindset agile.
Ser intuitiva.
Fácil importação de dados de outras ferramentas. Filtros diversos, customização na visualização.

4.5.1. Resultado da análise das respostas do questionário

Após conhecer as ferramentas utilizadas por cada inquirido que respondeu ao questionário, constatou-se que o principal problema é a falta de informações para auxiliar no planeamento, estimativas e acompanhamento dos projetos e tarefas, pois mais de metade não possui dados suficientes, não exibe estimativas claras, gráficos de projetos e algumas métricas necessárias. Outros problemas identificados foram a complexidade, já que é necessário realizar demasiadas configurações; a exibição de dados irrisórios e desalinhados, o que dificulta o entendimento e análise destes; e, por último, a demora nas respostas pela ferramenta, principalmente, no cálculo e construção de métricas e gráficos. Por estes motivos, quase metade dos inquiridos opta por outros meios e ferramentas para suprir as suas necessidades, colocando de parte a ferramenta oficial.

Dada a importância que as ferramentas têm para tomadas de decisões mais assertivas pelas empresas, torna-se cada vez mais necessário o seu uso para a

monitorização, análise e visualização de dados e métricas de maneira rápida, simples, clara e objetiva, o que permite aperfeiçoar estimativas e planeamentos, monitorizar o desempenho da equipa, identificar problemas e otimizar fluxos de trabalho. Diante destas necessidades, os utilizadores procuram uma ferramenta que apresente métricas ágeis relevantes para acompanhar as tarefas, os projetos e o progresso da equipa. Para tal, a ferramenta deverá ser intuitiva, organizada e de fácil uso, adaptável a qualquer processo de trabalho, com visualização rápida, permitindo uma análise dos dados em tempo real e a extração clara das informações inseridas. Deverá, acima de tudo, ser uma ferramenta baseada no *mindset* ágil, ou seja, flexível, dinâmica, simples e que contribua para promover a melhoria contínua.

4.5.2. Resultado da sessão de brainstorming

Diante dos resultados e entendimento das necessidades dos inquiridos, foi agendada uma sessão de *brainstorming* para fomentar a colaboração de ideias e a validação e definição dos requisitos finais da ferramenta. A sessão teve a duração de aproximadamente duas horas, foi dividida em duas partes e realizada a cinco dos 12 inquiridos que responderam ao questionário inicial, uma vez que a sua prática profissional e experiência em métricas ágeis se revelou particularmente relevante para o presente estudo. Na primeira parte, com duração de uma hora e vinte minutos, os inquiridos tiveram liberdade para expor as suas ideias e sugestões e debater sobre as demais contribuições para a ferramenta de visualização de métricas ágeis. Na segunda parte, cada participante selecionou três ideias que julgava importante, as melhores ideias, consideradas por todos presentes, foram priorizadas para definir os requisitos finais. Para auxiliar a sessão de *brainstorming* foi utilizada a ferramenta gratuita Miro, que tornou mais rápida, fácil e transparente a colaboração de ideias. O resultado da sessão pode ser visto no Apêndice F. As características consideradas mais importantes para uma ferramenta de acompanhamento de projetos e tarefas em equipas ágeis de desenvolvimento de *software* são apresentadas na Tabela 7.

Tabela 7 – Principais características para a ferramenta

Característica	Descrição
Intuitivo	Facilidade dos utilizadores em manusear o sistema e aproveitar o máximo de suas funcionalidades.

Flexível	Permitir visualizar diferentes projetos, filtrar por diferentes períodos e trabalhar com diferentes dados.
Transparente	O sistema deve prezar pela transparência para que todos possam consultar os resultados.
Métricas relevantes	Exibir somente as métricas relevantes para auxiliar no acompanhamento de tarefas, projetos e tomadas de decisões.
Informações atualizadas	Consultar os dados em tempo real.
Confiável	As informações e os cálculos das métricas devem ser corretas.
Segura	O sistema deve permitir aos utilizadores autenticados visualizar somente as informações que podem ter acesso.

O resultado desta sessão resultou na listagem de requisitos funcionais, detalhados no próximo tópico.

4.6. Especificação de Requisitos

A especificação de requisitos é uma etapa crítica para o sucesso de um sistema, já que define os objetivos e as funções que precisam de ser realizadas, o que deve ser construído e as suas restrições. É nesta etapa que se compreende exatamente o que deve ser desenvolvido e o resultado que se espera (Sage & Rouse, 2014). Os requisitos podem ser classificados em funcionais e não funcionais, definidos de seguida. Neste tópico será detalhado o que foi validado na etapa anterior com alguns utilizadores.

4.6.1. Requisitos Funcionais

Os Requisitos Funcionais (RF) definem essencialmente as funcionalidades de um sistema ou de um subsistema e impõem restrições na sua conceção e/ou implementação. Os RF têm um impacto significativo na arquitetura do sistema de *software*, são voláteis e, se modificados, podem envolver esforços dispendiosos por implicarem alterações no sistema. Como tal, é de extrema importância que sejam escritos de forma clara e objetiva para que todos entendam o pretendido (Chatterjee et al., 2020). A Tabela 8 apresenta os requisitos funcionais determinados e que deverão existir na ferramenta para atender às necessidades dos utilizadores.

Tabela 8 - Requisitos Funcionais

RF#	Funcionalidade
RF01	O sistema deve ser configurado por utilizador

RF02	O sistema deve validar o utilizador por *token* de acesso
RF03	O sistema deve exibir os projetos vinculados ao utilizador e permitir selecionar o projeto para visualização das métricas
RF04	O sistema deve permitir visualizar os gráficos por dia, semana ou mês e filtrar por período
RF05	O sistema deve exibir a métrica Wip (*Work In Progress*), que consiste na quantidade de tarefas em progresso dentro do período informado
RF06	O sistema deve exibir a quantidade de tarefas para cada fase do projeto dentro do período informado
RF07	O sistema deve exibir a métrica *Troughput*, ou seja, a quantidade de tarefas concluídas dentro do período informado
RF08	O sistema deve exibir o tempo que uma tarefa levou para ser desenvolvida, ou seja, o tempo que ela ficou "em progresso" até sua conclusão (Métrica *Cycle Time*), de acordo com o período informado
RF09	O sistema deve exibir o *Lead Time* das tarefas, ou seja, o tempo de abertura até a conclusão da tarefa
RF10	O *Lead Time* médio das tarefas deve ser exibido, dentro do período selecionado
RF11	O sistema deve apresentar um gráfico de Fluxo Cumulativo (CFD) das tarefas que passam pelo fluxo de trabalho e distinguir as etapas

4.6.2. Requisitos Não Funcionais

Após mais de 40 anos, os requisitos não funcionais (RNF) continuam a ser um desafio. O trabalho tem-se concentrado em como considerar a qualidade e RNF de forma eficaz como parte do desenvolvimento de *software*. São definidos como um atributo ou uma restrição do sistema e não estão diretamente ligados às funcionalidades, ou seja, enquanto o RF descreve o que deve ser feito, o RNF descreve como deve ser feito (Habibullah & Horkoff, 2021). A Tabela 9 mostra os RNF definidos para a ferramenta.

Tabela 9 - Requisitos Não Funcionais

RFN#	Funcionalidade
RFN01	O sistema deve ser *web*.
RFN02	O sistema deve ser compatível com o navegador Google Chrome
RFN03	O sistema deve ser integrado à plataforma GitHub, utilizando o GitHub Rest API v3.
RFN04	O sistema deve ser desenvolvido em Java e utilizar o *framework* Angular
RFN05	O sistema deve recolher as informações do GitHub em tempo real

4.7. Modelação da Ferramenta

Com base na análise das repostas do questionário e na especificação dos requisitos, esta ferramenta deve exibir as métricas *Lead Time*, *Cycle Time*, *Throughput*, WIP e CFD e permitir selecionar o projeto, o período e a forma de visualização dos gráficos (conforme

o dia, semana ou mês). Ainda, o utilizador poderá configurar as colunas que correspondem às tarefas para fazer, em estado de progresso e concluídas. Será de seguida explicada detalhadamente a construção da ferramenta.

4.7.1. ***Diagrama de caso de uso***

Os casos de uso revelaram-se particularmente valiosos como parte das atividades de requisitos do processo do *software*. Melhoraram consideravelmente a comunicação entre as equipas de desenvolvimento e as partes interessadas e tornaram a determinação dos requisitos muito mais fácil e precisa. Os casos de uso são únicos na sua capacidade de ajudar as equipas a compreender o valor que o sistema deve proporcionar aos seus intervenientes (Bittner & Spence, 2003).

Um diagrama de caso de uso representa um sistema em termos de interação do utilizador, mostra cenários de utilização do sistema, a sequência de eventos, e pode também representar um outro sistema ou um *hardware* (B'Far, 2004). É tipicamente constituído por um ator, um caso de uso e as suas relações. Deve ser primeiramente definido o(s) ator(es), que pode ser uma pessoa, um outro sistema ou até mesmo um dispositivo de *hardware*, representando sempre um conjunto de papéis que interagem com o sistema a partir do exterior. Com os atores definidos, são listadas todas as interações que podem acontecer, cada uma delas é chamada de caso de uso e representa um comportamento específico do sistema (Bittner & Spence, 2003).

A Figura 14 apresenta o diagrama de caso de uso da ferramenta proposta neste trabalho, a qual representa os atores, casos de uso e a relação entre eles. Os casos de uso serão apresentados em detalhe no próximo tópico.

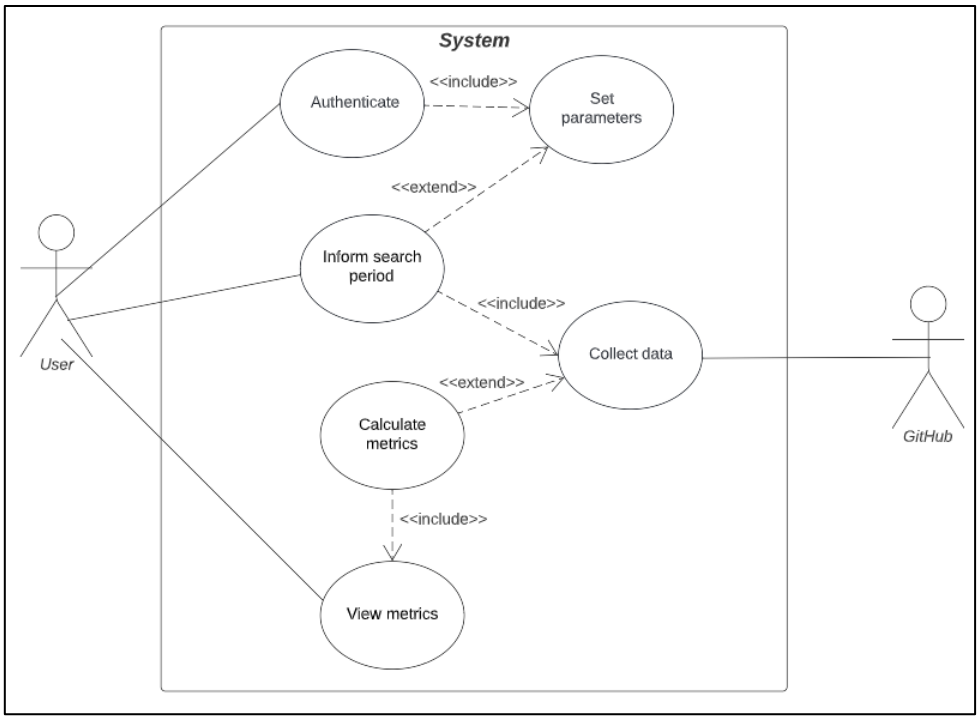


Figura 14 - Diagrama caso de uso

4.7.2. Casos de uso detalhado

Conforme o diagrama, a ferramenta é composta por dois atores: *User* e *GitHub* e seis casos de uso. O ator *User* (utilizador) é qualquer utilizador da ferramenta que irá visualizar as métricas. O ator GitHub representa a plataforma GitHub, sistema externo, para recolher os dados necessários. Os casos de uso representados são:

UC01 – *Authenticate*: a ferramenta deve autenticar o utilizador da plataforma GitHub, será necessário informar *user name* e *token* do GitHub.

UC02 – *Set parameters*: a ferramenta deve permitir parametrizar as informações necessárias para o correto funcionamento. Com as credencias válidas, o utilizador seleciona o repositório e o projeto do qual deseja visualizar as métricas, a frequência de visualização (dia, semana ou mês) e identifica qual das colunas do quadro corresponde às tarefas que estão para fazer e às tarefas em progresso e define a ordem das colunas.

UC03 - *Inform search period*: a ferramenta deve permitir ao utilizador informar o período de visualização das métricas (informar data de início e de fim).

UC04 - *Collect data*: a ferramenta irá recolher na plataforma GitHub (sistema externo) os dados necessários para o cálculo das métricas do projeto selecionados pelo utilizador autenticado.

UC05 – *Calculate metrics*: a ferramenta deve calcular as métricas *Lead Time, Cycle Time, Cumulative Flow Diagram, Work in Progress* e *Throughput* a partir dos dados recolhidos da plataforma GitHub.

UC06 – *View metrics*: o utilizador pode visualizar o resultado das métricas calculadas.

4.8. Arquitetura geral da ferramenta

A ferramenta é uma aplicação *web* construída na sua totalidade com as tecnologias *open source*. No *front-end*, para a construção da interface de aplicação com o utilizador, utiliza o *framework* Angular baseado em JavaScrypt. No *back-end* utiliza a linguagem de programação Java juntamente com Spring Boot (*framework* Java) que facilita e otimiza o tempo de desenvolvimento de aplicações Java.

A comunicação com a plataforma GitHub para a recolha de dados é realizada através da *API Rest* v3 do GitHub. Quando o utilizador solicita a criação das métricas, é feita uma requisição à API que retorna os dados para uma estrutura JSON, a ferramenta interpreta e trata esses dados para o cálculo e exibição das informações em tempo real.

Para desenvolver o *front-end* da ferramenta foi utilizado o Visual Studio Code e para o *back-end* o IDE Eclipse. A análise e teste da *API Rest* do GitHub foi realizada com o auxílio da aplicação Postman que permite executar requisições em qualquer API. Essas tecnologias foram escolhidas pela familiaridade e conhecimento em utilizá-las. A Figura 15 ilustra a arquitetura geral da ferramenta.

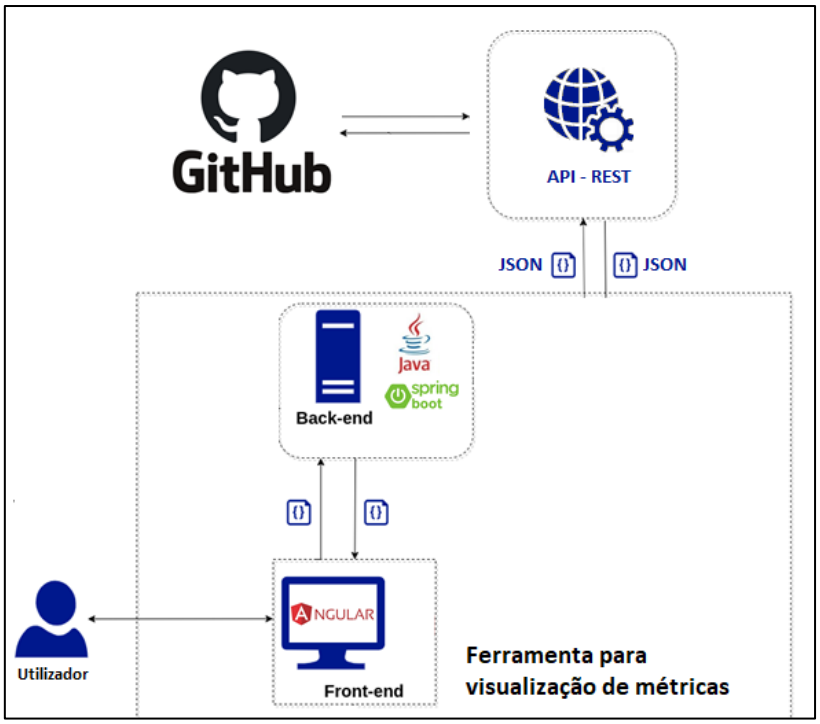


Figura 15 - Arquitetura geral ferramenta

4.9. Protótipos

Neste tópico será apresentado o fluxo de utilização da ferramenta com imagens das funcionalidades desenvolvidas.

1) Informar *user name* e *token* para autenticação na ferramenta (Figura 16).

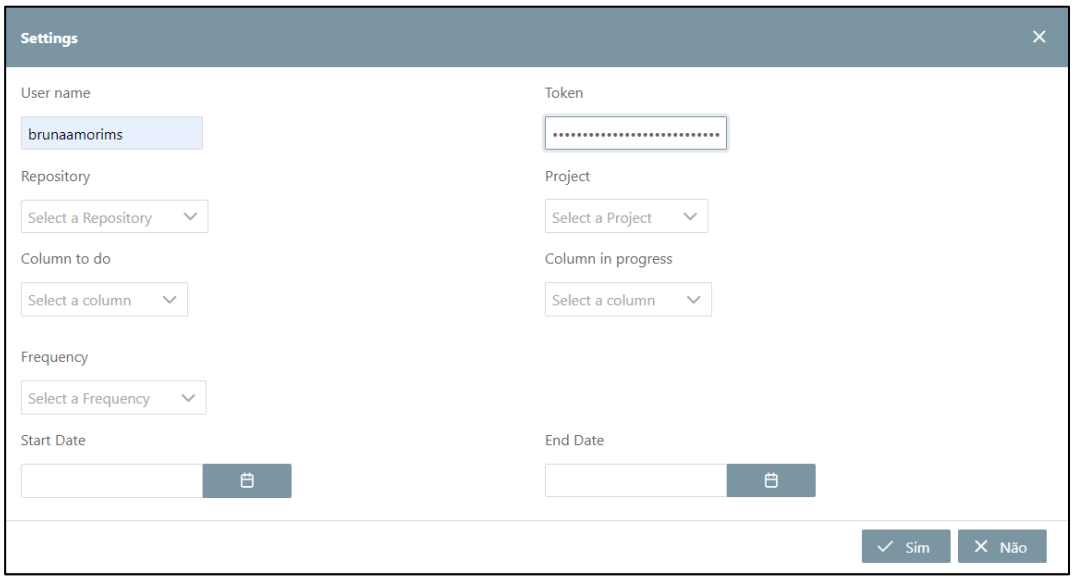


Figura 16 - Ecrã "Settings" – Autenticar utilizador GitHub

2) Após a autenticação do utilizador, este vai selecionar o repositório e o projeto do qual deseja visualizar as métricas, a frequência de visualização (dia, semana ou mês), informar a coluna do quadro que corresponde às tarefas para fazer e às tarefas em progresso e indicar a ordem das colunas (coluna tarefas finalizadas/ coluna tarefa em progresso/ coluna tarefas para fazer; Figura 17).

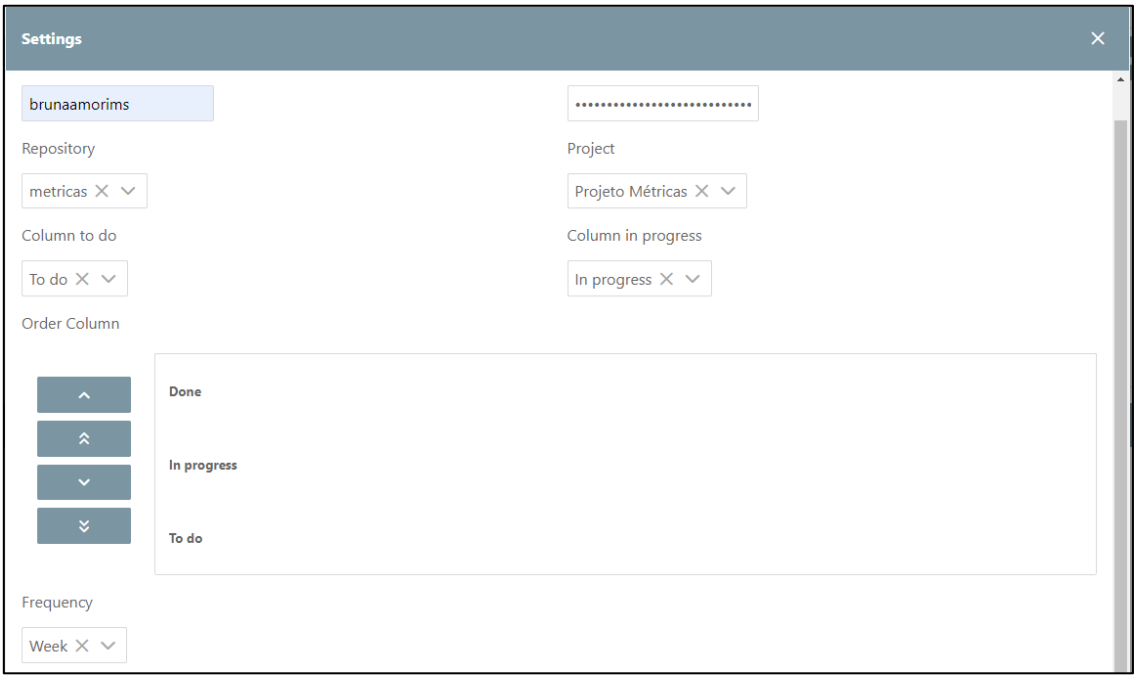


Figura 17 - Ecrã "Settings" – Parametrizar ferramenta

3) De seguida vai informar o período para a visualização das métricas, ou seja, a data de início e fim para pesquisar as tarefas (*issues*), como mostra a Figura 18.

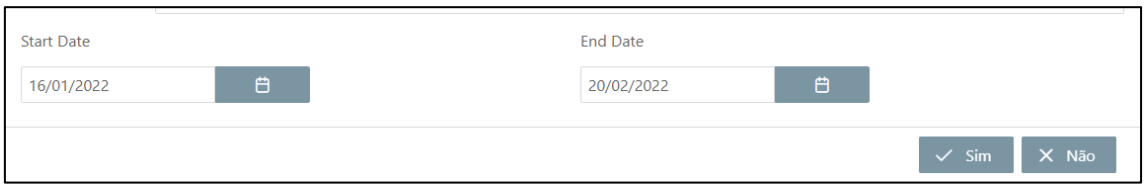


Figura 18 - Ecrã "Settings" – Informar data início e fim

4) Com todas as informações preenchidas na opção de "*Settings*", a ferramenta irá calcular e exibir as métricas *Lead Time*, *Cycle Time*, *Cumulative Flow Diagram*, *Work in Progress* e *Throughput*. A Figura 19 mostra o painel de exibição de todas as métricas.

Figura 19 - Ferramenta a exibir todas as métricas

O gráfico "*Issue x Column*" (Figura 20) apresenta a quantidade de tarefas (*issues*) em cada coluna do quadro do projeto. Nele podemos visualizar a quantidade de *issues* em progresso através da métrica WIP (*Work In Progress*).

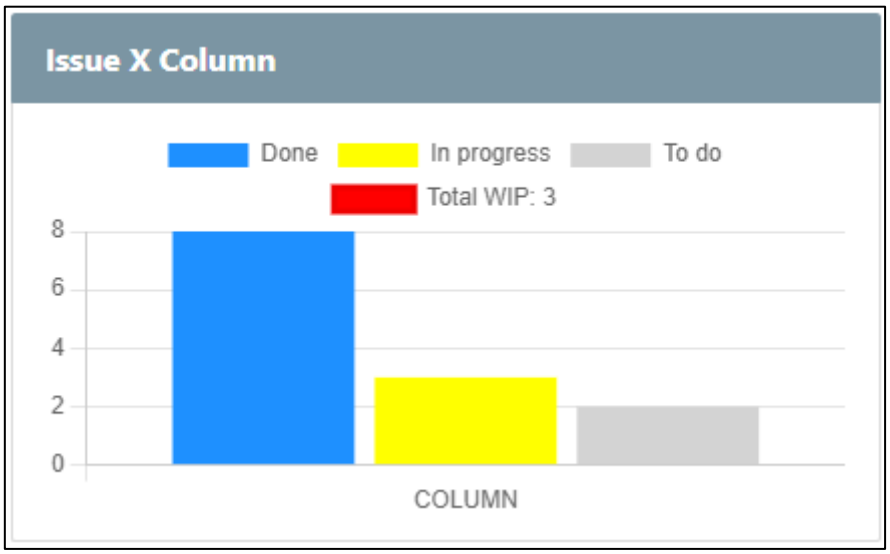


Figura 20 - Gráfico Issue x Column – Métrica WIP

O gráfico CFD representa graficamente a evolução do trabalho através do diagrama de fluxo cumulativo, como pode ser observado na Figura 21.

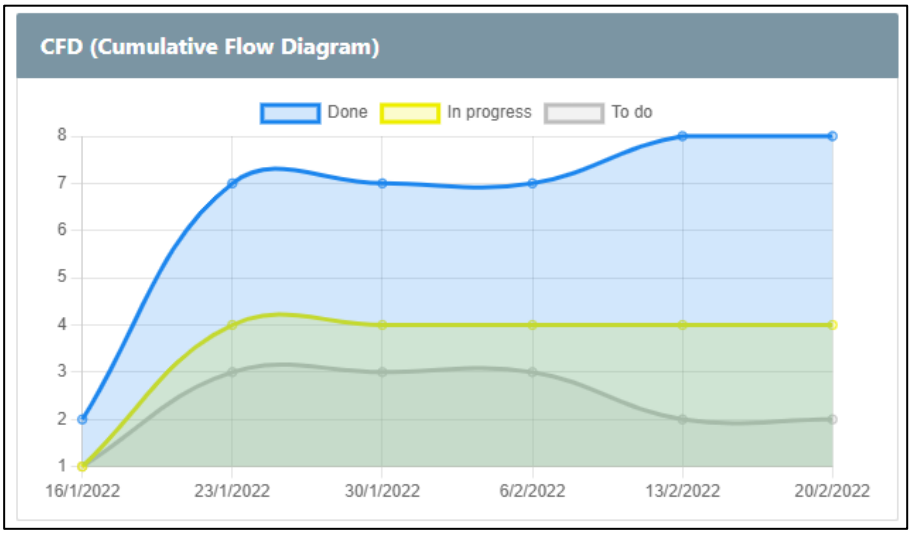


Figura 21 - Gráfico CFD (Cumulative Flow Diagram) – Métrica diagrama de fluxo cumulativo

O gráfico *Throughput* indica a quantidade de itens entregues num determinado período (Figura 22).

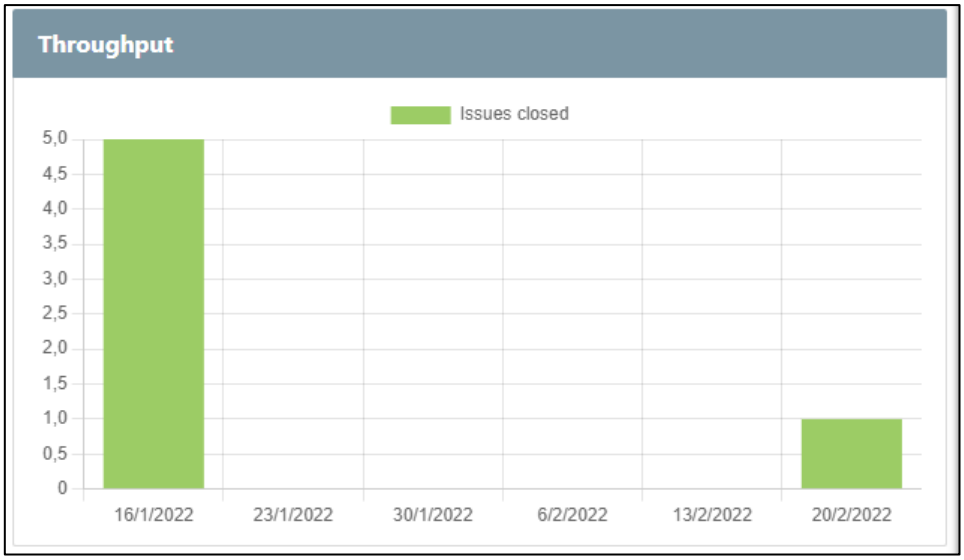


Figura 22 - Gráfico Throughput – Métrica Troughput

O gráfico *Lead Time* indica o tempo médio que um pedido demora a percorrer todo o processo, do início ao fim. O gráfico também mostra o *Lead Time* médio das *issues* (Figura 23).

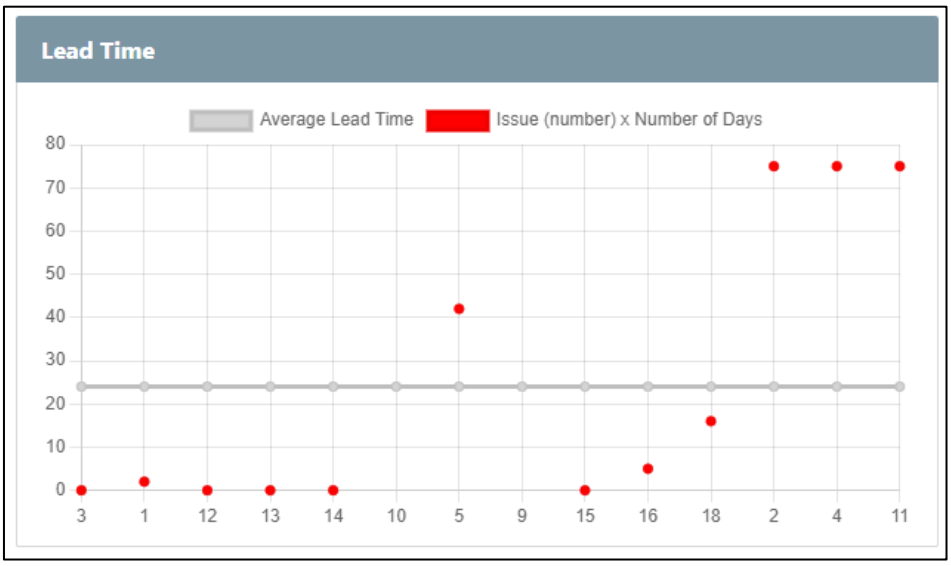


Figura 23 - Gráfico Lead Time – Métrica Lead Time

Por último, o gráfico *Cycle Time* exibe o total de dias desde o início do desenvolvimento de uma *issue* até à sua conclusão (Figura 24).

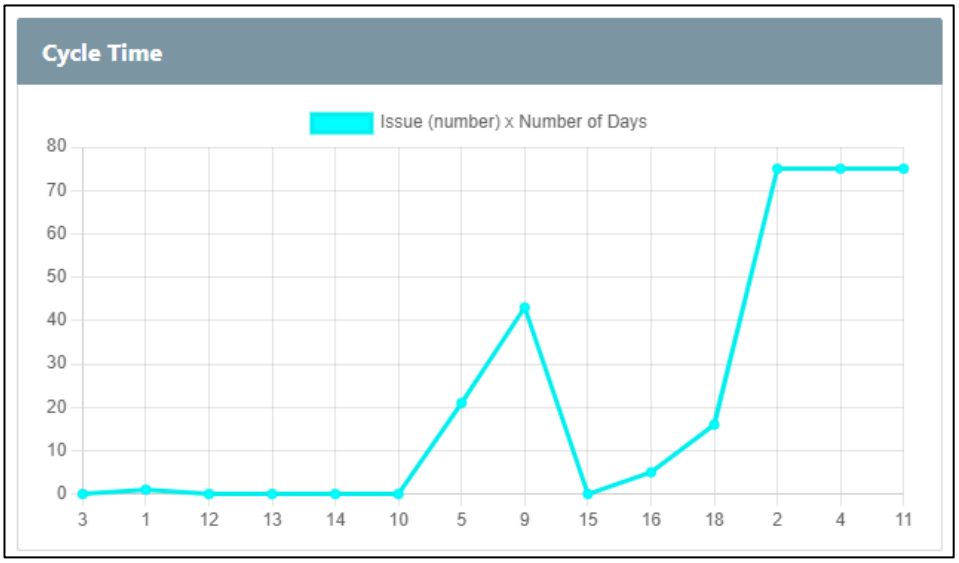


Figura 24 - Gráfico Cycle Time – Métrica Cycle Time

4.10. Resultados do Teste e Avaliação da ferramenta

Com o objetivo de auxiliar e guiar a avaliação, foi aplicado um questionário composto por 24 questões, com três perguntas sobre informações pessoais, cinco para avaliar o critério de usabilidade, três o critério de eficiência de desempenho, duas a confiabilidade da ferramenta, duas referentes à segurança, oito para avaliar a adequação

funcional e uma pergunta aberta para comentários ou sugestões. Todas as perguntas para avaliar as características de qualidade são fechadas e as respostas foram dadas numa escala de *Likert* de cinco pontos (1 - Discordo Totalmente, 5 - Concordo Totalmente).

O primeiro critério de qualidade avaliado foi a usabilidade, que permite perceber o grau de facilidade e sucesso com que os utilizadores (U) lidam com a ferramenta e executam as tarefas (Sarrab & Rehman, 2014). A média obtida foi 4.2 pontos, ou seja, foi considerada uma ferramenta simples, fácil de usar, com gráficos de fácil visualização e que agrega valor ao utilizador. No critério eficiência de desempenho, definido como a capacidade do sistema em alcançar respostas no tempo apropriado (Kroeger et al., 2014), a média da avaliação foi 4.8 pontos, tendo os utilizadores considerado que a ferramenta apresenta os resultados no tempo apropriado. A confiabilidade do sistema, que corresponde à taxa de falhas ao executar funções num período de tempo específico, teve uma média de 5 pontos, não apresentando erros inesperados durantes os testes. A segurança, critério que corresponde ao controle no acesso de dados e informações e a garantia da integridade do sistema (Sarrab & Rehman, 2014), apresentou uma média de 5 pontos, já que, como esperado, somente utilizadores autenticados podem visualizar informações dos seus respetivos repositórios e projetos. Por último, a adequação funcional, que verifica se as funcionalidades do sistema satisfazem as necessidades e requisitos esperados pelos utilizadores (Kroeger et al., 2014), apresentou uma média de 4.7 pontos, permitindo ao utilizador planear, estimar e acompanhar projetos e tarefas através da visualização de métricas ágeis de desenvolvimento de software. As médias das avaliações estão descritas na Tabela 10.

Tabela 10 - Resumo critérios de qualidade avaliados

Critérios de qualidade	U1	U2	Média
Usabilidade	4.4	4	*4.2*
Eficiência de desempenho	5	4.7	*4.8*
Confiabilidade	5	5	*5*
Segurança	5	5	*5*
Adequação funcional	4.7	4.6	*4.7*
Média Total	**4.82**	**4.66**	***4.74***

Com base nas respostas das avaliações é possível concluir que os utilizadores consideraram a ferramenta simples, útil e prática para equipas ágeis de desenvolvimento de *software*. Os gráficos exibidos foram de fácil visualização e análise, com informações confiáveis e o tempo de carregamento considerado adequado. A ferramenta foi considerada segura e as métricas disponíveis foram avaliadas como suficientes para realizar o planeamento, estimativas, acompanhar projetos e tarefas, verificar gargalos e instabilidades no fluxo de trabalho, atividades importantes no dia-a-dia das equipas.

Capítulo 5 – Conclusões e Recomendações

5.1. Principais conclusões

Este trabalho teve como finalidade a análise e o desenvolvimento de uma ferramenta para a visualização de métricas ágeis de acompanhamento de projetos e tarefas de desenvolvimento de *software* através da recolha de dados da plataforma GitHub.

Para alcançar o objetivo geral do trabalho e responder à pergunta "*Que métricas são consideradas relevantes para medir e avaliar a eficiência do modelo de trabalho em equipas ágeis de desenvolvimento de software?*", foram definidos e cumpridos os seguintes objetivos específicos

1. Identificar e compreender quais as métricas ágeis existentes para acompanhar o desempenho do projeto e das equipas de desenvolvimento de *software*.

Realizou-se uma revisão de literatura acerca deste tema, verificando-se que existem vários tipos de métricas que se aplicam a diferentes necessidades, como para medir o código, a qualidade do produto, métricas de teste, de desempenho de execução, para acompanhar a evolução de projetos e tarefas, planear e priorizar tarefas, entre outros. A equipa ou empresa, primeiramente, precisa de entender o seu problema e definir o objetivo que pretende alcançar e, a seguir, escolher as métricas mais adequadas e relevantes.

2. Identificar quais as métricas ágeis que podem ser concebidas a partir da plataforma GitHub para acompanhar o desempenho do projeto e das equipas.

Das métricas identificadas na revisão de literatura, identificaram-se as seguintes para acompanhar os projetos e tarefas das equipas de desenvolvimento de *software*: a *lead time*, que indica o tempo total de entrega de uma tarefa, considerando a data da sua criação até à data da sua entrega; o *cycle time*, que exibe a quantidade de tempo de desenvolvimento de uma tarefa, considerando a data em que iniciou o desenvolvimento até à data da entrega; o WIP (*Work In Progress*), que exibe a quantidade de trabalho em progresso; o *throughput*, que indica a quantidade de tarefas entregues e a métrica *Cumulative Flow Diagram* (CFD), que representa graficamente a evolução do trabalho, exibe obstáculos e alerta sobre possíveis instabilidades.

Com as métricas definidas, foi verificado que o GitHub possui todos os dados necessários para os cálculos. O GitHub é uma plataforma social utilizada por muitas equipes ágeis para o desenvolvimento de *software* colaborativo e que possui outros benefícios além de armazenar repositórios, como visualizar o progresso da equipa, acompanhar os trabalhos em progresso, gerenciar os projetos, os problemas e as alterações. Pela plataforma podemos ter conhecimento do andamento das *issues* (tarefas) através de um quadro Kanban, da quantidade de tarefas em cada etapa do quadro e da *timeline* de cada tarefa, bem como da sua data de criação, conclusão e data em que esteve em cada etapa. Com essas informações foi possível calcular as métricas de acompanhamento de projetos e tarefas de desenvolvimento de *software*.

3. Desenvolver uma ferramenta para a visualização dos resultados das métricas recolhidas da plataforma GitHub;

Para desenvolver a ferramenta, foi feito inicialmente o levantamento de requisitos através das seguintes etapas: na primeira foi realizado um questionário *online* para recolher informações de pessoas com conhecimento e experiência em projetos ágeis de desenvolvimento de *software* (cf. Apêndice B); na segunda etapa foi feita a análise das respostas (cf. Apêndice C) e definida uma lista inicial de requisitos; e na última etapa realizou-se uma sessão de *brainstorming* para colaboração de ideias, discussão sobre as necessidades e problemas existentes e validação e definição de requisitos funcionais para a ferramenta. Foi identificado que os principais problemas existentes nas ferramentas de acompanhamento de projetos e tarefas são a falta de informações e informações irrisórias e erradas, o que dificulta a análise e a interpretação dos dados. Dada a importância do seu uso na tomada de decisões mais assertivas pelas empresas, observou-se que os utilizadores procuram uma ferramenta intuitiva, ou seja, fácil de manusear e que permita aproveitar ao máximo as suas funcionalidades; flexível, permitindo visualizar diferentes projetos, filtrar por diferentes períodos e trabalhar com diferentes dados; transparente, ou seja, que preze pela transparência para que todos possam consultar os resultados; com métricas relevantes, exibindo somente as métricas relevantes para auxiliar o acompanhamento de tarefas, projetos e tomadas de decisões; com informações atualizadas para consultar os dados em tempo real; e confiável, com as informações e os cálculos das métricas corretos. Com a aquisição deste conhecimento e o entendimento das necessidades, foram validados os requisitos funcionais e não funcionais para a ferramenta.

Após a definição dos requisitos, foi possível realizar a modelação da ferramenta através do diagrama de casos de uso, indicando graficamente como diferentes utilizadores interagem com o sistema e detalhando as funcionalidades existentes. A arquitetura geral da ferramenta também foi determinada e ficou definido que a comunicação com a plataforma GitHub para a recolha de dados seria feita através da *API Rest* v3 e as tecnologias Java e Angular seriam utilizadas para o seu desenvolvimento.

Com as funcionalidades e necessidades identificadas, com os cenários e a interação do utilizador modelados, e com as tecnologias para o desenvolvimento e a comunicação com o GitHub delineados, foi desenvolvida a ferramenta para a visualização dos resultados das métricas recolhidas da plataforma GitHub.

4. Aplicação e avaliação da ferramenta desenvolvida por utilizadores da plataforma GitHub e com conhecimento em métricas ágeis de desenvolvimento de *software*.

A ferramenta foi testada e avaliada por dois utilizadores com conhecimento e experiência em métricas ágeis de desenvolvimento de *software*. Solicitou-se que a testassem e a avaliassem de acordo com o objetivo principal do trabalho e com o auxílio de um questionário, construído de acordo com o modelo de qualidade de produto definido na ISO/IEC 25010, que compreende a avaliação de cinco critérios de qualidade: usabilidade, eficiência de desempenho, confiabilidade, segurança e adequação funcional (cf. Apêndice D). Constatou-se que a ferramenta teve resultados positivos em todos os critérios de qualidade. Através da análise das respostas percebeu-se que os utilizadores consideram a ferramenta intuitiva, simples, segura, com informações confiáveis, que agrega valor à equipa e empresa. As métricas existentes foram consideradas relevantes e de fácil análise e visualização e possibilitam acompanhar os projetos e as tarefas no dia-a-dia (cf. Apêndice E).

Uma boa ferramenta de visualização de métricas permite acompanhar e monitorizar o desempenho da equipa, indica onde estão as falhas no processo de trabalho, auxilia na tomada de decisões mais assertivas e possibilita a melhoria contínua da equipa e empresa. Desta forma, entende-se que o objetivo geral deste trabalho em elaborar uma ferramenta de visualização de métricas para acompanhar projetos e tarefas de desenvolvimento de *software* integrados na plataforma GitHub foi concretizado.

5.2. Contributos para a comunidade científica e empresarial

Devido à grande demanda de soluções tecnológicas, as empresas de *software* estão cada vez mais competitivas e precisam de adotar estratégias para serem mais produtivas e construírem produtos com altos padrões de qualidade. Face a este cenário, as métricas estão a ser adotadas como aliadas, já que fornecem dados reais para analisar e avaliar o produto e o seu processo de construção. Para auxiliar a visualização dessas métricas são utilizadas ferramentas que nem sempre atendem a essas necessidades, tornando relevante a realização deste estudo, com a elaboração de uma ferramenta simples e confiável de visualização de métricas importantes para as equipas.

A nível científico, este trabalho acrescenta valor à área de sistemas de informação, principalmente pela adoção de sistemas para a gestão de projetos e atividades de desenvolvimento de *software*. Ainda, contribui para a automatização de processos manuais com uma nova proposta de ferramenta para visualização de métricas ágeis de equipas de desenvolvimento de *software.* A automatização permite alcançar rapidamente a qualidade do produto, aumenta a quantidade entregue, diminui custos e erros, possibilita uma gestão segura e com qualidade, otimiza tempo e aumenta a competitividade da empresa (Alward et al., 2022). Já a nível empresarial, a adoção de métricas ágeis releva-se de extrema importância para acompanhar a qualidade e a produtividade do processo de desenvolvimento e do uso de uma ferramenta para visualização e para auxiliar a análise dos dados. Esses recursos proporcionam à empresa tomar decisões mais assertivas, identificar problemas e melhorias no produto e no processo de trabalho, melhorar a estimativa e o planeamento das tarefas e projetos, proporcionando uma melhoria contínua na equipa e empresa.

5.3. Limitações do estudo

Uma das principais limitações do presente estudo prende-se com a indisponibilidade das pessoas em responder ao questionário, em participar na sessão de *brainstorming* e em executar os testes, tornando-se uma tarefa morosa e difícil na conciliação de horários. Tal repercutiu-se numa amostra relativamente pequena que, embora tenha permitido compreender em detalhe os requisitos necessários e esperados pelos utilizadores, pode ter contribuído para que os resultados obtidos careçam de maior abrangência na aplicabilidade da ferramenta nas mais diversas áreas profissionais. O

tempo limitado para a realização do trabalho foi também um dos fatores que contribuiu para a dificuldade em angariar participantes.

5.4. Propostas de investigação futura

Embora tenham sido obtidos resultados relevantes com o presente estudo, é possível propor melhorias para investigações futuras. Em linha com o anteriormente mencionado, seria relevante obter parcerias com empresas de várias áreas com conhecimento em métricas ágeis de desenvolvimento de *software* de forma a permitir o estudo mais abrangente da aplicabilidade e funcionalidade da ferramenta. Tal permitiria também perceber a necessidade de ajustes à mesma de acordo com a área profissional.

Adicionalmente, poderão ser feitas melhorias à ferramenta de visualização dos resultados das métricas recolhidas da plataforma GitHub, tais como:

- Adicionar mais métricas ágeis relevantes para acompanhar o projeto e tarefa como a estimativa de Monte Carlo, *work item age*, eficiência de fluxo, *burndown* e *burn-up*;
- Integrar a ferramenta com outras plataformas de gestão de tarefas, para além do GitHub;
- Ajustar a ferramenta para visualizar dados do quadro Kanban customizados no GitHub;
- Permitir exportar as informações para Excel;
- Mais avaliadores para a ferramenta.

Referências Bibliográficas

Abrahamsson, P., Baskerville, R., Conboy, K., Fitzgerald, B., Morgan, L., & Wang, X. (2008). *Agile Processes in Software Engineering and Extreme Programming* (Springer, Vol. 9th).

Albino, R. D. (2017). *Métricas Ágeis: Obtenha melhores resultados em sua equipe* (Caso do Código).

Alward, Y., Singh, O., & Ansari, M. (2022). *Industrial automation and control systems development future and challenges. Journal of Information and Optimization Sciences*, 43, 71–83.

Al-Samarraie, H., & Hurmuzan, S. (2018). A review of brainstorming techniques in higher education. *Thinking Skills and Creativity*, *27*, 78–91. https://doi.org/10.1016/j.tsc.2017.12.002

Baumeister, H., Lichter, H., & Riebisch, M. (2017). *Agile Processes in Software Engineering and Extreme Programming* (Springer).

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., & Marick, B. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Beer, B. (2018). *Introducing GitHub: A Non-Technical Guide* (I. O'Reilly Media, Ed.).

B'Far, R. (2004). *Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML* (Cambridge University Press, Ed.).

Bhasin, T., Murray, A., & Storey, M. A. (2021). Student Experiences with GitHub and Stack Overflow: An Exploratory Study. *Proceedings - 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2021*, 81–90. https://doi.org/10.1109/CHASE52884.2021.00017

Bittner, K., & Spence, I. (2003). *Use Case Modeling* (Addison-Wesley Professional, Ed.).

Budacu, E., & Pocatilu, P. (2018). Real Time Agile Metrics for Measuring Team Performance. *Informatica Economica*, *22*(4/2018), 70–79. https://doi.org/10.12948/issn14531305/22.4.2018.06

Caroli, P. (2020). *Diagrama de Fluxo Cumulativo: Uma ferramenta valiosa para melhorar o fluxo de trabalho* (Caroli).

Chatterjee, R., Ahmed, A., & Anish, P. R. (2020). Identification and Classification of Architecturally Significant Functional Requirements. *Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020*, 9–17. https://doi.org/10.1109/AIRE51212.2020.00008

Chemuturi, M. (2010). *Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers* (J. Ross Publishing, Ed.).

Chemuturi, M. (2012). *Requirements Engineering and Management for Software Development Projects* (Springer, Ed.).

Chonoles, M. J., & Schardt, J. A. (2011). *UML 2 For Dummies* (John Wiley & Sons, Ed.).

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2014). *Three Metrics to Explore the Openness of GitHub projects*. http://arxiv.org/abs/1409.4253

Davis, C. (2015). *Agile Metrics in Action: How to measure and improve team performance* (Simon and Schuster, Ed.).

Dawson, C., & Straub, B. (2016). *Building Tools with GitHub: Customize Your Workflow* (B. MacDonald & M. Blanchette, Eds.).

Eisty, N. U., Thiruvathukal, G. K., & Carver, J. C. (2018). A survey of software metric use in research software development. *Proceedings - IEEE 14th International Conference on EScience, e-Science 2018*, 212–222. https://doi.org/10.1109/eScience.2018.00036

el Sheikh, A., & Alnoukari, M. (2012). *Bussiness Intelligence and Agile Methodologies for Knowledge - Based Organizations : Cross - Disciplinary Applications* (Business Science).

Fenton, N., & Bieman, J. (2014). *Software Metrics: A Rigorous and Practical Approach, Third Edition* (CRC Press).

Fernandes, J., & Machado, R. (2017). *Requisitos em Projetos de Software e de Sistemas de Informação* (Abril).

Galup, S., Dattero, R., & Quan, J. (2020). What do agile, lean, and ITIL mean to DevOps? *Communications of the ACM*, *63*(10), 48–53. https://doi.org/10.1145/3372114

GitHub Docs. (2022, January 28). https://docs.github.com/

Guthals, S., & Haack, P. (2019). *GitHub For Dummies* (John Wiley & Sons, Ed.; John Wiley & Sons).

Habibullah, K. M., & Horkoff, J. (2021). Non-functional Requirements for Machine Learning: Understanding Current Use and Challenges in Industry. *Proceedings of the IEEE International Conference on Requirements Engineering*, 13–23. https://doi.org/10.1109/RE51729.2021.00009

Hata, H., Novielli, N., Baltes, S., Kula, R. G., & Treude, C. (2022). GitHub Discussions: An exploratory study of early adoption. *Empirical Software Engineering*, *27*(1). https://doi.org/10.1007/s10664-021-10058-6

Hazzan, O., & Dubinsky, Y. (2007). *Agile Software Engineering*. Springer.

Heller, P. (2021). *Automating Workflows with GitHub Actions: Automate software development workflows and seamlessly deploy your applications using GitHub Actions* (Packt Publishing Ltd, Ed.).

Hemalatha, C., Sankaranarayanasamy, K., & Durairaaj, N. (2021). Lean and agile manufacturing for work-in-process (WIP) control. *Materials Today: Proceedings*, *46*, 10334–10338. https://doi.org/10.1016/j.matpr.2020.12.473

Islam, K. (2013). *Agile Methodology for developing e measuring learning: Training development for today's world* (AuthorHouse).

ISO/IEC 25010. (2022). *ISO 25000 software and data quality*. Https://Iso25000.Com/Index.Php/En/Iso-25000-Standards/Iso-25010.

Kalliamvakou, E., Damian, D., Blincoe, K., Singer, L., & German, D. M. (2015). Open source-style collaborative development practices in commercial projects using GitHub. *Proceedings - International Conference on Software Engineering*, *1*, 574–585. https://doi.org/10.1109/ICSE.2015.74

Kan, S. H. (2003). *Metrics and Models in Software Quality Engineering* (Addison-Wesley Professional, Ed.).

Keng, S., & Terry, H. (2000). *Unified Modeling Language: Systems Analysis, Design and Development Issues: Systems Analysis, Design and Development Issues* (Idea Group Inc (IGI), Ed.).

Kim, J., & Lee, S. (2021). An empirical study on using multi-labels for issues in github. *IEEE Access*, *9*, 134984–134997. https://doi.org/10.1109/ACCESS.2021.3116061

Kroeger, T. A., Davidson, N. J., & Cook, S. C. (2014). Understanding the characteristics of quality for software engineering processes: A Grounded Theory investigation. *Information and Software Technology*, *56*(2), 252–271. https://doi.org/10.1016/j.infsof.2013.10.003

Kupiainen, E., Mäntylä, M. v., & Itkonen, J. (2015). Using metrics in Agile and Lean software development - A systematic literature review of industrial studies. In *Information and Software Technology* (Vol. 62, Issue 1, pp. 143–163). Elsevier B.V. https://doi.org/10.1016/j.infsof.2015.02.005

LABIUTIL - Laboratório de Utilizabilidade da Universidade Federal de Santa Catarina. (2018). *Usabilidade: Uma versão atualizada da ErgoList desenvolvida pela UFSC*. Https://Usabilidade.Github.Io/.

Laplante, P. A., & Kassab, M. H. (2022). *Requirements Engineering for Software and Systems: Requirements Engineering for Software and Systems* (CRC Press, Ed.; 4st ed.).

Lehner, F. (2013). *Software Metrics: Research and Practice in Software Measurement* (Springer).

Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development* (Inc. " "O'Reilly Media, Ed.; 2ª).

Mate, J. L., & Silva, A. (2005). *Requirements Engineering for Sociotechnical Systems* (Idea Group Inc (IGI), Ed.; Idea Group Inc (IGI)).

Matthies, C., Kowark, T., Uflacker, M., & Plattner, H. (2016). Agile metrics for a university software engineering course. *Proceedings - Frontiers in Education Conference, FIE, 2016-November*. https://doi.org/10.1109/FIE.2016.7757684

Mendonc, M. G., & Basili, V. R. (n.d.). *Validation of an Approach for Improving Existing Measurement Frameworks*.

Meyers, P. J. (2009). *User Effect*. Http://Drpete.Co/Pdf/Checklist.Pdf.

Miles, R., & Hamilton, K. (2006). *Learning UML 2.0: A Pragmatic Introduction to UML* (Inc. O'Reilly Media, Ed.).

Mishra, J., & Mohanty, A. (2011). *Software Engineering* (Pearson Education India, Ed.).

Miyashita, Y., Hazeyama, A., Yamada, Y., & Hashiura, H. (n.d.). *Design of the Inspection Process Using the GitHub Flow in Project Based Learning for Software Engineering and Its Practice*.

Muniz, A., Irigoyen, A., Mafra, C., Trierveiler, F., & Villanova, G. (2021). *Jornada Kanban na prática: unindo teoria e prática para acelerar o aprendizado para quem está iniciando* (Brasport).

Neumann, A., Laranjeiro, N., & Bernardino, J. (2021). An Analysis of Public REST Web Service APIs. *IEEE Transactions on Services Computing*, *14*(4), 957–970. https://doi.org/10.1109/TSC.2018.2847344

Niemi, T., Gallay, O., & Hameri, A.-P. (2021). Technical Note: Mean Lead-Time as a Real-Time Key Performance Indicator Subject Areas: Finite Observation Time Window, Lead-Time Analysis, and Performance Measure. In *Decision Sciences* (Vol. 52).

Ortu, M., Destefanis, G., Graziotin, D., Marchesi, M., & Tonelli, R. (2020). How do you Propose Your Code Changes? Empirical Analysis of Affect Metrics of

Pull Requests on GitHub. *IEEE Access*, *8*, 110897–110907. https://doi.org/10.1109/ACCESS.2020.3002663

Ozkan, N., Gok, M. S., & Kose, B. O. (2020). Towards a Better Understanding of Agile Mindset by Using Principles of Agile Methods. *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020*, 721–730. https://doi.org/10.15439/2020F46

Pachouly, J., Ahirrao, S., & Kotecha, K. (2022). SDPTool : A tool for creating datasets and software defect predictions. *SoftwareX*, *18*, 101036. https://doi.org/10.1016/j.softx.2022.101036

Paing, Y., Vélez, T. C., & Khatchadourian, R. (2022). *QuerTCI: A Tool Integrating GitHub Issue Querying with Comment Classification*. http://arxiv.org/abs/2202.08761

Patkai, N. (2022). *Web Usability Checklist*. Https://Teamsuccess.Io/UX.

Pohl, K., & Rupp, C. (2016). *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam* (2a ed.). Rocky Nook.

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F. da V., Fufezan, C., Ternent, T., Eglen, S. J., Katz, D. S., Pollard, T. J., Konovalov, A., Flight, R. M., Blin, K., & Vizcaíno, J. A. (2016). Ten Simple Rules for Taking Advantage of Git and GitHub. In *PLoS Computational Biology* (Vol. 12, Issue 7). Public Library of Science. https://doi.org/10.1371/journal.pcbi.1004947

Pipinellis, A. (2018). *GitHub Essentials: Unleash the power of collaborative development workflows using GitHub* (Packt Publishing Ltd, Ed.; 2nd ed.).

Presser, S., Rothgeb, J. M., Couper, M. P., Lessler, J. T., Martin, E., Martin, J., & Singer, E. (2004). *Methods for Testing and Evaluating Survey Questionnaires* (John Wiley & Sons, Ed.).

Ram, P., Rodriguez, P., Oivo, M., & Martinez-Fernandez, S. (2019). Success factors for effective process metrics operationalization in agile software development: A multiple case study. *Proceedings - 2019 IEEE/ACM International Conference on Software and System Processes, ICSSP 2019*, 14–23. https://doi.org/10.1109/ICSSP.2019.00013

Ren, L., Zhou, S., & Kästner, C. (2018). Poster: Forks insight: Providing an overview of GitHub forks. *Proceedings - International Conference on Software Engineering*, 179–180. https://doi.org/10.1145/3183440.3195085

Rusk, D., & Coady, Y. (2014). Location-based analysis of developers and technologies on GitHub. *Proceedings - 2014 IEEE 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2014*, 681–685. https://doi.org/10.1109/WAINA.2014.110

Sage, A. P., & Rouse, W. B. (2014). *Handbook of Systems Engineering and Management: Wiley series in systems engineering and management* (John Wiley & Sons, Ed.; 2nd ed.).

Sarrab, M., & M. Hussain Rehman, O. (2014). *Empirical study of open source software selection for adoption,based on software quality characteristics*. Advances in Engineering Software, 69, 1–11.

Şeker, A., Diri, B., & Arslan, H. (2021). New developer metrics for open source software development challenges: An empirical study of project recommendation systems. *Applied Sciences (Switzerland)*, *11*(3), 1–26. https://doi.org/10.3390/app11030920

Shimada, N., Xiao, T., Hata, H., Treude, C., & Matsumoto, K. (2022). *GitHub Sponsors: Exploring a New Way to Contribute to Open Source*. https://doi.org/10.1145/3510003.3510116

Singh, S. R. (2007). *Information System Management* (APH Publishing, Ed.).

Spinellis, D., Kotti, Z., & Mockus, A. (2020). *A Dataset for GitHub Repository Deduplication: Extended Description*. http://arxiv.org/abs/2002.02314

Subramanian, V. N., Rehman, I., Nagappan, M., & Kula, R. G. (2022). Analyzing First Contributions on GitHub: What Do Newcomers Do? *IEEE Software*, *39*(1), 93–101. https://doi.org/10.1109/MS.2020.3041241

Suresh, Y., Pati, J., & Rath, S. K. (2012). Effectiveness of Software Metrics for Object-oriented System. *Procedia Technology*, *6*, 420–427. https://doi.org/10.1016/j.protcy.2012.10.050

Venkatesh, V., Thong, J. Y. L., Chan, F. K. Y., Hoehle, H., & Spohrer, K. (2020). How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills. *Information Systems Journal*, *30*(4), 733–761. https://doi.org/10.1111/isj.12282

Vijayasarathy, L., & Turk, D. (2012). Drivers of agile software development use: Dialectic interplay between benefits and hindrances. *Information and Software Technology*, *54*(2), 137–148. https://doi.org/10.1016/j.infsof.2011.08.003

Webb, A. (2015). *Agile Project Management Methodology for Beginners: Scrum Project Management for Beginners* (Smashwords).

Zhou, J., Wang, S., Bezemer, C. P., Zou, Y., & Hassan, A. E. (2021). Studying the Association between Bountysource Bounties and the Issue-Addressing Likelihood of GitHub Issue Reports. *IEEE Transactions on Software Engineering*, *47*(12), 2919–2933. https://doi.org/10.1109/TSE.2020.2974469

APÊNDICES

Apêndice A. Princípios do manifesto ágil (Fonte: Beck et al., 2001)

Princípios do manifesto ágil	
1	A prioridade é, desde as primeiras etapas do projeto, satisfazer o cliente através da entrega rápida e contínua de *software* com valor.
2	Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente;
3	Fornecer frequentemente *software* funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos;
4	O cliente e a equipa de desenvolvimento devem trabalhar juntos, diariamente, durante o decorrer do projeto
5	Desenvolver projetos com base em indivíduos motivados, dando-lhes o ambiente e o apoio de que necessitam, confiando que irão cumprir os objetivos
6	O método mais eficaz de passar informação a uma equipa de desenvolvimento é através de conversação pessoal e direta
7	A principal medida de progresso é a entrega de *software* funcional
8	Os processos ágeis promovem o desenvolvimento sustentável. Os promotores, a equipa e os utilizadores deverão ser capazes de manter, indefinidamente, um ritmo constante
9	A atenção permanente à excelência técnica e um bom desenho da solução aumentam a agilidade
10	Simplicidade – a arte de maximizar a quantidade de trabalho que não é realizado – é essencial
11	As melhores arquiteturas, requisitos e desenhos surgem de equipas auto-organizadas
12	A equipa reflete regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e as adaptações necessárias

Apêndice B. Questionário sobre métricas ágeis

Métricas Ágeis - 20 minutos

Questionário sobre Métricas Ágeis

Caro Sr(a).,

O presente questionário faz parte de um estudo realizado no âmbito da dissertação de Mestrado em Gestão de Sistemas de Informação do Instituto Superior de Ciências do Trabalho e da Empresa – Instituto Universitário de Lisboa (ISCTE-IUL).

Pretende-se elaborar um dashboard, relevante para equipas de desenvolvimento de software que trabalham segundo as metodologias ágeis, a partir de métricas extraídas da plataforma GitHub.

Não existem respostas certas ou erradas, interessa-nos apenas a sua resposta sincera e espontânea.

O questionário é composto por 15 questões.

Estima-se que o tempo médio de resposta seja de 20 min.

Obrigado pela sua colaboração.

Bruna Santos (Bruna_Amorim_Santos@iscte-iul.pt)

Informe seu nome completo *

Texto de resposta curta

Informe seu cargo atual *

Texto de resposta curta

Qual ferramenta sua empresa utiliza para acompanhamento de projetos e tarefas? *

Texto de resposta curta

A ferramenta é de fácil uso? *

- ◯ Sim
- ◯ Não

A ferramenta é útil para sua equipa no dia-a-dia? *

- ◯ Sim
- ◯ Não

Você utiliza a ferramenta para acompanhar as tarefas e projetos de sua equipa?

- ◯ Sim
- ◯ Não

As informações fornecidas pela ferramenta são suficientes para realizar o planejamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa?

- ◯ Sim
- ◯ Não

Quais informações faltam na ferramenta para auxiliar no planejamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa?

Texto de resposta longa

Quais os pontos positivos da ferramenta?

Texto de resposta longa

Quais os pontos negativos da ferramenta?

Texto de resposta longa

O que é importante em uma ferramenta para acompanhamento de projetos e tarefas no contexto ágil (Como ela deve ser, o que ela deve ter)?

Texto de resposta longa

Quais dessas métricas ágeis são usadas por sua empresa/equipa? *

- [ ] Nenhuma métrica
- [ ] OKR
- [ ] WIP (Work in progress)
- [ ] Lead time
- [ ] Cycle Time
- [ ] Throughput
- [ ] Gráficos de burndown e burn-up
- [ ] Diagrama de fluxo cumulativo (CFD)
- [ ] Estimativa de Monte Carlo
- [ ] Outros...

Onde são visualizadas as métricas?

- ( ) Na própria ferramenta
- ( ) Outros...

A ferramenta possui todos os dados necessários para as métricas?

- ○ Sim
- ○ Não

Quais dessas métricas ágeis você acha importante usar em sua equipa? *

- ☐ Nenhuma métrica
- ☐ OKR
- ☐ WIP (Work in progress)
- ☐ Lead time
- ☐ Cycle Time
- ☐ Throughput
- ☐ Gráficos de burndown e burn-up
- ☐ Diagrama de fluxo cumulativo (CFD)
- ☐ Estimativa de Monte Carlo
- ☐ Outros...

Apêndice C. Resumo das respostas do questionário sobre métricas

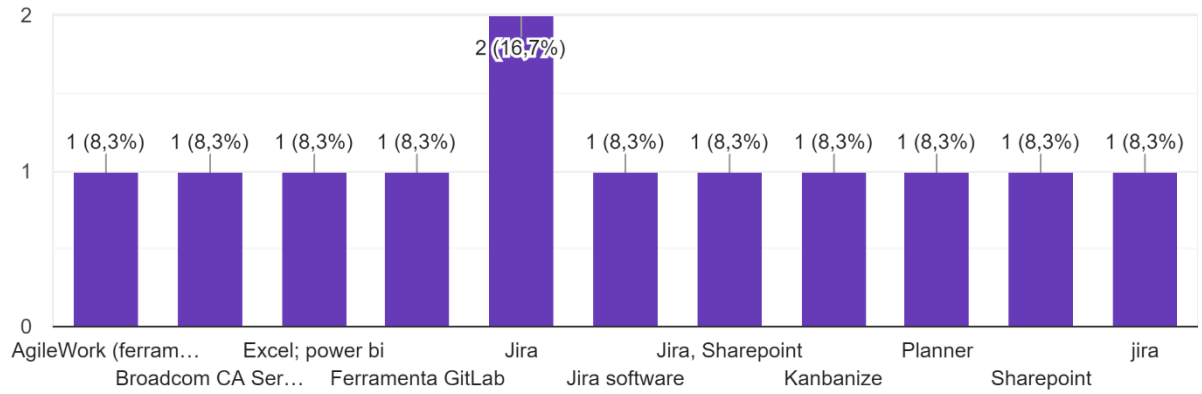
Informe seu cargo atual

12 respostas

Analista de Negócios
Application Support Engineer
Analista de Testes
Product Manager
Agile Master
Desenvolvedor
Desenvolvedor
Tester
Analista Funcional
Data analyst
Infrastructure Consultant
Líder de projetos

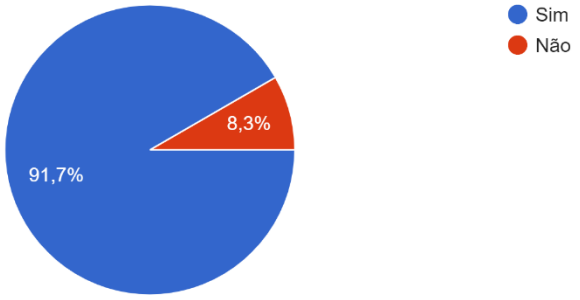
Qual ferramenta sua empresa utiliza para acompanhamento de projetos e tarefas?

12 respostas



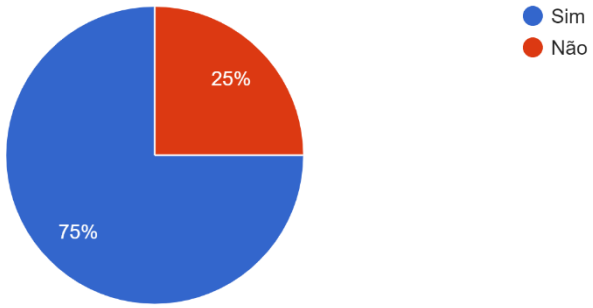
A ferramenta é de fácil uso?

12 respostas



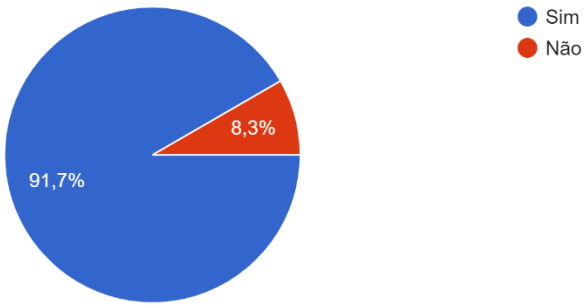
A ferramenta é útil para sua equipa no dia-a-dia?

12 respostas



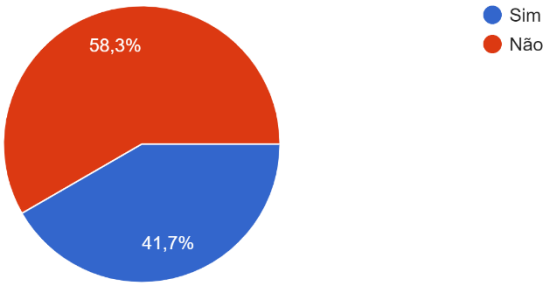
Você utiliza a ferramenta para acompanhar as tarefas e projetos de sua equipa?

12 respostas



As informações fornecidas pela ferramenta são suficientes para realizar o planejamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa?

12 respostas



Quais informações faltam na ferramenta para auxiliar no planejamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa?

9 respostas

A programação do sharepoint é difícil. Apenas algumas pessoas conseguiram entender o funcionamento
Entrar em detalhe como tempos, orientação de reuniões, existir uma area de personal data
Da forma como foi customizada na minha empresa, não mostra estimativas muito claras
A ferramenta é completa
Talvez se ela tivesse Burnup/Donw
Faltam gráficos para acompanhar o projeto
Menos subtarefas de forma a que não tenha de estar sempre a atualizar cada uma delas
Nenhuma, a ferramenta é completa.
Actionable Agile e planilhas

Quais os pontos positivos da ferramenta?

10 respostas

Visualização fácil
Tarefa a realizar e ponto de situação
É boa apenas para o registro de horas trabalhas
acompanhamento das tarefas/projetos, exportação dos dados, geração de gráficos, automatização, customização de campos
Facilidade de uso

A usabilidade da ferramenta é boa, possui varias informações que minha equipe necessita, permite pesquisar por várias informações e a ferramenta armazena várias informações importantes
Tudo acessível
Objetiva, fácil de utilizar (friendly), rápida/leve e completa.
Gestão de IN e CH
Métricas diversas

Quais os pontos negativos da ferramenta?

10 respostas

Dificuldade na programação
Não ter personal data
Precisa ser muito bem customizada, pois trás muitas informações irrelevantes
por ser uma ferramenta complexa, a curva de aprendizado é grande; existem muitas configurações por vários caminhos, então é necessário conhecer bem o jira
Nenhum... talvez a métrica Scrum poderia ter
Sinto um pouco de dificuldade em acompanhar os projetos e também podia ter mais gráficos, permitindo fácil visualização e análise dos projetos
Muitas subtarefas
Não vejo atualmente.
Muitos bugs
Pouco caro para grandes empresas

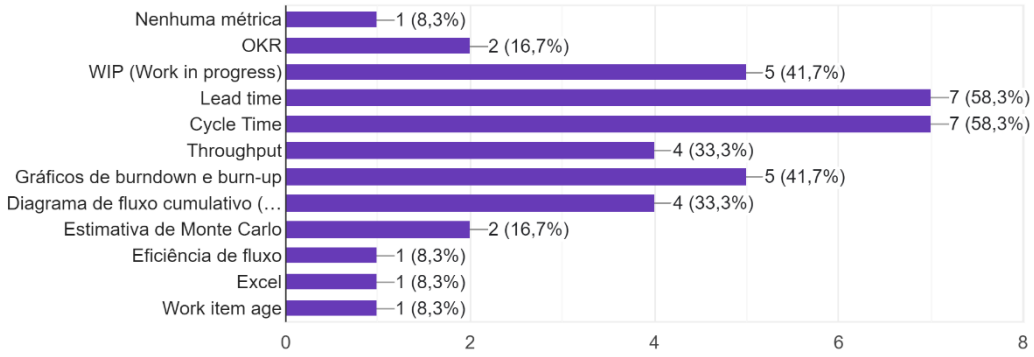
O que é importante em uma ferramenta para acompanhamento de projetos e tarefas no contexto ágil (Como ela deve ser, o que ela deve ter)?

10 respostas

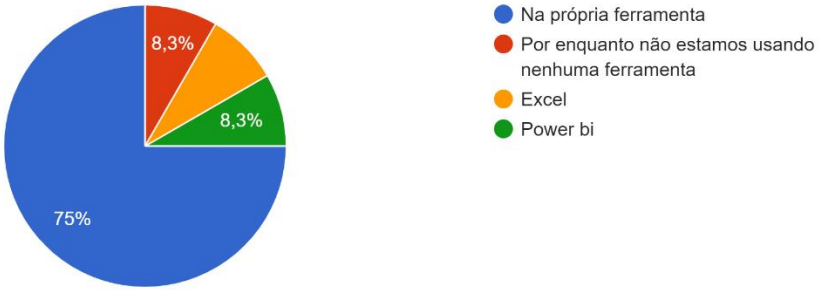
Facilidade no manuseio
Deve ser dividida por estado de processo, quem está envolvido, duração, se é prioritária ou não
Uma visão de estimativa de tempo, com possibilidade de acompanhamento em tempo real do andamento das tarefas
A ferramenta deve ser adaptável a qualquer processo de trabalho, e permitir extrair de forma clara as informações inseridas nela. Também deve permitir gerenciar os projetos até o nível de tarefa. Ela deve gerar relatórios com as métricas mais utilizadas do mercado (citadas aqui no questionário) sendo capaz de fornecer excelentes relatórios.
Deve ser de fácil uso, design atraente e com recursos de métricas facilitados.
Informações corretas e atualizadas, conter as informações necessárias para o acompanhamento, gráficos que facilitem a análise e visualização, ferramenta ser fácil de usar e a possibilidade de filtros e pesquisas

Melhor organizado e menos subtarefas
Friendly, ter apenas o que é realmente necessário, ser desenvolvida a pensar em quem vai realmente utilizar e a ter como base o mindset agile.
Ser intuitiva
Fácil importação de dados de outras ferramentas. Filtros diversos, customização na visualização

Quais dessas métricas ágeis são usadas por sua empresa/equipa?
12 respostas

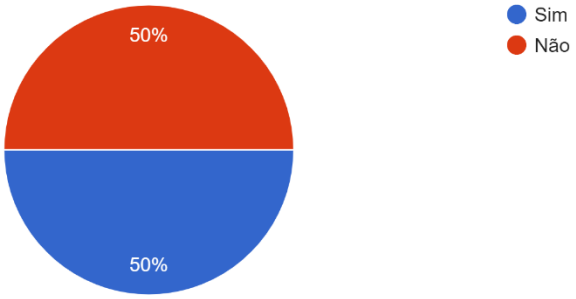


Onde são visualizadas as métricas?
12 respostas



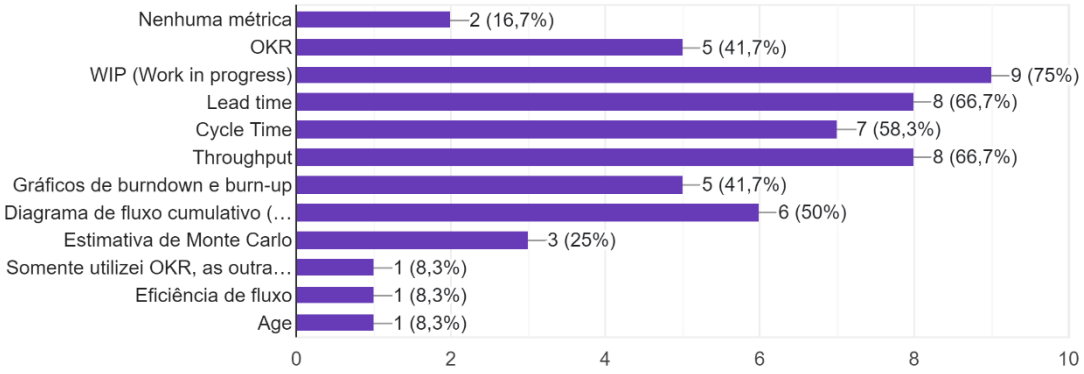
A ferramenta possui todos os dados necessários para as métricas?

12 respostas

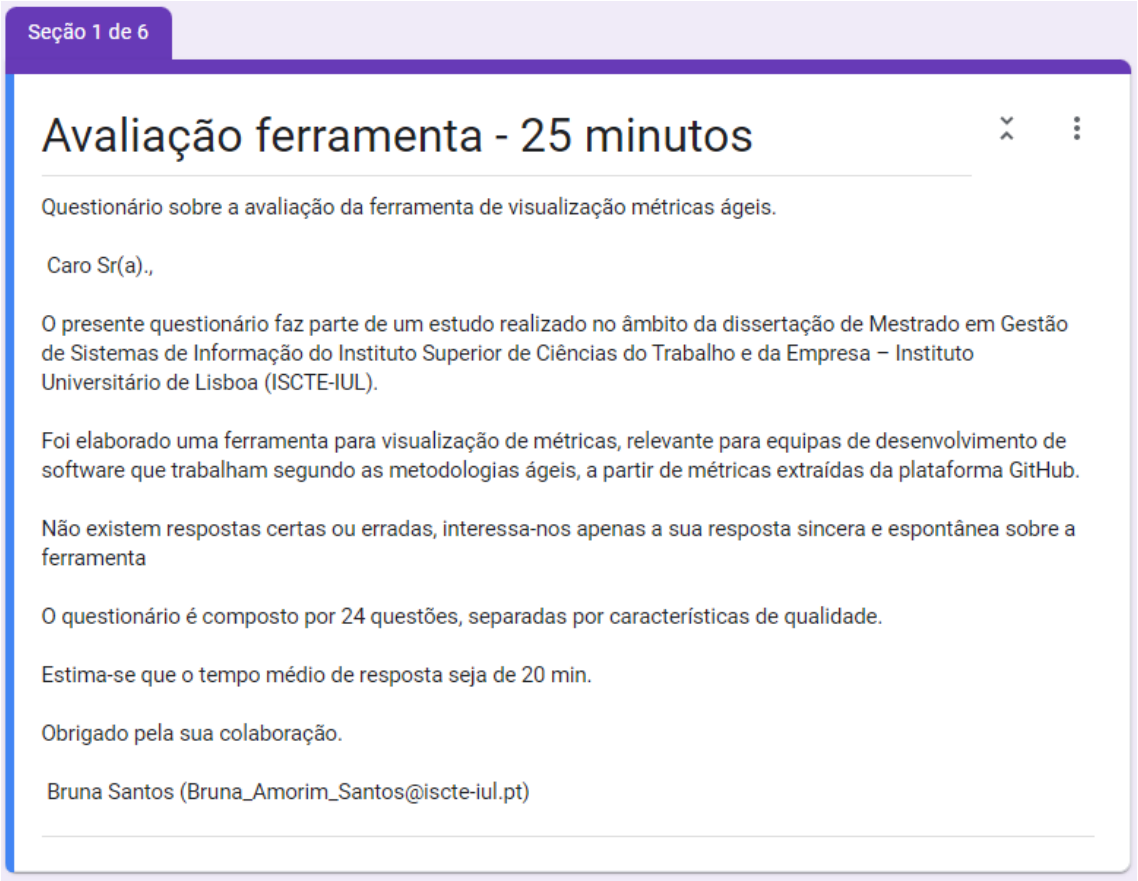


Quais dessas métricas ágeis você acha importante usar em sua equipa?

12 respostas



Apêndice D. Questionário sobre a avaliação da ferramenta de visualização de métricas ágeis



Seção 1 de 6

Avaliação ferramenta - 25 minutos

Questionário sobre a avaliação da ferramenta de visualização métricas ágeis.

Caro Sr(a).,

O presente questionário faz parte de um estudo realizado no âmbito da dissertação de Mestrado em Gestão de Sistemas de Informação do Instituto Superior de Ciências do Trabalho e da Empresa – Instituto Universitário de Lisboa (ISCTE-IUL).

Foi elaborado uma ferramenta para visualização de métricas, relevante para equipas de desenvolvimento de software que trabalham segundo as metodologias ágeis, a partir de métricas extraídas da plataforma GitHub.

Não existem respostas certas ou erradas, interessa-nos apenas a sua resposta sincera e espontânea sobre a ferramenta

O questionário é composto por 24 questões, separadas por características de qualidade.

Estima-se que o tempo médio de resposta seja de 20 min.

Obrigado pela sua colaboração.

Bruna Santos (Bruna_Amorim_Santos@iscte-iul.pt)

Nome *

Texto de resposta curta

Cargo *

Texto de resposta curta

Empresa *

Texto de resposta curta

Seção 2 de 6

Usabilidade

Grau em que um produto ou sistema pode ser usado por utilizadores para atingir objetivos específicos com eficácia, eficiência e satisfação.

A ferramenta é intuitiva, simples e fácil de usar? *

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

As informações disponíveis agregam valor ao utilizador? Informações desnecessários são evitadas? *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Confordo Totalmente

O conteudo é escrito com termos comuns e de fácil entendimento? *

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

A ferramenta pode ser personalizada por utilizador? *

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

Os gráficos são de fácil visualização e análise? *

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

Seção 3 de 6

Eficiência de desempenho

Essa característica representa o desempenho em relação à quantidade de recursos utilizados nas condições estabelecidas.

O tempo de carregamento da ferramenta é adequado? *

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

O tempo de exibição dos gráficos é adequado? *

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

O conteúdo é atualizado sempre que solicitado pelo utilizador? *

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

Seção 4 de 6

Confiabilidade

Grau em que um sistema, produto ou componente executa funções sob condições especificadas por um período de tempo determinado.

A ferramenta NÃO apresenta falhas inesperadas? *

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

A ferramenta apresenta dados confiáveis?

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

Seção 5 de 6

Segurança

Grau em que um produto ou sistema protege informações e dados para que pessoas, outros produtos ou sistemas tenham acesso apropriado aos seus tipos e níveis de autorização.

Apenas utilizadores autorizados conseguem usar o sistema? *

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

O utilizador consegue visualizar informações somente dos seus repositórios e projetos? *

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

Seção 6 de 6

Adequação Funcional

Característica que representa o grau em que um produto ou sistema fornece funções que atendem às necessidades declaradas e implícitas quando usado sob condições especificadas.

A ferramenta é útil para acompanhar as tarefas e projetos da equipa no dia-a-dia? *

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

As informações fornecidas são suficientes para realizar o planeamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa? *

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

A ferramenta permite visualizar a quantidade de tarefas em cada etapa do board? *

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

A ferramenta possui informações suficientes para a verificar o tempo total do ciclo de uma tarefa (Lead Time)? *

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

A ferramenta indica corretamente as tarefas em progresso (estado work in progress) ? *

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

Através da ferramenta é possível visualizar a quantidade de tarefas entregue em um período (Throughput)? *

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

Através da ferramenta é possível visualizar o tempo de desenvolvimento de uma tarefa (Cycle Time)? *

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

A ferramenta permite verificar, através do gráfico CFD, os gargalos e prever instabilidades no fluxo de trabalho? *

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

Comentários/Sugestões

Texto de resposta longa

Apêndice E. Resumos das respostas do questionário sobre a avaliação da ferramenta de visualização de métricas ágeis

Cargo

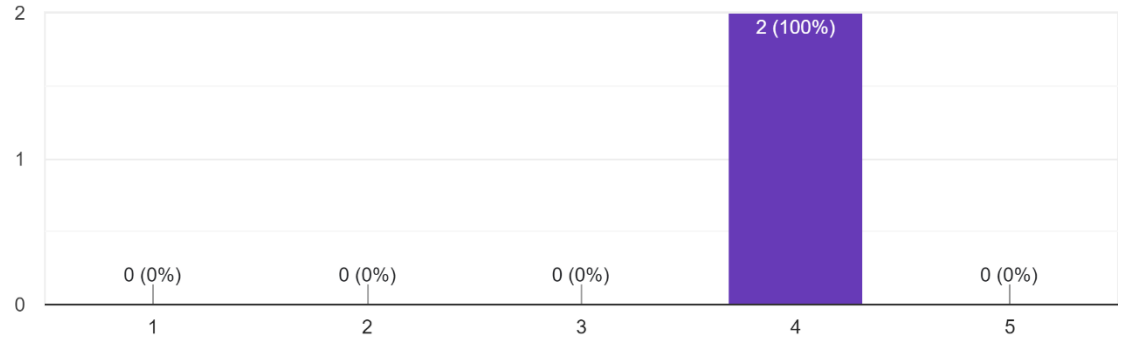
2 respostas

Analista de Testes
Product Manager

Usabilidade

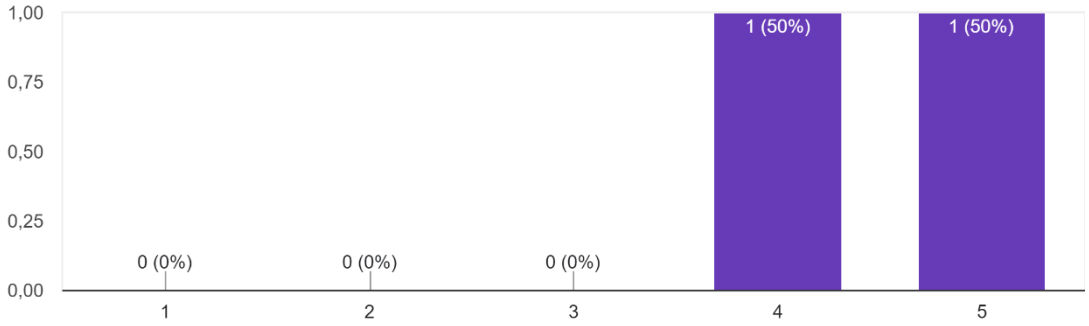
A ferramenta é intuitiva, simples e fácil de usar?

2 respostas



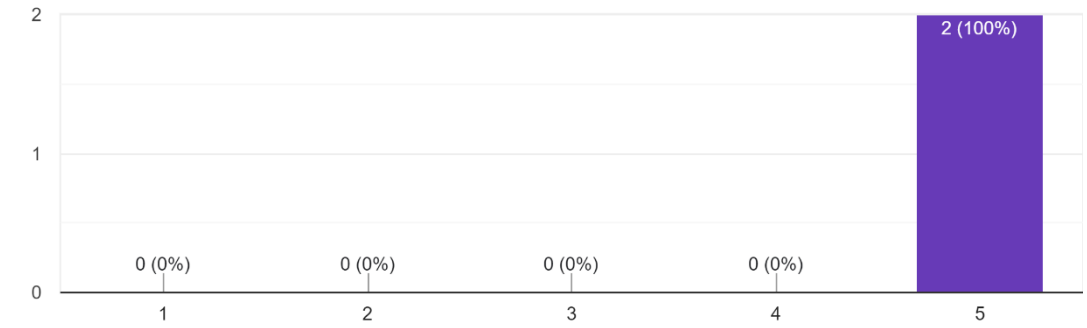
As informações disponíveis agregam valor ao utilizador? Informações desnecessários são evitadas?

2 respostas



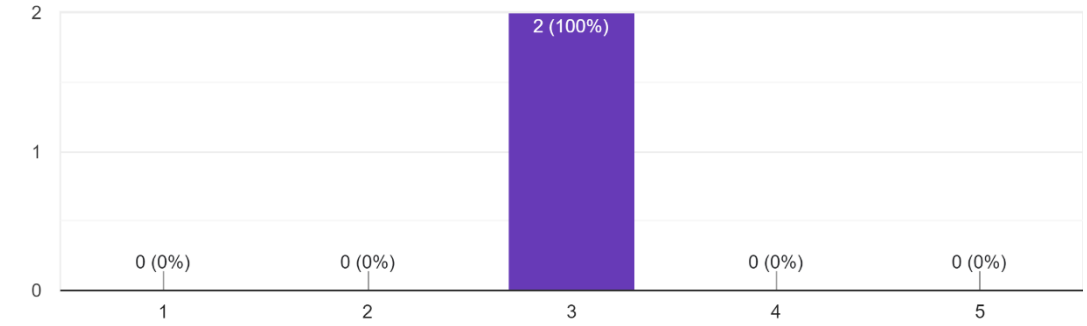
O conteudo é escrito com termos comuns e de fácil entendimento?

2 respostas



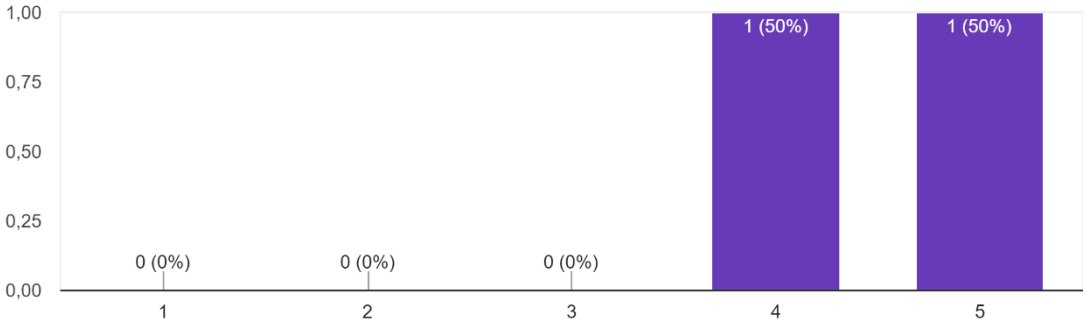
A ferramenta pode ser personalizada por utilizador?

2 respostas



Os gráficos são de fácil visualização e análise?

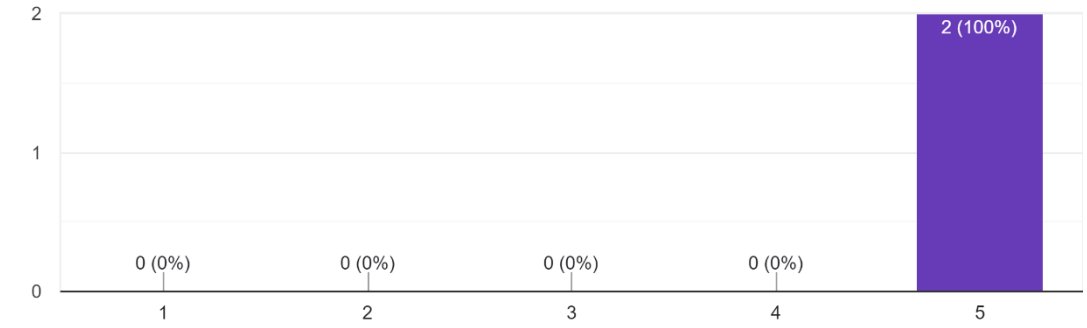
2 respostas



Eficiência de desempenho

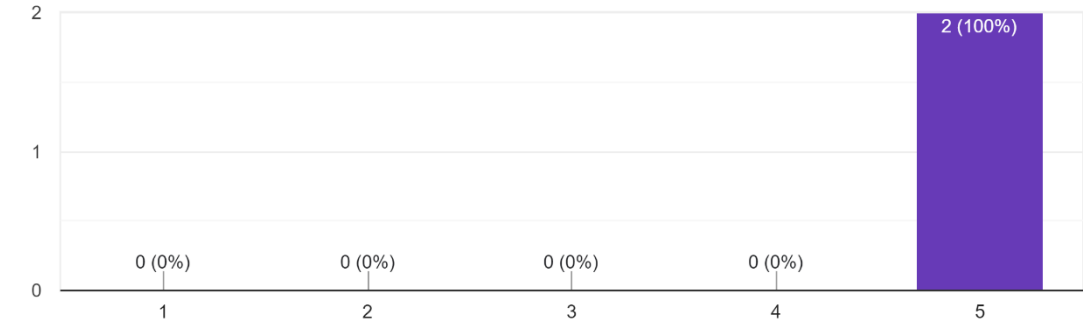
O tempo de carregamento da ferramenta é adequado?

2 respostas



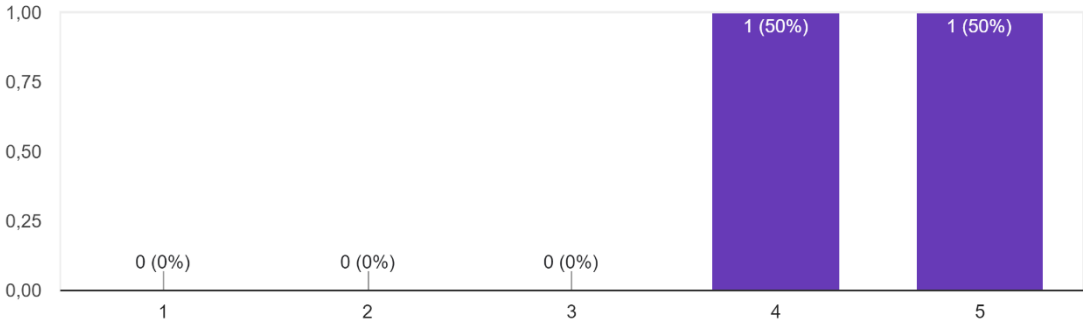
O tempo de exibição dos gráficos é adequado?

2 respostas



O conteúdo é atualizado sempre que solicitado pelo utilizador?

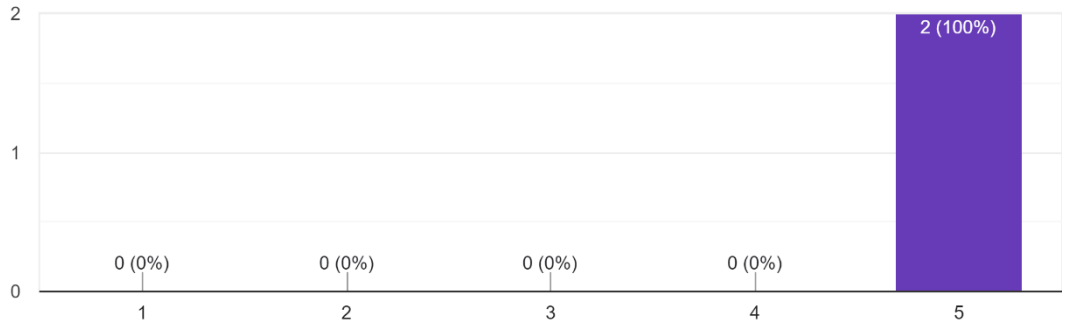
2 respostas



Confiabilidade

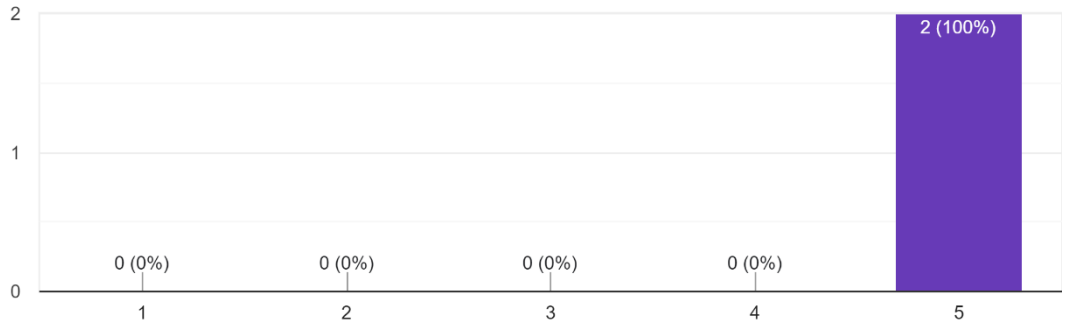
A ferramenta NÃO apresenta falhas inesperadas?

2 respostas



A ferramenta apresenta dados confiáveis?

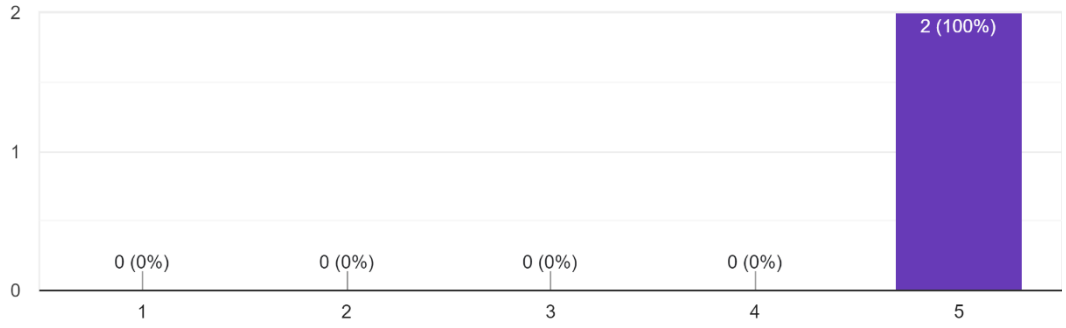
2 respostas



Segurança

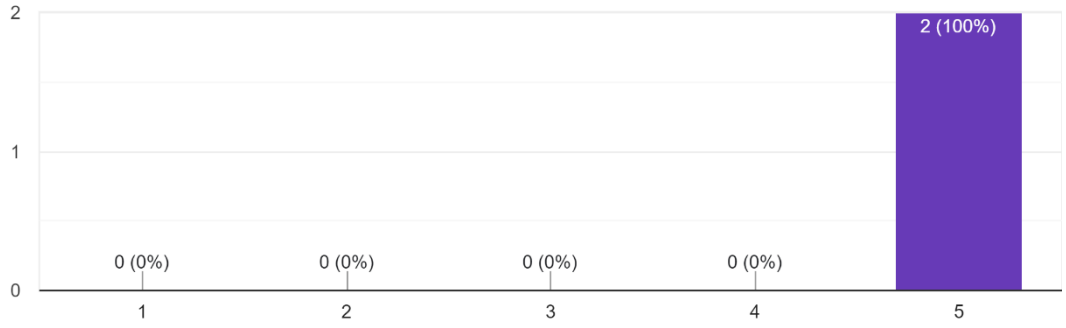
Apenas utilizadores autorizados conseguem usar o sistema?

2 respostas



O utilizador consegue visualizar informações somente dos seus repositórios e projetos?

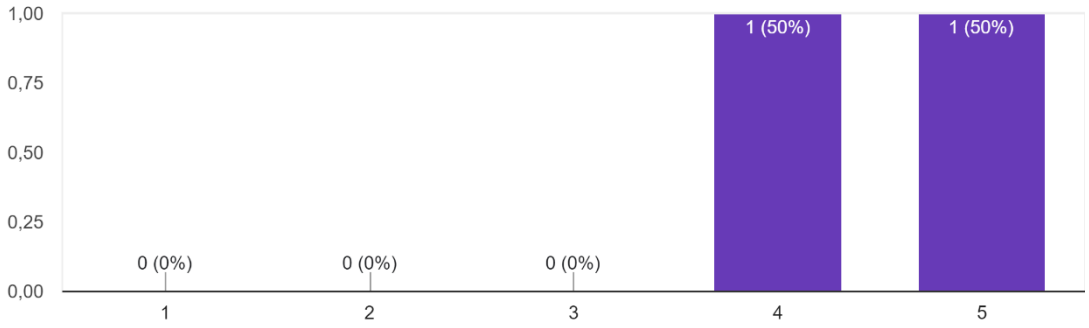
2 respostas



Adequação Funcional

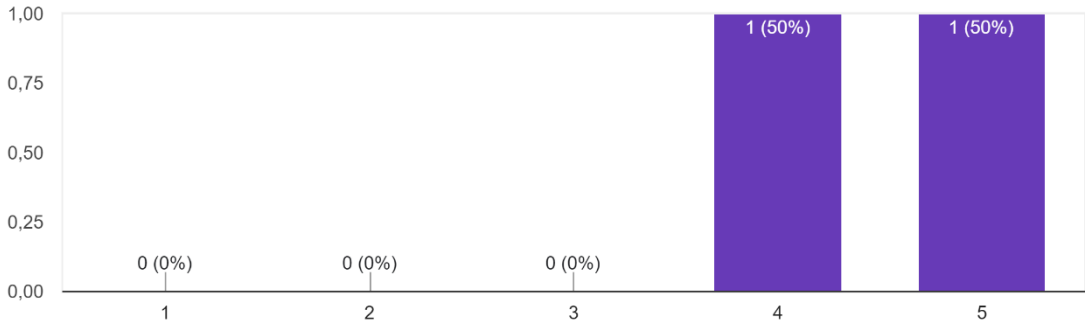
A ferramenta é útil para acompanhar as tarefas e projetos da equipa no dia-a-dia?

2 respostas



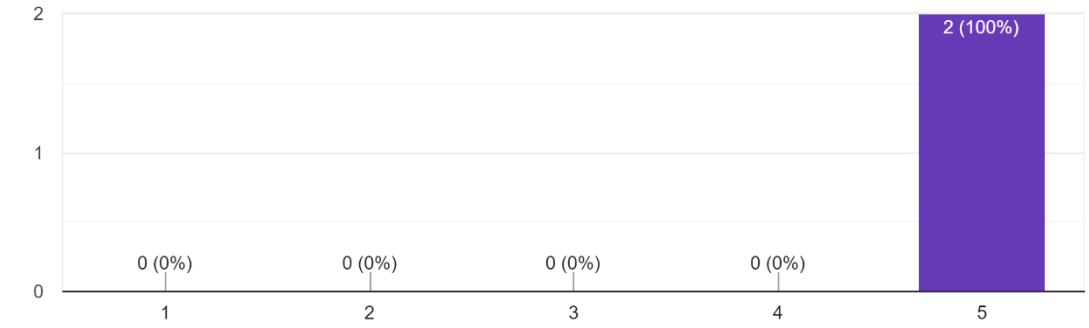
As informações fornecidas são suficientes para realizar o planeamento, estimativas e acompanhamentos do projeto e das tarefas de sua equipa?

2 respostas



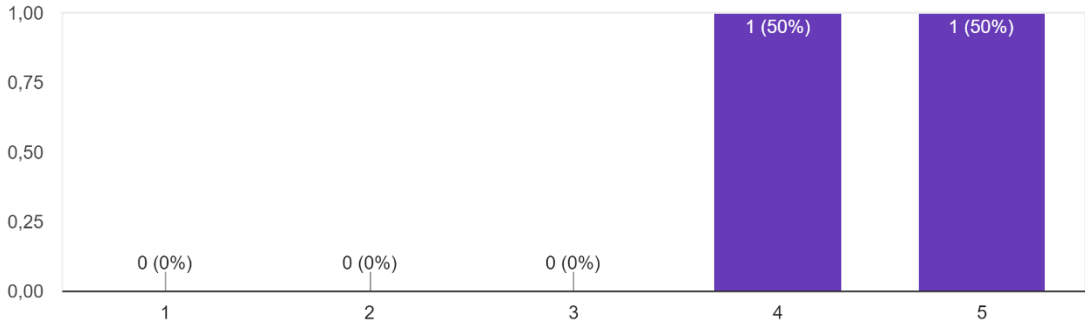
A ferramenta permite visualizar a quantidade de tarefas em cada etapa do board?

2 respostas



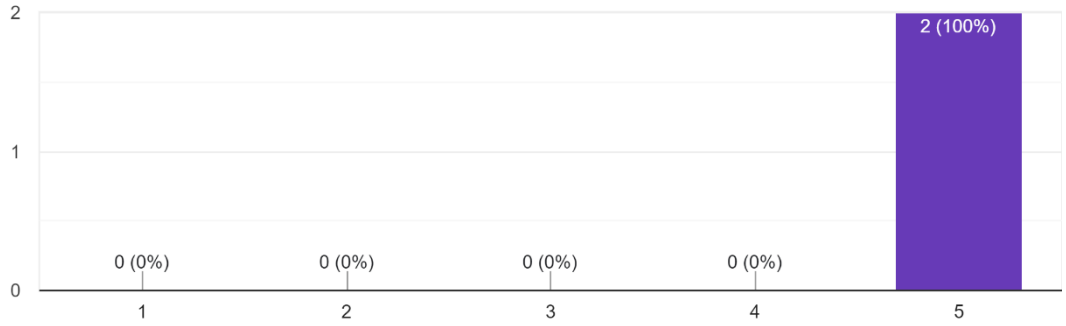
A ferramenta possui informações suficientes para a verificar o tempo total do ciclo de uma tarefa (Lead Time)?

2 respostas



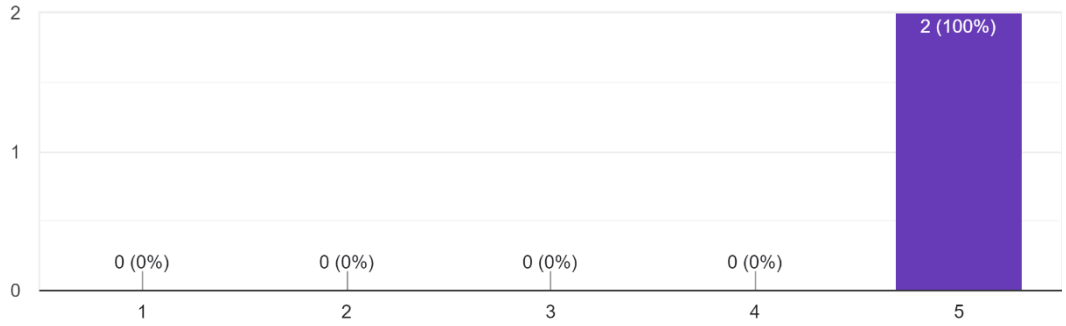
A ferramenta indica corretamente as tarefas em progresso (estado work in progress) ?

2 respostas



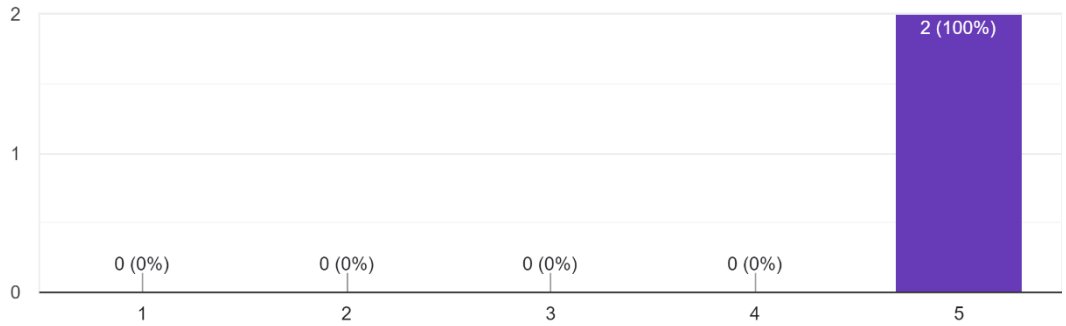
Através da ferramenta é possível visualizar a quantidade de tarefas entregue em um período (Throughput)?

2 respostas



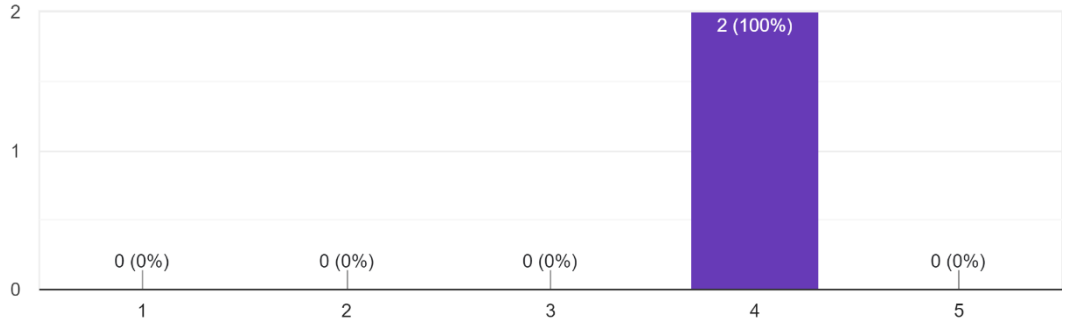
Através da ferramenta é possível visualizar o tempo de desenvolvimento de uma tarefa (Cycle Time)?

2 respostas



A ferramenta permite verificar, através do gráfico CFD, os gargalos e prever instabilidades no fluxo de trabalho?

2 respostas

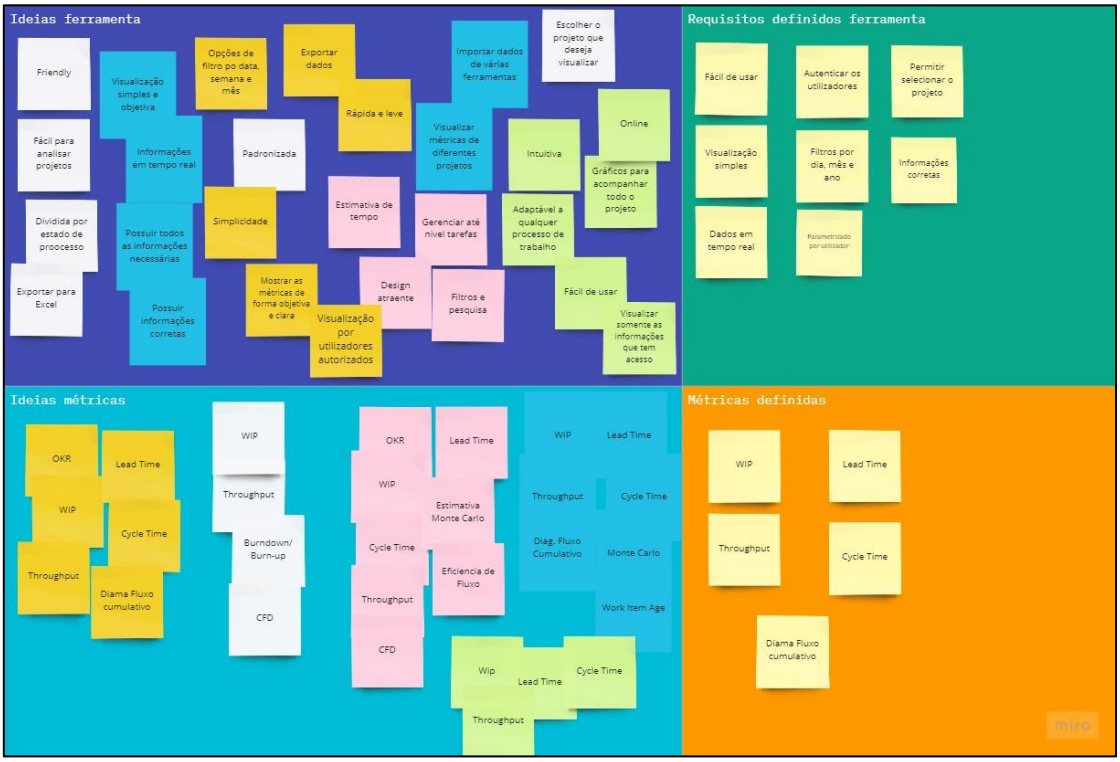


Comentários/Sugestões

1 resposta

Acredito que a ferramenta seja útil para minha equipe. Os gráficos são claros e fáceis de analisar.

Apêndice F. Resultado da sessão de *brainstorming*



Apêndice G. Repositório do GitHub referente ao projeto

A ferramenta desenvolvida encontra-se no seguinte repositório do GitHub: https://github.com/brunaamorims/metrics_ISCTE.